INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2026IPPAX003

Thèse de doctorat

ÉCOLE POLYTECHNIQUE

IP PARIS

# The Structure of Information in Distributed Robotics

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Mathématiques et informatique

Thèse présentée et soutenue à Palaiseau, le 22 janvier 2026, par

**BERNARDO HUMMES FLORES**

Composition du Jury :

| | |
|---|---|
| Uli Fahrenberg<br>Professeur, Université Paris-Saclay (LMF) | Rapporteur, Président |
| Thomas Ågotnes<br>Professeur, University of Bergen | Rapporteur |
| Georg Struth<br>Professeur, University of Sheffield | Rapporteur |
| Ulrich Schmid<br>Professeur émérite, TU Wien | Examinateur |
| Hans van Ditmarsch<br>Directeur de recherche, CNRS (IRIT) | Examinateur |
| Nobuko Yoshida<br>Professeure, University of Oxford | Examinatrice |
| Éric Goubault<br>Professeur, École polytechnique (LIX) | Directeur de thèse |
| Luc Jaulin<br>Professeur, ENSTA (Lab-STICC) | Co-directeur de thèse |
| Sylvie Putot<br>Professeure, École Polytechnique (LIX) | Co-directrice de thèse |

## RÉSUMÉ

La robotique distribuée se situe à l'intersection de la robotique mobile et du calcul distribué. La robotique mobile considère d'abord la physique, où un robot est soumis à des forces et le défi consiste à le diriger malgré des capteurs bruités et des actionneurs imprécis. Le calcul distribué se focalise sur l'information, où le défi est de garantir un comportement collectif malgré le fait que chaque processus s'exécute avec seulement sa vue locale. Malgré ces hypothèses différentes, les deux communautés partagent l'intérêt de modéliser des agents qui prennent des décisions locales sous information incomplète, lesquelles doivent néanmoins être globalement cohérentes. Des efforts récents ont rapproché ces perspectives, en adaptant des procédures algorithmiques au contrôle de systèmes dynamiques, et en ajoutant une dimension spatiale à des modèles computationnels. Cette thèse développe les fondements mathématiques de cette intersection.

L'observation centrale est qu'une structure pertinente à la coordination vient de l'information disponible aux robots lorsqu'ils doivent décider. La première contribution formalise cette intuition en établissant que les systèmes dynamiques hybrides et des modèles computationnels discrets peuvent encoder la même information épistémique. En séparant les descriptions physiques et computationnelles d'un robot en deux sous-systèmes connectés, une relation de simulation préservée au cours des exécutions justifie l'adoption d'abstractions discrètes pour raisonner sur des robots à dynamique continue.

Cette équivalence fournit un modèle d'exécution commun à partir duquel trois cadres complémentaires sont développés. Le premier formalise les besoins épistémiques pour la coordination robotique à travers une logique temporelle-épistémique avec propositions spatiales dérivées de l'espace de mission. La connaissance au cours d'une exécution est modélisée géométriquement, où les états globaux sont reliés par des relations de causalité ou d'indiscernabilité. La connaissance des faits statiques est préservée par les systèmes robotiques avec mémoire parfaite, et des conditions suffisantes pour l'exploration, la surveillance et le rassemblement sont dérivées, séparant propriétés comportementales de celles liées à la communication.

Le second cadre reformule la résolvabilité des tâches comme un problème d'extension d'une information locale à la structure globale, à travers la théorie des faisceaux. Un faisceau de tâche assigne des décisions localement valides aux

cellules d'une catégorie dérivée de la structure d'exécution, avec des restrictions imposant la cohérence par robot. Ces faisceaux peuvent être définis individuellement par agent et se composent de manière unique, permettant une analyse modulaire. La résolvabilité correspond à l'existence de sections globales, et la cohomologie de degré zéro de sa linéarisation fournit une procédure semi-calculable pour construire des solutions lorsqu'elles existent.

Le troisième cadre récupère le processus génératif des exécutions qui a été abstrait par les cadres précédents. En restreignant l'attention aux communications sans-attente et spécifiées par rounds, la spécification de ce qui peut arriver à un seul état global pour un round devient suffisante pour dériver toutes les configurations atteignables. Les protocoles de communication sont modélisés comme des endofoncteurs sur les ensembles semi-simpliciaux chromatiques, dont les algèbres libres accumulent ces configurations par une itération non bornée. Une logique temporelle-épistémique sur ces algèbres exhibe la persistance de la connaissance comme propriété structurelle plutôt que comme axiome externe. Le cadre est ensuite étendu aux foncteurs de protocole avec valeurs de décision, où des protocoles concrets modélisent explicitement le processus de décision itératif.

Ensemble, ces approches relient les perspectives épistémique, topologique et algébrique sur la coordination dans un langage commun. Cette collection d'outils auparavant isolés permettent de raisonner préalablement à l'implémentation sur la faisabilité de missions robotiques distribuées.

ACKNOWLEDGEMENTS

The writing of these words marks the conclusion of a beautiful experience that spanned the last three and a half years. This moment brings many emotions, foremost among them gratitude, which I try to put into words here.

This work owes much to my thesis supervisors, Eric, Sylvie, and Luc. You pushed me further and wider than I thought possible with your candor and intellectual generosity. I am grateful for your trust, for constantly inciting my curiosity, and for teaching me to find the ground whenever I floated too far.

I am deeply thankful to Georg, Thomas, and Uli, who thoughtfully reviewed this thesis and provided careful commentary. I also thank Hans, Nobuko, and Ulrich for kindly agreeing to be part of my jury.

Those years have been a source of great joy, learning, and openness. I feel lucky to have crossed paths with each of you, and delighted to see so many friendships that outlasted the circumstances in which they began.

Central to it were my colleagues at LIX. I first thank Maria, for lending me a hand when I arrived in Brest, and for remaining just as kind when we met again in Palaiseau. To Roman, whose frankness was surprising at first and quickly made discussing life and science more amusing. To Aly-Bora, for sharing an appreciation for good (or free) food and for absurdly long board games. To Riccardo, for helping me see life with lighter eyes, through an inner peacefulness that one day I hope to learn. To Elena, for so quickly supporting every shot at some shared fun. To Sergio, for the many coffees and small grumbles we vented over them. And to Martin, for always being there to help, and meaning it.

A special mention is due to my co-thésards, who vastly enriched the life inside and outside the lab. I thank Nathan, for the fun and camaraderie you brought to our office, offering your honest engagement with matters big and small. To Manon, for withstanding many of my long questions, and for your eagerness to share experiences and pieces of what brought joy, nearly matched by the fullness of your calendar. To Aloysio, for the longest discussions about the smallest of things, e de forma geral pela tua gentileza e disponibilidade quando importava. To Mathilde, for the good company when our travels did and did not work. To Elies, for your laudable communication ethic, making every discussion a fun compilation problem. To Namratha, for being so relentlessly upbeat and sincere at every interaction that no mood could resist joining. To Yorgo, for always being

available to share a beer or a mathematical inquiry, before even knowing which one it was. And, finally, to Anaëlle, thank you for your liberal sharing of time and kindness, especially in these final moments, and for just being a great friend.

I wish also to thank my other colleagues, Nan, Mehdi, Mathilde, Nuwan, Louise, Gaspard, Augustin, Jessica, Constantin, Emmanuel, Sam, and Jérémy, for sharing your time, in our corridor and beyond. I thank Émile, Théo, and Sarah for the more or less successful, but always fun, seminar undertakings. I am thankful to have finally met Yulia, Rafael, Omar, and Zach, after years of unknowingly sharing the same building.

More than once I found myself in new and foreign places, but never for long truly displaced. Across summer schools and conferences, for that I owe thanks to Max, Cédric, Lena, Mike, Timon, Tiana, Vincent, Arun, Giorgio, Hugo, Stephan, David, and Mirko.

An especially meaningful moment, both academically and personally, was the autumn of 2024 I spent in Vienna. The relationships I built there meant the most. I am firstly thankful to Ulrich, for helping to make this exchange happen. I am grateful to Giorgio, Hugo, and Stephan, for the astonishingly little time between my arrival and feeling at home, and for the deeply stimulating and candid conversations that spanned those months. I extend this thanks to Hans, Roman, Florian, Johannes, Sebastian, Ines, Ruslan, and Colombelli, who coincidentally and incidentally became central to my time there.

I am grateful to all of those who have been present in the surroundings of the doctoral experience. I thank Pascal, Jonathan, and Lucia, for the warmest welcome as I arrived in Palaiseau, and for renewing that feeling at every moment we shared. I am grateful for the lasting friendship of Artur and Pedro. Pontual e comedida, obrigado pela companhia de qualidade que sempre ignorou tempo e distância. Ringrazio Simona e Luca per avermi accolto con calore nella loro casa e nelle Dolomiti, mentre iniziavo a scrivere questa tesi. Non avrei potuto desiderare un inizio migliore. Agradeço aos meus pais, Dione e Luis, pelo apoio e atenção que acompanharam e acompanham todo novo desafio que me ponho a enfrentar.

At last, I wish to thank Francesca, whose role in lifting the ups and softening the downs of the life that ran alongside these years cannot be overstated. Thank you for always being there, bringing joy to the ordinary, and more to what would have felt enough.

# CONTENTS


ABSTRACT iii

ACKNOWLEDGEMENTS v

1 INTRODUCTION 1

2 PRELIMINARIES 6
- 2.1 Mobile robotics . . . 6
- 2.2 Distributed computing . . . 14
- 2.3 Mathematical frameworks . . . 23

3 MODELS FOR DISTRIBUTED ROBOTICS 37
- 3.1 Time . . . 39
- 3.2 Robots, dynamical systems . . . 42
- 3.3 Robots, transition systems . . . 48
- 3.4 Correspondence between the two views . . . 56
- 3.5 Conclusion . . . 62

4 KNOWLEDGE IN MULTI-ROBOT SYSTEMS 65
- 4.1 System frames and knowledge . . . 68
- 4.2 A temporal epistemic logic of space . . . 72
- 4.3 Modeling robot tasks . . . 79
- 4.4 Conclusion . . . 98

5 CELLULAR SHEAVES OF DISTRIBUTED TASKS 100
- 5.1 System slices and local consistency . . . 103
- 5.2 Task sheaves . . . 114
- 5.3 Task solvability and sheaf sections . . . 125
- 5.4 Computing solutions . . . 131
- 5.5 Conclusion . . . 141

6 ITERATED ALGEBRAS OF DISTRIBUTED PROTOCOLS 145
- 6.1 From transition systems to protocol functors . . . 148

Chapter Page

6.2 Algebras of protocols . . . . . 161
6.3 Temporal-epistemic semantics on protocol algebras . . . . . 168
6.4 Incorporating local decision functions . . . . . 177
6.5 Conclusion . . . . . 186

7 CONCLUSION 189

REFERENCES 193
INDEX 205

# 1

# INTRODUCTION

The coordination of multi-robot systems requires solving two intertwined problems: controlling physical systems under dynamical constraints, and guaranteeing the collective behavior of distributed agents with limited information. The first set of problems is studied within mobile robotics, while the second is the subject of distributed computing. Their intersection encompasses what can be understood as distributed robotics.

The two fields have evolved towards multi-agent systems from distinct perspectives. Mobile robotics considers first the physics of the system, where a robot is subject to forces and the challenge is to steer it despite the gap between intended and actual behavior. It deals with noisy sensors, imprecise actuators, and dynamics that impose hard constraints on which executions are feasible. A natural evolution was to assemble several robots into larger coupled dynamical systems, where control theory was already well-developed. The transition to multiple robots was marked by the seminal work of Olfati-Saber, Fax, and Murray [OFM07] that framed consensus as a stability problem over interaction graphs while retaining a global viewpoint of the system.

Distributed computing begins instead with a focus on information. Computation is abstracted into processes that communicate by exchanging messages or writing in a shared memory, where the challenge is to guarantee collective

behavior despite each process executing independently with only its local view. The central objects of study are structures derived from the possible executions, encoding varying communication protocols, activation schedules, among other sources of uncertainty. Lamport's abstraction of concurrent events into partial orders [Lam78] made this approach explicit, while the impossibility of consensus with one faulty process established by Fischer, Lynch and Paterson [FLP85] revealed that the obstacles lie not in computational power, but in the epistemic limitations of what can be inferred from local observations. Mobility came later through the LOOK-COMPUTE-MOVE model of Suzuki and Yamashita [SY99], arriving as a discrete abstraction where robots jump between positions, observe instantaneously, and compute in negligible time. Such simplicity enables a more precise carving of the solvability landscape, but creates the need for a translation back to continuous dynamics, where physical robots operate.

Despite different assumptions about dynamics and communication, both communities share the interest in modeling agents that make local decisions under incomplete information, which must nevertheless be globally coherent. This observation was made explicit by Alcántara et al. [Alc+19], who showed that robot coordination under crash failures reduces to distributed agreement. This reduction opened the way for the topological characterizations of task solvability from Herlihy and Shavit [HS99] to be applied in the mobile setting. In the opposite direction, algorithms for fault-tolerance have also been shown relevant for networked dynamical systems by Dibaji and Ishii [DI17], who applied message filtering to agents with second-order dynamics. Such transfers succeeded because the relevant structure comes not from the physics, but from the information available to robots when they must decide. This thesis develops that insight systematically.

The first contribution makes precise under which conditions hybrid dynamical systems and discrete computational models produce equivalent epistemic states, showing that reasoning within a computational abstraction preserves the information available to robots with continuous dynamics. Three complementary frameworks emerge from this common foundation, each addressing a different aspect of distributed coordination. A temporal-epistemic logic with spatial propositions formalizes task requirements in terms of the information available to robots, yielding sufficient conditions for exploration, surveillance and gathering

that separate behavioral properties from communication requirements. Cellular sheaves reformulate task solvability as a question of whether local choices extend to global decisions, for which cohomology provides a semidecidable procedure to answer. Protocol functors capture how communication builds the distributed state space as structure-preserving transformations, with free algebras accumulating all reachable configurations through unbounded iteration.

**Organization and contributions** Chapters 2 and 3 establish the modeling foundations.

Chapter 2 reviews the frameworks used in mobile robotics and distributed computing to represent multi-robot systems, and recalls the mathematical tools used throughout.

Chapter 3 addresses how dynamic and computational models of distributed robotics relate. The main contribution lies in making precise when reasoning with a computational abstraction transfers meaningfully to the dynamical model. The convenient separation of a robot state into ontic and epistemic components, isolating the computational information from the physical dynamics, makes evident limitations arising from the information available. Time paths are introduced as a geometric model defining the relationship between local and global time in a multi-robot system. A hybrid dynamical system captures the physical evolution of robots, later abstracted into a compositional I/O automaton. The two are connected by a system abstraction that preserves their epistemic content (Corollary 3.36), justifying the adoption of the discrete model for the subsequent chapters. This is an extension of the first part of [Cig+25].

The discrete compositional model defined in Chapter 3 provides a common execution structure from which three complementary frameworks are developed in Chapters 4 to 6, each addressing a different aspect of distributed coordination.

In Chapter 4, we formalize epistemic requirements for different levels of robot coordination. System frames extend the runs and systems framework of Fagin et al. [Fag+95] with explicit causality and indistinguishability relations parametrized by groups of robots, providing a geometric representation of how knowledge evolves throughout executions. A temporal-epistemic logic with spatial propositions connects regions of the mission space to formulas modeled on these frames, and we show that the knowledge of static facts is preserved

by robotic systems with perfect recall (Theorem 4.13). This language is used to derive sufficient epistemic conditions for three tasks, separating behavioral from communication requirements. Exploration requires both physical coverage and communication liveness (Theorem 4.21). Surveillance adds a temporal dimension to the mission space and requires analogous coverage properties (Theorem 4.30). Gathering is reduced to a variation of stabilizing agreement with regions of space as decision values (Theorem 4.36), bridging this classical distributed problem to the robotic setting. This chapter is an extension of the second part of [Cig+25].

In Chapter 5, we move from specifying what robots must know to characterizing when they can act on that knowledge. The tension between local decisions and global validity is central to distributed computing, and sheaf theory provides a natural language to express it. System slices extract finite substructures from system frames via execution cuts, and are shown to be as expressive as the original frame for assessing whether a task can be solved (Theorem 5.10). A cell category derived from the slice encodes the causal consistency constraints that any valid terminating decision function must respect. Task sheaves assign to each global state the space of valid decisions, with restriction maps enforcing agent-wise consistency. These sheaves can be defined individually per agent and compose uniquely (Theorem 5.29), enabling a modular analysis. The solvability of a task corresponds then to the existence of sections in a task sheaf from a given system and task (Theorem 5.33). Additionally, we show the first advantage of this framework by observing that the zeroth cohomology of the linearized task sheaf provides a semidecidable procedure for constructing solutions when they exist (Theorem 5.44). This chapter is based on [FHR25b] and its extended version [FHR25a].

In Chapter 6, we recover the generative process of the executions that both previous chapters abstracted away. System frames, while derived from an operational semantics described in Chapter 3, assume a given set of executions to work from. Protocol functors model communication as structure-preserving endofunctors on chromatic semi-simplicial sets, while their free algebras accumulate all reachable configurations through unbounded iteration from any initial state (Corollary 6.21). A temporal-epistemic logic is then interpreted on these algebras, where knowledge persistence emerges as a structural property (Theorem 6.29) rather than an external axiom, contrasting with the axiomatic approach of Chap-

ter 4. The framework is then lifted to protocol functors with decision values, where concrete protocols model the iterative decision process explicitly. These are also shown to be models of an extended temporal-epistemic logic, recovering specifications analogous to the spatial task properties of Chapter 4. This chapter is adapted from [Gou+25].

# 2

# PRELIMINARIES

Distributed multi-robot systems live at the intersection between mobile robotics and distributed computing. Robots have dynamical constraints imposed by their physical construction, such as limited maneuverability or sensor range, while they must solve complicated coordination problems as a distributed system. This chapter reviews preliminary concepts from three perspectives: mobile robotic systems as physical entities (Section 2.1), and as computational entities in distributed computing (Section 2.2), followed by the mathematical frameworks used for reasoning about distributed systems (Section 2.3). The progression from concrete dynamics to abstract models of computation mimics the structure of our main results, where the overlap between perspectives provides the main insight of this thesis: these different views are representations of the same content and evolution of information.

## 2.1 MOBILE ROBOTICS

Mobile robotics centers on three fundamental challenges: mapping unknown environments, localizing within those environments, and planning safe paths to goals. We define now the minimal vocabulary needed to speak of robots as

physical systems. Robots are first described as individual entities, and then as part of a multi-robot system.

### 2.1.1 MODELING ONE ROBOT

We model a robot as a (possibly hybrid) dynamical system capable of performing observations through its sensors and of being steered through a controller. The main reference for this section is the textbook by Khalil [Kha02].

▶ **Definition 2.1 (Dynamical system).** A *dynamical system* is a tuple $(X, f)$ where $X \subseteq \mathbb{R}^d$ is the *state space*, a $d$-dimensional Euclidean space, and $f\colon X \times \mathbb{R}^+ \to X$ is a *dynamics* function describing the system evolution $\dot{x} = f(x,t)$ for a state vector $x \in X$, which satisfies the *locally Lipschitz* property, i.e., for every $x_0 \in X$, there exists a neighborhood $U$ of $x_0$ and a constant $L$ such that $\|(f(x,t) - f(y,t))\| \leqslant L\|x - y\|$ for all $x, y \in U$. ◀

The state of a dynamical system encodes the physical quantities of the robot, such as position, velocity and orientation. It may include arbitrarily more variables that one may want to keep track of, such as the individual speed of each of its motors, or its logical counters. The dynamical system's evolution is described by the initial value problem $\dot{x} = f(x,t)$ with $x(0) = x_0$, where $f$ is assumed to be locally Lipschitz in $x$ and continuous in $t$ to ensure the existence and uniqueness of solutions by the Cauchy–Lipschitz theorem.

A robot interacts with an environment through actuators, to influence its own state, and sensors, to obtain information. The influence mechanism is formalized by extending the dynamical system's evolution to depend on external inputs, such that they "steer" the state of the system. This forms a controlled dynamical system.

▶ **Definition 2.2 (Controlled dynamical system).** A *controlled dynamical system* is a tuple $(X, U, f)$, where $U \subseteq \mathbb{R}^m$ is a set of possible *control inputs* and the dynamics $f\colon X \times U \times \mathbb{R}^+ \to X$ incorporates $U$. Its evolution is described by $\dot{x} = f(x,u,t)$. ◀

The set of control inputs describes the possible ways in which a dynamical system can be influenced. There are multiple ways to generate such inputs. The

simplest is to define a function dependent of the current state of the system acting as a controller.

▶ **Definition 2.3 (State-feedback controller).** Let $(X, f)$ be a dynamical system modeling the dynamics of a robot with evolution $f$ and $U \subseteq \mathbb{R}^m$ a set of available control inputs. A *state-feedback controller* is a map $h\colon X \times \mathbb{R}^+ \to U$. ◀

A state-feedback controller selects a control input $u \in U$ according to the state $x \in X$ and time $t \in \mathbb{R}^+$. When time-invariant, the dependence on $t$ is omitted.

The controlled dynamical system can equivalently be written component-wise as follows.

$$\dot{x} = f(x,u,t) = \begin{cases} \dot{x}_1 &= f_1(x_1,\dots,x_d,u_1,\dots,u_m,t) \\ \dot{x}_2 &= f_2(x_1,\dots,x_d,u_1,\dots,u_m,t) \\ \vdots & \vdots \\ \dot{x}_d &= f_d(x_1,\dots,x_d,u_1,\dots,u_m,t) \end{cases} \tag{2.1}$$

▶ **Example 2.4 (Dubins car).** The *Dubins car* is a dynamic model commonly used for modeling a robot that moves forward through a controllable linear velocity along its heading direction, and turns with bounded angular velocity. Such a robot is nonholonomic, meaning that its velocity is constrained by its configuration so that it can only move forward, without any lateral velocity. Cars and torpedo-like submarines are common examples that can be simplified as Dubins cars. As a controlled dynamical system $(X, U, f)$, it is characterized by a state $(x, y, \theta) \in \mathbb{R}^2 \times S^1$, where $(x, y)$ is a position in the plane and $\theta$ its heading angle, and by a control input $u = (v, \omega) \in U$ consisting of a linear velocity $v$ and an angular velocity $\omega$, constrained by $|\omega| \leq \omega_{max}$. The dynamical system is defined as follows.

$$\dot{x} = f(x,u) = \begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{2.2}$$

A state-feedback controller can be used for generating the control inputs as $u(t) = h(x(t), y(t), \theta(t))$ ◀

In the context of robot modeling, the usage of such a state-feedback controller assumes full information about its state. In practice, the information available is

derived from multiple procedures and rarely perfectly accurate. When no external sensors are available, a robot may perform dead reckoning, where it simulates its own dynamics with the knowledge of control inputs, and uses its estimated state in place of its real one. Sensors improve estimations through the observations they provide. This interface is formalized by an observer.

▶ **Definition 2.5 (State observer).** A *state observer* for a dynamical system $(X, f)$ equipped with an observation function $g\colon X \to Y$, where $Y$ is a set of possible observations, is a dynamical system $(\widehat{X}, h)$, where the function $h\colon \widehat{X} \times Y \times \mathbb{R}^+ \to \widehat{X}$ produces an estimate $\widehat{x}(t)$ of $x(t)$, designed so that $||\widehat{x}(t) - x(t)|| \to 0$ as $t \to \infty$. ◀

Through the observer and the controller one models both the information gathering capacities of a robot and its intended behavior. These two components, alongside the robot's dynamical system model, form the most basic representation of a robot in mobile robotics. An important extension is to consider discrete transitions in its dynamics, accommodating that the trajectory of the system may perform occasional "jumps". Examples are the (intentional) discrete logical steps of an algorithm and (unintentional) events such as the introduction of a dangerous element that must be handled differently. This type of system is formalized as a hybrid dynamical system.

▶ **Definition 2.6 (Hybrid dynamical system).** A *hybrid dynamical system* is a tuple $(X, E, U, V, f, \varphi)$, where $(X, f)$ define its continuous dynamics with possible control inputs $U$, and $(E, \varphi)$ define its discrete *jump* dynamics with possible jump inputs $v \in V$. The complete system dynamics is defined as

$$\begin{aligned} \dot{x}(t) &= f(x, u, t) \quad t \in \mathbb{R}^+ \setminus \mathcal{J} \\ x^+(t) &= \varphi(x, v, t) \quad t \in \mathcal{J} \end{aligned}$$

where $\mathcal{J} = \{t_i \in \mathbb{R}^+ | t_i < t_{i+1}\}$ is an increasing sequence of jump times. ◀

We observe now that the challenges found in problems of mapping, localization and path planning motivate questions in the design of controlled dynamical systems. The relationship between the different problems is depicted in Figure 2.1.

In the localization problem, a robot must estimate its state $x \in X$ from potentially noisy observations $y = g(x) + \varepsilon$, such as GPS data or distance measurements

to landmarks, where $\varepsilon$ is some sensor noise. This requires the design of a state observer that accounts for the noise to produce an estimation of the state that converges to the true value over time. Path planning can be depicted as the need to generate control inputs that steer the robot from its initial state towards some goal state, potentially having to avoid obstacles. Obstacle avoidance can be formalized as a set of unsafe states, $X_{unsafe} \subset X$, which should not be reached through the inputs of the controller. The mapping problem is usually tackled through some variation of the exploration task, where robots must construct a map representing an unknown environment by iteratively updating it based on observations, while simultaneously using this partial map for localization and path planning. A vast literature exists on tackling these problems through control theory [Kha02], reachability analysis [BM15], interval analysis [Jau+01], probabilistic methods [TBF05], among others. There is a considerable overlap across these approaches, as models or theoretical tools are often shared. Chapter 3 will depict a modeling approach that allows us to include distributed computing techniques to this list, and Chapter 4 will show the first of such usages, through a logical characterization of the knowledge required to solve such tasks.

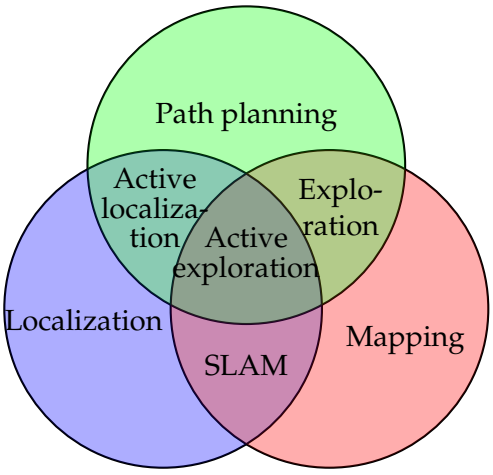



Figure 2.1
Relationships between mapping, localization, and path planning problems. SLAM stands for simultaneous localization and mapping.

Control theory addresses these challenges through tools such as the stability analysis via Lyapunov theory, systematically building a stable control with PID

controllers, and handling some cases of non-linear systems with linearization. Multi-robot coordination introduces a new communication layer to these problems, which we review next.

### 2.1.2 MODELING MULTIPLE ROBOTS

While individual robots can solve many problems in path planning, localization, and mapping, multi-robot systems offer distinct advantages: parallel execution reduces mission time, redundancy improves robustness to failures, and coordinated control enables complex behaviors impossible for a single robot, such as the simultaneous surveillance of an area and distributed sensing. In this section, we focus on the changes found when considering the control of multiple robots instead of one. The main reference for this section is the recent textbook of Postoyan et al. [Pos+24] on networked dynamical systems.

The simplest extension possible to depict $n$ robots as controlled dynamical systems is to combine their dynamics them in a single stacked representation. Their individual states $x_i \in \mathbb{R}^d$ and dynamics $\dot{x}_i(t) = f(x_i, u_i, t)$ are merged in a global state $\mathbf{x} \in \mathbb{R}^{d \times n}$ with dynamics

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, u_1, t) \\ \vdots \\ f_n(x_n, u_n, t) \end{bmatrix} \tag{2.3}$$

The decoupled system above, however, ignores the interactions that make coordination possible. Robots exchange information through perception and communication[1]. Interaction naturally forms a graph structure, $G = (V, E)$, where vertices $V = \{1, \ldots, n\}$ correspond to robots and edges $E \subseteq V \times V$ represent interaction links. The graph of a robot network can be compactly represented through its adjacency matrix $A \in \mathbb{R}^{n \times n}$, with entries $a_{i,j} > 0$ if $(i, j) \in E$ and zero otherwise. This algebraic representation of connectivity enables the use of spectral properties of graphs as a new source of tools to design and analyze multi-robot systems. The most important such matrix is the graph Laplacian, which encodes how local interactions propagate through a network.

[1] Perception and communication will be formalized as distinct information channels in Chapter 3.

▶ **Definition 2.7 (Graph Laplacian).** Let $G = (V, E)$ be an undirected graph with adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $a_{ij} \geq 0$ for all $i, j \in V$. The *graph Laplacian* of $G$ is the matrix $L = D - A$, where $D = \text{diag}(\sum_j a_{1,j}, \ldots, \sum_j a_{n,j})$ is the diagonal matrix. Equivalently, its entries are

$$L_{ij} = \begin{cases} \sum_{k \neq i} a_{ik} & \text{if } i = j \\ -a_{ij} & \text{if } i \neq j \end{cases} \tag{2.4}$$

◀

The entries of the adjacency matrix are non-negative real weights denoting some form of coupling strength between robots, such as the flow of information exchange, while Boolean entries are used when only connectivity information is relevant. Note that, while pairwise interactions are sufficient for numerous applications, higher-order interactions (across groups of robots, simultaneously) can be modeled through hypergraphs and simplicial complexes [Bat+20]. This modeling choice has been little explored in robotic applications, and here we focus on the standard pairwise case.

The gathering robot task, where robots are required to meet in a common position in space, can be solved in some cases using the Laplacian dynamics

$$\dot{x}_i(t) = -L_i = -\sum_{j \in \mathcal{N}(i)} a_{i,j}(x_i(t) - x_j(t)), \quad i \in V \tag{2.5}$$

where $\mathcal{N}(i) = \{j \mid (i, j) \in E\}$ is the set of neighbors of robot $i$. The complete dynamics of the network becomes $\dot{\mathbf{x}} = -L(\mathbf{x})$, and corresponds to each robot updating its own position towards the average of its neighbors. Under assumptions such as of graph connectivity, with a path between any pair of vertices, the system is guaranteed to converge.

▶ **Proposition 2.8 (Convergence of simple integrators following Laplacian [OM04]).** Let $G = (V, E)$ be a static undirected and connected network graph with adjacency matrix $A$. The system $\dot{\mathbf{x}} = -L\mathbf{x}$ converges asymptotically towards consensus, i.e., $\lim_{t \to \infty} \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| = 0$ for all $i, j$. ◀

While Proposition 2.8 only considers simple systems with single integrator dynamics and static networks, a large body of work exists on generalizations to switching and time-varying networks [RB05], double integrator and nonholonomic dynamics [LFM05], malicious [DI17] and Byzantine faults [Sau+17],

delays in the communication [Nuñ+22], beyond all the different combinations of such constraints, of which Postoyan et al. [Pos+24] provides a recent survey. We will later see in Section 4.3 that the gathering robot task is closely related to the consensus distributed computing task, which further increases this taxonomy to include other variations in network and communication properties.

These extensions share the common objective of characterizing stability under time-varying interactions. While the hybrid dynamical systems formalism can model any type of discrete events, the simultaneous treatment of multiple problems becomes combinatorially more complex to analyze, as the jump dynamics must track all possible transitions. In this light, switched systems provide a cleaner abstraction to model the extra layer of complexity in cases such as asynchronous activations, as seen in Lee [Lee21].

▶ **Definition 2.9 (Switched system).** A *switched dynamical system* is a tuple $(\mathcal{P}, X, \{f_\rho\}_{\rho\in\mathcal{P}}, \xi)$, where $\mathcal{P}$ is a finite set of *modes*, $\{f_\rho\}_{\rho\in\mathcal{P}}$ is a family of dynamics indexed by the modes $\rho \in \mathcal{P}$, such that each pair $(x, f_\rho)$ is a dynamical system, and $\xi : \mathbb{R}^+ \to \mathcal{P}$ is a *switching signal*. The system evolution is described by

$$\dot{x}(t) = f_{\xi(t)}(x(t), t) \tag{2.6}$$

◀

▶ **Remark 2.10 (Switched systems are hybrid systems).** Observe that a continuous switched systems are special cases of hybrid systems, where one neglects the discrete jump dynamics and only uses the switching of modes, as it is argued in Liberzon [Lib03]. We use switched systems as a more convenient formalism for multi-robot systems. ◀

A common choice in multi-robot systems is to let each mode $\rho \in \mathcal{P}$ encode which robots are active at a given time. The switching signal captures then the activation schedule[2]. When considering the gathering problem, a switched linear system

$$\dot{x(t)} = -L_{\xi(t)}x(t) \tag{2.7}$$

models the averaging procedure, now dependent on the time-varying interaction graph represented at each mode $\rho = \xi(t)$. The stability of such systems can be

[2]Section 3.1 will define a geometric model of such schedules using time paths,

analyzed through the joint spectral radius of the family of dynamics, as reviewed in Liberzon [Lib03]. The computational complexity aspects of such generalization, as well as its approximations, are discussed in Theys's thesis [The05].

The hybrid and switched systems formalisms will serve as the grounding model for robots and multi-robot systems in this thesis. In order to bring forward the discrete computational processes behind coordinated behavior, Chapter 3 will partition the state space into ontic (physical) and epistemic (computational) components enabling the individual treatment of the physical evolution and the handling of information. As such, we will preserve the backing theory of hybrid systems with our continuous and discrete separation, as well as that of switching modes to model different activation patterns.

## 2.2 DISTRIBUTED COMPUTING

### 2.2.1 MODELING PROCESSES

Distributed computing consists of using multiple independent computing units, or *processes*, to solve a common task, exploiting as much as possible their concurrent execution to speed up computations. The distributed nature of the computations comes from their physical separation, and introduces the need for *communication* between processes. This adds a complexity layer to handle the information flow in the distributed version of an algorithm, which is now split into a decision function and a communication protocol. The fundamental challenge in distributed computing is to coordinate local (process-wise) decisions to achieve the desired global outcome. The main reference used in this section for modeling such systems is Lynch [Lyn97].

A distributed system consists of a set of $n$ processes $\Pi = \{p_1, \ldots, p_n\}$ executing an *algorithm* $\mathcal{A}$, which describes the communication protocol $\Psi$ and decision function $\delta$. Each process $p_i$ contains a *local state* $s_i$ with its private information, and the set of all local states forms a *global state* $\mathbf{c} = \{s_1, \ldots, s_n\}$. A process can only obtain information from another process' local state through communication, which can be modeled in several forms. We present now the two most relevant ones for our work.

▶ **Definition 2.11 (Shared memory model).** A *shared memory system* consists of $n$ processes communicating through a *shared object*, the simplest of which being an *atomic register*, supporting atomic operations of READ and WRITE. A process executes an algorithm composed of READ, WRITE and COMPUTE steps, where COMPUTE abstracts a local computation. ◀

The shared object determines how to handle simultaneous accesses, i.e., at the same time step[3] in the perspective of the object. A common choice for data consistency is that READ operations can be done in parallel, but only one process may WRITE at a time.

An important example is the *immediate snapshot* (IS), which guarantees that processes execute atomically the READ and WRITE sequence of operations. That is, if process $p_i$ executes a READ while another process $p_j$ has only executed a WRITE (and not the following READ yet), then process $p_j$ must execute its READ operation before any other process $p_{k\neq j}$ is allowed to WRITE again. This constraint ensures that the value of $p_j$ seen by $p_i$ is either the current one (they performed READ operations simultaneously) or one value too old ($p_j$ executed READ before $p_i$ did a WRITE), that is, that the WRITE and READ retain a causal relation. This behavior can be simulated in a fully synchronous setting [BG93], simplifying protocol analysis. The shared memory system may also define different guarantees for its atomic operations, changing its computation power.

▶ **Definition 2.12 (Message-passing model).** A *message-passing system* consists of $n$ processes $\Pi = \{p_1, \ldots, p_n\}$ in a network connected by directed *communication channels*. For each pairwise channel $(p_i, p_j)$, process $p_i$ has available operations $\text{send}_{i,j}(m)$, which places *message* $m$ in the channel, and $\text{receive}_{j,i}()$, which either retrieves a message or $\perp$ from the channel. A process executes an algorithm composed of SEND, RECEIVE and COMPUTE operations. ◀

The communication channel determines the message-passing system's limitations. It may be reliable or unreliable depending on whether messages can be lost, and implemented using different distributed structures, such as a FIFO queue

[3] Time itself is a complicated matter in distributed computing, which we will address in Section 3.1.

that preserves order or with randomized delivery. The channels can be formalized through explicit communication graphs, which specify which processes can communicate at each round.

▶ **Definition 2.13 (Communication graph).** Let $\Pi$ be a finite set of processes. A *communication graph* of $\Pi$ is a directed graph $G = (V, E)$, where $V = \Pi$ and each edge $(p_i, p_j) \in E \subseteq V \times V$ is a communication channel from $p_i$ to $p_j$. Edges are represented as $p_i \to p_j$ when the graph is clear from context. A process $p_i$ is *active* in $G$ if there exists the edge $p_i \to p_i$, denoted $p_i \in \text{active}(G)$ for the set of all active processes. The *view* of an active process $p_i$ is $\text{view}_G(p_i) = \{p_j \in V \mid p_j \to p_i\}$. ◀

An important generalization of the static communication graph is the *dynamic network model* [KO11], when the set of agents that can communicate with each other may change at each round. The specific communication dynamics is given by a *message adversary* $M$, that chooses it at each round through a process that may depend on the system specification or be *oblivious* [CGP15], when the possible graphs are the same every round.

Both shared memory and message-passing models face the fundamental challenge of synchronicity, i.e., determining when each process activates, executing one of its local steps, and how the activation order influences the validity of the information being processed. The possible activation orders are abstracted by a *scheduler*, which captures the relative logical time of processes' executions, that is, an ordering of events based on causal precedence instead of physical clocks, as introduced by Lamport [Lam78]. They range from *synchronous* (FSYNC), when every process executes simultaneous step of identical duration, through *$k$-synchronous* ($k$-SYNC), that bounds the difference between any two processes by a parameter $k$, up to *asynchronous* (ASYNC), when no shared notion of time exists and steps may be executed in any order and duration. Section 3.1 will formalize these schedulers geometrically as time paths.

Another important challenge within distributed systems is resilience to failures. These can be either *crash faults*, when a process may halt (temporarily or permanently), or *Byzantine faults*, when a process may behave arbitrarily (with malicious or simply erroneous ways). A *$t$-resilient* system tolerates up to $t$ process failures of either crash or Byzantine faults, in which case it is ensured to present a globally correct behavior despite some local faults. A significant class of systems

are those that guarantee that every correct process completes its operations in a bounded number of steps, regardless of the failures of others, in which case the system is called *wait-free* (often abbreviated as WFSM for the shared memory model). A wait-free system is necessarily $(n-1)$-resilient, as it must tolerate all but one process failing.

▶ **Remark 2.14 (Correspondence of shared memory and message-passing models).** Note that the shared memory and message-passing models possess equivalent computational power when both are (up to) $t$-resilient and asynchronous, for $t < n/2$ with reliable channels, shown in Attiya, Bar-Noy and Dolev [ABD95]. This allows us to continue working with the more convenient shared memory systems without loss of generality, whose I/O automata implementation [Lyn97] we use in Chapter 3 to model distributed multi-robot systems. ◀

The distributed computing literature encompasses a vast taxonomy of tasks, distributed versions of computational problem, which are derived from the variations of synchronicity and resilience above. A common question to pose is whether a distributed task is solvable under some communication protocol. This question accepts a convenient re-framing, first seen in Elrad and Francez [EF82]: the system first executes several rounds of communication and then collectively decides once enough information has been exchanged. This leads to the standard task-theoretic formulation of distributed problems.

▶ **Definition 2.15 (Distributed task).** Let $\Pi = \{p_1, \ldots, p_n\}$ be a set of $n$ processes. A *distributed task* is a triple $\mathbf{T} = (I, O, \Delta)$, where:

- $I$ is a set of possible initial global states, each specifying *initial values* $v_i$ per process $p_i$ in which the task may start;
- $O$ is a set of task-compliant terminal global states, each specifying *output values* $d_i$ per process $p_i$;
- $\Delta \subseteq I \times O$ is a *task specification* relating valid pairs of input and output global states.

◀

The question of task solvability becomes asserting whether a set of processes with specified initial local states, after enough rounds of communication, are able to deterministically and simultaneously decide on some valid output that matches the requirements of the distributed task.

▶ **Definition 2.16 (Distributed task solvability).** Let $\Pi = \{p_1, \ldots, p_n\}$ be a set of processes, and $\mathbf{T} = (I, O, \Delta)$ be a distributed task. A distributed system composed of $\Pi$ under *communication protocol* $\Psi$ *solves* the task $\mathbf{T}$ if there exists a deterministic *decision function* $\delta \colon \Psi^n(I) \to O$, defined on local states $\delta \colon C_p^n \to O_p$ identically for every $p \in \Pi$, that maps every reachable global state $C^n \in \Psi^n(I)$ to a valid output in $O$, i.e., $(C^0, \delta(\Psi(C^n))) \in \Delta$, for some $n \geq 0$. ◀

The decision function requirement can be translated to achieving coordinated distributed behavior, where the local decisions $d_i \in im(\delta)$ are consistent according to the task definition. In complex tasks, the first step to achieve such coordination is often to establish some simpler common ground in which all processes can agree. An additional common requirement for decision functions is that they are *terminating*, that is, that once all local decisions are consistent with the task, the processes should terminate their executions[4], in which case decisions are said to be *final*. The essence of these requirements is captured by the binary consensus problem.

▶ **Example 2.17 (Terminating binary consensus).** Consider $n$ processes $\Pi = \{p_1, \ldots, p_n\}$, among which $t < n$ are faulty and $n - t$ are correct. Each process $p_i$ starts with an input value $v_i \in \{0, 1\}$ and must eventually decide on a value $d_i \in \{0, 1\}$.
The system composed of processes $\Pi$, a given communication protocol $\Psi$, and decision function $\delta_i$, solves *terminating binary consensus* if the following is satisfied.

- TERMINATION. Every correct process eventually decides a value, i.e., each correct $p_i$ reaches a state where $\delta_i$ outputs $d_i$;

[4]The requirement of a terminating decision function can be understood and impose in multiple manners: either as an actual termination of the computational process, or simply that the decision value does not change after some point. We formalize such termination conditions in the context of robots in Chapter 4 in terms of their knowledge, and in Chapter 5 we make explicit a sort of stability requirement for terminal decision values

- VALIDITY. Every correct process decides a value that is proposed by at least one process, i.e., every decision $d_i \in \{v_1, \ldots, v_n\}$;
- CONSENSUS. All decided values are the same, i.e., $d_i = d_j$ for all correct processes $p_i, p_j$.

◀

In sufficiently unreliable networks, this level of consensus is in fact impossible, as shown in the celebrated FLP impossibility result, from Fischer, Lynch and Paterson [FLP85].

▶ **Proposition 2.18 (Impossibility of consensus in asynchronous networks with faults [FLP85]).** Consensus is impossible in an asynchronous distributed system with at least one crash fault. ◀

As evidenced by Proposition 2.18, consensus can be too strict a requirement when the network is unreliable. A more practical requirement is to demand only approximate agreement, a relaxation of consensus when it is possible to choose among a larger pool of values that should be sufficiently near to each other. Concretely, the *$\varepsilon$-agreement* task in a system of processes $\Pi = \{p_1, \ldots, p_n\}$ is a modification of binary consensus requiring, for each input global state $\mathbf{c}_I \in I$

- VALIDITY requires that the decisions $d_i$ are within the convex hull of the inputs, i.e., $d_i \in \{\sum_{i=1}^{n} \lambda_i v_i \mid v_i \in \mathbf{c}_I, \sum_{i=1}^{n} \lambda_i = 1\}$;
- AGREEMENT requires that the decisions are no more than a parameter $\varepsilon$ apart, i.e., for all $(d_i, d_j)$, $|d_i - d_j| < \varepsilon$.

With $\varepsilon$-agreement, it is possible for asynchronous and fault-tolerant systems to achieve approximate coordination, as it was shown in Dolev et al. [Dol+86] with the introduction of filtering schemes on top of an averaging algorithm.

We will shed some light on this taxonomy by casting it to a unified algebraic framework in Chapter 6, where immediate snapshots will be seen as a special case of dynamic networks, later generalized to include fault models.

### 2.2.2 MODELING ROBOTS

Robots are also present in the distributed computing literature. A review of the models and problems can be found in Flocchini et al. [FPS19], and is the main reference for this section. They are algorithmically described as mobile computing entities that follow a cycle of LOOK-COMPUTE-MOVE operations as they interact with a shared environment. The environment is either a graph or a continuous region of space, changing between combinatorial or geometrical approaches, e.g., [Flo+05] and [CMB06]. This sequence is known as the LCM cycle:

1. LOOK. The robot retrieves a snapshot of the other robots in the environment.
2. COMPUTE. The robot computes its next desired state from the information obtained during the LOOK operation.
3. MOVE. The robot moves to its desired state based on its latest computation.
4. WAIT. The robot does not perform any action, but remains observable.

Two models for robots running LCM cycles are predominant in the literature: oblivious, introduced by Suzuki and Yamashita [SY99], and luminous, introduced by Das et al. [Das+16]. They contrast in the means for communication and memory made available to the robots. Robots under the *oblivious robot model* (OBLOT), have no memory and can only obtain information from observing other robots in the LOOK step. Robots under the *luminous robot model* (LUMI), are enhanced with a set of programmable lights that can be used for both (indirect) communication and storing information for future COMPUTE steps. The lights are a physical element representing the exposed memory of a robot, and hence can be observed with the rest of the physical state of a robot during LOOK operations and are updated during a MOVE operation. We focus on the LUMI model for their capacity to perform intentional communication, which is closer in nature to the distributed systems we are interested in.

Uncertainty can be defined in the behavior of each step, such as the accuracy of the information or the actual moved distance. Other possible choices for constraints or adversarial behavior in the LCM model are listed in Table 2.1.

This algorithmic perspective on robot tasks provides a direct insight into the coordination problems that are present in *multi-robot tasks*, abstracting away

the physical and dynamical constraints. Two specially important examples are the gathering and exploration tasks, which illustrate the traditional separation between static and dynamic termination problems.

▶ **Definition 2.19 (Gathering task).** Let $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ be a set of robots in a graph $G = (V, E)$. Each robot has a position $v_i \in V$ and can be displaced in one MOVE operation to a neighboring vertex $\mathcal{N}_G(v_i) = \{v_j \in V \mid (v_i, v_j) \in E\}$. The system composed of robots $\mathcal{R}$ solves *exact gathering* if the following are satisfied.

- TERMINATION: every robot decides in a vertex of $G$ in a bounded number of LCM cycles;
- VALIDITY: the decided vertex cannot be decided in advance;
- GATHERING: all decided vertices are the same.

◀

It is also possible to relax exact gathering to an approximate version, where the GATHERING requirement changed to one of the following.

- EDGE-GATHERING: all decided vertices share an edge.
- CLIQUE-GATHERING: all decided vertices form a clique.

▶ **Definition 2.20 (Exploration task).** The system composed of robots $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ on a graph $G = (V, E)$ solves *terminating exploration* if the following are satisfied.

- EXPLORATION: all vertices are eventually visited;
- TERMINATION: the robots complete exploration in a bounded number of LCM cycles.

◀

Extra properties can be imposed in order to model more specific task scenarios. For instance, constrained environments where the space cannot accommodate more than one robot, such as the verification of some tunnel infrastructure, may impose the following.

- EXCLUSIVITY: no two robots visit the same vertex at the same time.

The termination property can be substituted by the following to describe a permanent surveillance requirement.

- PATROLLING: all vertices are visited infinitely often.

These fundamental problems can be extended to more complex coordination tasks. Gathering, for instance, is a building block towards pattern formation, where a specific geometrical disposition of the robots must be achieved rather than the same point in space. Similarly, flocking can be understood as a gathering task over other components of the robots' state, such as velocity and orientation. Surveillance consists of ensuring that some region of space is either free of intruders and that their detection is guaranteed otherwise, which can be reframed as exploration with a temporal component, as will be seen in Chapter 4.

One should note that the models used by the LCM literature are frequently too simplistic, as they completely abstract the physical dynamics involved in a real robotic system. While this is a necessary step for being able to tackle such complex systems, it creates a gap between theory and application that is hard to surpass. Chapter 3 expands on this contrasting relationship between physical and computational dynamics, deriving a sufficiently rich model to capture the information that must be preserved through an abstraction procedure.

### 2.2.3 CORRESPONDENCE OF MODELS

The community working with robots under the LOOK-COMPUTE-MOVE framework study problems analogous to the ones found in distributed computing. This similarity has been formally stated in the work of Alcántara et al. [Alc+19], which shows that any robot algorithm in the extended asynchronous luminous robot model (*EALR*), where the robots execute asynchronously and may start their execution at different moment, can be simulated in a wait-free shared memory model (WFSM). The simulation maps the lights of a robot to the shared memory registers, where the LOOK-COMPUTE-MOVE cycle translates to READ-COMPUTE-WRITE operations. The assumptions of the asynchronous scheduler match those of the wait-free, where any subset of robots or processes may be active at each round of execution.

▶ **Proposition 2.21 (Equivalence of robot task solvability in WFSM and EALR [Alc+19]).** A robot task $\mathbf{T}$ on graph $G$ is solvable in the EALR model with $n \geq 2$ robots tolerating $f$ failures if and only if $\mathbf{T}$ is solvable in the WFSM model with $n \geq 2$ processes tolerating up to $f$ failures. ◀

The equivalence of Proposition 2.21 becomes concrete when considering specific tasks. For instance, exact gathering on a graph, where the robots must meet on the same vertex, corresponds to consensus in distributed computing. Approximate agreement is a relaxation of consensus in the same manner edge or clique gathering relax exact gathering, where robots may be in any subset of vertices that meet the approximation property. This is a fundamental correspondence, as it opens space for the transfer of results and tools across communities. The connection is exemplified in the literature with Proposition 2.28, where the topological characterization of distributed tasks is used for robot tasks. Chapter 4 makes use of this correspondence to apply a logical framework commonly found in distributed computing to analyze task solvability.

## 2.3 MATHEMATICAL FRAMEWORKS

We present now some background on logical and topological frameworks that have been developed for analyzing distributed computing systems.

### 2.3.1 COMBINATORIAL TOPOLOGY IN DISTRIBUTED COMPUTING

Combinatorial and algebraic topology has found usage in distributed computing as a structured representation of the combinatorial state space of distributed systems. The connection between the two fields has been largely exploited for characterizing distributed task solvability. The main insight is in understanding the global states of the system as simplicial complexes and communication protocols as simplicial maps. An extensive presentation of the topic and the principal reference for this section is Herlihy, Kozlov and Rajsbaum [HKR14].

▶ **Definition 2.22 (Simplicial complex).** A *simplicial complex* $K = (V, S)$ consists of a set of vertices $V$ and a family of *simplices* $S$ of non-empty subsets of $V$, such that for all $X \in S$, $Y \subseteq X$ implies $Y \in S$. ◀

Given simplicial complexes $K = (V, S)$ and $K' = (V', S')$, a *simplicial map* $K$ and $K'$ is a map $f\colon K \to K'$ that sends vertices of $K$ to vertices of $K'$, such that the simplicial structure is preserved, i.e., if $X \in S$, then $f(X) \in S'$. The simplicial complex $K$ is said to be a *subcomplex* of $K'$, denoted $K \subseteq K'$, if $V \subseteq V'$ and $S \subseteq S'$.

A relation between simplicial complexes $K$ and $K'$ is defined as a *carrier map* $\Delta\colon K \to 2^{K'}$, a function defined on simplices $X \in S$, assigning to each simplex a subcomplex $\Delta(X) \subseteq K'$, respecting inclusions, i.e., if $X \subseteq Y$, then $\Delta(X) \subseteq \Delta(Y)$.

For a simplicial complex $K = (V, S)$, when two simplices $X, Y \in S$ are such that $Y \subseteq X$, it is said that $Y$ is a *face* of $X$. Simplices are called *facets* when they are maximal with respect to inclusion. A simplex $X \in S$ has *dimension* $|X| - 1$, and a simplex of dimension $n$ is called an $n$-simplex. Simplices of dimensions $1$ and $2$ are called edges and triangles, respectively. We identify the $0$-simplex $\{v\} \in S$ with the vertex $v \in S$. A simplicial complex is *pure* if all its facets are of the same dimension. A pure simplicial complex has the same dimension as its facets.

By defining a coloring of the vertices of a simplicial complex we form a chromatic simplicial complex, which will be the base structure for representing joint local states of a distributed system.

▶ **Definition 2.23 (Chromatic simplicial complex).** Let $A$ be a set of colors. A *chromatic simplicial complex* $K = (V, S, \chi)$ consists of a simplicial complex $(V, S)$ and a *coloring map* $\chi : V \to A$, such that for all $X \in S$, all vertices of $X$ have distinct colors. ◀

Both simplicial maps and carrier maps extend to chromatic simplicial complexes, with the added condition that colors must be preserved, as *chromatic simplicial maps* and *chromatic carrier maps*, respectively. That is, for a chromatic simplicial map $f$, one requires $\chi'(f(v)) = \chi(v)$ for all vertices $v$.

A distributed system composed of $n$ processes will possess, at any given moment, $n$ different local states. Each possible combination of local states consists of a global state. By labeling the set of vertices with colors corresponding to the processes, their global states can be associated to chromatic simplices of dimension $n - 1$, whose faces are partial views of subsets of processes. In this fashion, a pure chromatic simplicial complex of dimension $n - 1$ models a static view of the distributed system. With this mapping, we can now formally specify distributed task, defining the desired computation outcomes.

▶ **Definition 2.24 (Simplicial task specification).** A *distributed task* is a triple $\mathbf{T} = \langle \mathcal{I}, \mathcal{O}, \Delta \rangle$, where $\mathcal{I}$ and $\mathcal{O}$ are chromatic simplicial complexes containing all pos-

sible initial global states and final states, respectively, and $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ is a *task specification* chromatic carrier map. ◀

A task specification maps each possible initial global state of a distributed system in a given task to all allowed final global states. Similarly, the communication protocol employed by a distributed system can also be seen as a chromatic carrier map on global states to the reachable states in one step of the protocol. The protocol is formalized as follows.

▶ **Definition 2.25 (Iterated protocol).** A *simplicial protocol* is a chromatic carrier map $\Psi : \mathcal{A} \to 2^{\mathcal{B}}$, where $\mathcal{A}$ and $\mathcal{B}$ are chromatic simplicial complexes. The $n$-iterated application of the protocol is denoted $\Psi^n$, and its image $\Psi^n(\mathcal{A})$ is called the *protocol complex*. ◀

The protocol complex contains all possible states of the system after an iteration of the protocol. A protocol solves a distributed task if there is a decision function that maps every global state reachable from an *input state* to some *output state*. This concept can also be modeled in the simplicial framework.

▶ **Definition 2.26 (Simplicial task solvability).** The protocol $\Psi$ *solves* the task $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \Delta)$, for some $n \in \mathbb{N}$, if there exists a chromatic simplicial map $\delta \colon \Psi^n(\mathcal{I}) \to \mathcal{O}$, such that for all $X \in \mathcal{I}$, $\delta(\Psi(X)) \subseteq \Delta(X)$. ◀

Task solvability can also be understood process-wise. Note that such decision function is local, that is, it can only use local state of a process. The requirement of the decision to be a function enforces process' output to be identical across the global states it is found with the same local state[5]. This constraint gives rise to the epistemic interpretation of chromatic simplicial complexes, reviewed in Section 2.3.2.

An interesting feature of this topological formulation is that connectivity appears as an important property in determining the solvability of consensus-like tasks, as they are related to continuous deformations from arbitrary configurations to solving ones. Finding meaning in topological properties and invariants of this sort is among the main benefits of adopting the simplicial framework for

[5]Note that there is yet another formulation of distributed task and their solvability condition using csets, as it will be seen in Chapter 6, Example 6.22.

analyzing distributed tasks. The following proposition illustrates this connection between topology and solvability, showing that path-connectedness of the protocol complex prevents consensus. We will see similar arguments in its generalizations in Chapters 5 and 6.

▶ **Proposition 2.27 (Impossibility of consensus for path-connected protocol complex [HKR14]).** Let $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \Delta)$ be a consensus task where not every process starts with the same input in every configuration of $\mathcal{I}$. If $\Psi$ is a protocol such that $\Psi(\sigma)$ is path-connected for all simplices in $\mathcal{I}$, then it cannot solve $\mathbf{T}$. ◀

Furthermore, following from the connection between distributed computing and mobile robotics shown in Section 2.2.3, the same theory of simplicial task solvability can be applied to certain robot tasks expressed in the LOOK-COMPUTE-MOVE framework.

▶ **Proposition 2.28 (Simplicial robot task solvability [Alc+19]).** A robot task $(\mathcal{I}, \mathcal{O}, \Delta)$ is solvable for $n$ robots in the extended asynchronous luminous model tolerating $n-1$ crash failures if and only if there a chromatic simplicial map $\delta$ from $Subd(\mathcal{I})$ to $\mathcal{O}$, such that for every simplex $\sigma$, $\delta(Subd(\sigma)) \subseteq \Delta(\sigma)$. ◀

### 2.3.2 EPISTEMIC LOGICS AND CHROMATIC SIMPLICIAL COMPLEXES

Epistemic logic provides a framework for reasoning about knowledge in distributed systems. While its philosophical foundations trace back to Hintikka [Hin62], the application to distributed computing was developed by Fagin, Halpern, Moses and Vardi [Fag+95], where each process only has partial information about some global state. Informally, a process knows a fact when it holds in every global state compatible with its local state. This sort of information is encoded in a formal language, which is defined for a given set of possible facts, known as atomic propositions, from the following grammar.

▶ **Definition 2.29 (Language of epistemic logic).** Let At be a countable set of *atomic propositions* and let $A$ be a set of processes. The *language of epistemic logic* is generated by

the following grammar.

$$\varphi \coloneqq p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_a\varphi \mid D_U\varphi \mid C_U\varphi \qquad p \in \mathsf{At}, a \in A, U \subseteq A$$

◀

The knowledge operator $K_a\varphi$ reads "process a knows $\varphi$", and a dual possibility operator is defined as $P_a\varphi := \neg K_a \neg\varphi$. The operator $D_A$ stands for "the group A has distributed knowledge of $\varphi$", that is, $\varphi$ would be known if the processes in $A$ could combine their local information. The syntax of epistemic logic consists of propositional formulas extended with a knowledge operator for each process. The *epistemic logic* $S5_n$ is defined by a set of axioms that captures properties of knowledge (truth, positive introspection, negative introspection). One of the main features of epistemic logic is access to higher-order knowledge, that is, knowledge about knowledge. For instance, from a snapshot of the system global states, one could verify if a formula saying that "process $a$ knows that process $b$ knows $\varphi$", written simply $K_aK_b\varphi$, holds true. *Common knowledge* extends individual and distributed knowledge to capture publicly shared information within a group. The operator $C_A\varphi$ is interpreted as all processes know $\varphi$, and all processes know that all processes know $\varphi$, and so on.

Halpern and Moses [HM90] showed that achieving coordination in distributed systems often requires common knowledge of the decision, establishing a fundamental connection between the logical analysis and impossibility results in distributed computing. The problem of coordinated behavior is then recast into assessing what information each process can distinguish to be true or possibly true. In order to assess truth, one provides a model in which a formula is verified to be satisfied or not. The usual model for $S5_n$ epistemic logic is a Kripke model.

▶ **Definition 2.30 (Kripke frame and Kripke model).** A *Kripke frame* for a set of processes $A$ is a pair $(W, \{\sim_a\}_{a\in A})$, where $W$ is a set of *possible worlds* and each $\sim_a \subseteq W \times W$ is an equivalence relation. A *Kripke model* is a triple $M = (W, \{\sim_a\}_{a\in A}, \ell)$, where $\ell\colon \mathsf{At} \to 2^W$ is a valuation function assigning to each atomic proposition $p \in \mathsf{At}$ the set of possible worlds in which it holds. ◀

The truth value of a formula $\varphi$ is then given by the following semantics, where $S5_n$ is sound and complete for the class of frames where $\sim_a$ is an equivalence

relation [Fag+95]. Furthermore, a group indistinguishability relation can be added as the intersection of the individual ones, it too satisfying the $S5$ axioms.

▶ **Definition 2.31 (Kripke semantics of $S5_n$).** Let $M = (W, \{\sim_a\}_{a\in A}, \ell)$ be a Kripke model. For a possible world $w \in W$, the *Kripke semantics* is defined as follows:

$$
\begin{aligned}
M, w \models p \quad & \text{iff} \quad w \in \ell(p) \\
M, w \models \neg\varphi \quad & \text{iff} \quad M, w \not\models \varphi \\
M, w \models (\varphi \wedge \psi) \quad & \text{iff} \quad M, w \models \varphi \text{ and } M, w \models \psi \\
M, w \models K_a\varphi \quad & \text{iff} \quad \forall w' \in W \text{ s.t. } w \sim_a w', M, w' \models \varphi \\
M, w \models D_A\varphi \quad & \text{iff} \quad \forall w' \in W \text{ s.t. } w \bigcap_{a\in A} \sim_a w', M, w' \models \varphi \\
M, w \models C_A\varphi \quad & \text{iff} \quad \forall w' \in W \text{ s.t. } w \left(\bigcup_{a\in A} \sim_a\right)^* w', M, w' \models \varphi
\end{aligned}
$$

◀

An interesting connection has been shown between models from Kripke frames and models from chromatic simplicial complexes by Goubault, Ledent and Rajsbaum [GLR21]. More precisely, a *simplicial complex model* is a tuple $M = (V, S, \chi, \ell)$, where $K = (V, S, \chi)$ is a chromatic simplicial complex and $\ell\colon \mathsf{At} \to 2^{\mathrm{facets}(K)}$ is a valuation map defined on facets. The following semantics is used then.

▶ **Definition 2.32 (Simplicial complex semantics of $S5_n$).** Let $M = (V, S, \chi, \ell)$ be a simplicial complex model. For a facet $X \in S$, the *simplicial complex semantics* is defined as follows:

$$
\begin{aligned}
M, X \models p \quad & \text{iff} \quad X \in \ell(p) \\
M, X \models \neg\varphi \quad & \text{iff} \quad M, X \not\models \varphi \\
M, X \models (\varphi \wedge \psi) \quad & \text{iff} \quad M, X \models \varphi \text{ and } M, X \models \psi \\
M, X \models K_a\varphi \quad & \text{iff} \quad \forall X' \in \mathrm{star}_a(X), M, X' \models \varphi \\
M, X \models D_U\varphi \quad & \text{iff} \quad \forall X' \in \bigcap_{a\in U} \mathrm{star}_a(X),\ M, X' \models \varphi \\
M, X \models C_A\varphi \quad & \text{iff} \quad \forall X' \in \left(\bigcup_{a\in A} \mathrm{star}_a\right)^* (X),\ M, X' \models \varphi
\end{aligned}
$$

where $\mathrm{star}_a(X) = \{X' \in \mathrm{facets}(K) \mid \exists\, v \in X \cap X' \text{ such that } \chi(v) = a\}$ is the star operator ◀

Note that this valuation function constrains the facts to be reasoned only at the level of global states, which will be generalized to arbitrary sets of processes with the maximal epistemic covering model in [Gou+23], allowing to model group knowledge.

The simplicial complex model has been shown equivalent to the Kripke model in the case where all processes are always present in all global states.

▶ **Proposition 2.33 (Equivalence of Kripke and simplicial models [GLR21]).** The category $\mathbf{SM}_{A,\mathsf{At}}$ of pure simplicial complex models is equivalent to the category $\mathbf{KM}_{A,\mathsf{At}}$ of proper Kripke models, when both have the same set of processes $A$ and countable set of atomic propositions $\mathsf{At}$. ◀

This simplicial approach to knowledge has also been extended to *dynamic epistemic logic* [vDHK07] by Goubault, Ledent and Rajsbaum [GLR21], integrating operators for information-changing events such as public announcements. It is worth noting that the topological and logical approaches do not always yield the same results. Van Ditmarsch et al. [vDit+21] showed that certain impossibility results provable through geometric methods, such as for equality negation tasks, cannot be established using the epistemic logic framework described above. Nevertheless, extensions of epistemic logic with fixed point operators have been shown capable of proving impossibilities for tasks of comparable difficulty, such as $k$-set agreement, as shown by Nishimura [Nis24].

### 2.3.3 GENERALIZATION TO CHROMATIC SEMI-SIMPLICIAL SETS

The simplicial complexes, as shown in Section 2.3.1, are special cases of simplicial sets. Two limitations of chromatic simplicial complexes motivate this generalization. First, since a simplex is uniquely determined by its set of vertices, two distinct executions that produce the same local states for the same subset of processes are necessarily identified, possibly losing information about the execution history. Second, the knowledge operator of the simplicial complex model (Definition 2.32) are defined only on facets, preventing the modeling of distributed knowledge for

arbitrary subgroups of processes. Chromatic semi-simplicial sets address both limitations, as we now review.

They have been recently studied in the context of distributed computing by Goubault et al. [Gou+23], which serves as the main reference for this section, alongside the more in-depth introduction by Goerss and Jardine [GJ09]. The following exposition will make use of the language of category theory, and will be followed by a concrete example.

We first define the category $\Gamma$, which provides the structure and relationship between simplices.

▶ **Definition 2.34 (Category $\Gamma$).** Let $A$ be a set of processes. The category $\Gamma$ contains the following data:

- Objects: possibly empty subsets $U \subseteq A$;
- Morphisms: a unique *coface* map $\delta_{U,V} : U \to V$ in $\Gamma$ whenever $U \subseteq V$.

Composition and identity are given by the transitivity and reflexivity of inclusion maps. ◀

The category $\Gamma$ playes the role of the standard simplex category $\Delta$ classically used, whose objects are the finite ordinals $[n] = \{0, \ldots, n\}$ with order-preserving maps. The difference is that objects of $\Gamma$ are subsets of a fixed set of process names $A$, so that vertices carry an identity (their color) rather than an index. This is the chromatic variant introduced in [Gou+23], and the restriction to injective morphisms makes $\Gamma$ a semi-simplicial indexing category, as there are no degeneracy maps, suitable to the fact that each process maintains a distinct identity throughout executions. Chromatic semi-simplicial sets are sets of maps assigning to each object of $\Gamma$, a set of simplices. This assignment is functorial, i.e., it must respect the coface maps between objects.

▶ **Definition 2.35 (Chromatic semi-simplicial set).** Let $A$ be a set of processes and $\Gamma$ its category of simplices. A *chromatic semi-simplicial set*, abbreviated cset, is a functor $X : \Gamma^{\mathrm{op}} \to \mathbf{Set}$ and a *morphism of csets* is a natural transformation between such functors. ◀

Concretely, for each subset $U \subseteq A$, the elements of $X(U)$ are sets of $U$-simplices. For $U \subseteq V$ and coface map $\delta_{U,V}$, the elements of $X(\delta_{U,V})$ are face maps $\partial X(V) \to X(U)$. The *standard $U$-simplex* $\Gamma[U]$ is the representable presheaf

$\Gamma(--, U)$, the image of $U$ by the *Yoneda embedding* $y \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ [Rie11]. The category of chromatic semi-simplicial sets is then the functor category $[\Gamma^{\mathrm{op}}; \mathbf{Set}]$ of presheaves on $\Gamma$.

▶ **Example 2.36 (Chromatic semi-simplicial set).** Given a set of processes $A = \{r, g, b\}$ identified with the colors red, green and blue, the figure below depicts cset $X(A)$. A cset $X$ is a map taking each subset $U \subset A$ to a set of simplices, while the boundary maps $\delta_U$ give the equations that describe how to glue these simplices. It contains eight 0-simplices (vertices), eight 1-simplices (edges) and two 2-simplices (triangles). More precisely, the sets $X(\{r\})$, $X(\{g\})$ and $X(\{b\})$ contain two elements each, which are the red, green and blue vertices, respectively. The edges are the elements of $X(\{r, g\})$, $X(\{g, b\})$, $X(\{r, b\})$. Note that the two edges between vertices red and blue are counted separately, a feature of chromatic simplicial sets that is not possible in chromatic simplicial complexes. $X(\{r, g, b\})$ corresponds to the two triangles. The set $X(\varnothing)$ contains the two dashed regions, which can be understood as generalized connected components. ◀

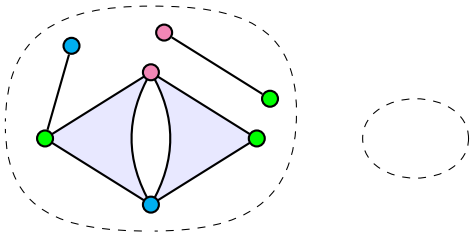


Figure 2.2
Chromatic semi-simplicial set from [Gou+23; Gou+25].

As with chromatic simplicial complexes, chromatic semi-simplicial sets have been used in [Gou+23] to model sets of global states and distributed protocols, with the local state of processes in its vertices, colored by their identities. Similarly, csets also serve as models for epistemic logic, which are now capable of assigning data and defining knowledge for arbitrary subsets of processes $U \subseteq A$, rather than only for maximal configurations. This representation will be used in Chapter 6 to define categorically the iterative application of a communication protocol in a set of global states, whose resulting structure will serve as another model for epistemic logic.

### 2.3.4 NETWORK DYNAMICS VIA CELLULAR SHEAVES

Sheaves offer a way to reason about the global properties of data specified locally, i.e., that cannot be immediately presented in a global picture. Curry [Cur14], building on earlier combinatorial approaches to sheaf theory [Zee62; Kas84], developed cellular sheaves as a combinatorial and computationally tractable framework that has since found applications in general multi-agent systems, from sensor integration [Rob17] to opinion dynamics [HG21], quantum contextuality [AB11] and distributed optimization [HG19a]. In the case of networked systems, where a communication graph uses vertices to represent agents and edges to encode interaction links, cellular sheaves model the data defined agent-wise. Our main reference regarding the general theory of cellular sheaves is Curry [Cur14], while its spectral theory is developed by Hansen [Han20].

The first information needed to define a cellular sheaf is its base space. The name derives from its usage of regular cellular complexes for that purpose, special case of a cellular complex where its topological data is completely determined by its combinatorial data.

▶ **Definition 2.37 (Regular cellular complex).** A *regular cellular complex* consists of a topological space $X$ alongside a partition into *cells* $\{X_\alpha\}_{\alpha \in P_X}$, where for each cell $X_\alpha$ there is a homeomorphism $D^{d_\alpha} \to \overline{X}_\alpha$ from the closed $d_\alpha$-dimensional ball to its closure, mapping the interior of $D^{d_\alpha}$ to $X_\alpha$, and $X_\alpha$ intersects finitely many other cells [Hat01]. The *face* incidence relation $\alpha \leqslant \beta$, defined by $X_\alpha \subseteq \overline{X}_\beta$, forms a finite poset $P_X$ equipped with a dimension function $\dim\colon P_X \to \mathbb{N}$, which completely determines the combinatorial structure of the complex. ◀

Intuitively, regular cellular complexes generalize graphs to higher dimensions, where vertices correspond to $0$-cells, edges to $1$-cells, faces to $2$-cells, and so on. The face incidence relation between these cells, e.g., as a vertex is incident to an edge, defines the poset $P_X$ that captures how the pieces fit together. This poset structure can be viewed as a category, giving rise to the notion of a *cell category* [Cur14], where morphisms encode the face relations and composition is given by transitivity[6]. For multi-agent systems modeled by graphs, this categorical

[6]These will be developed further in Chapter 5, as for this preliminary exposition we focus on the simpler case derived from graphs.

structure is particularly simple.

▶ **Example 2.38 (Cell category from an undirected graph).** An undirected graph $G = (V, E)$ gives rise to a cell category $\mathbf{Cell}(G)$. Note that an edge $v \overset{e}{\leftrightarrow} v'$ consists of an unordered pair of vertices $\{v, v'\}$, and the face incidence relation $v < e$ of a vertex to an edge satisfies the inclusion $v \hookrightarrow \{v, v'\} = e$. The category $\mathbf{Cell}(G)$ is then constructed with an object for each vertex $v \in V$, and for each edge $e \in E$, alongside morphisms $v \to e$ whenever $v < e$. Objects obtained from vertices are called $0$-cells, and those from edges $1$-cells. This construction preserves the information of the graph, while adding a categorical structure. ◀

A cellular sheaf is then a functor from the poset $P_X$, viewed as a category, to a category of data, such as $\mathbf{Set}$ or $\mathbf{Vect}_{\mathbb{K}}$. For multi-agent systems modeled by graphs, this reduces to assigning data to vertices and edges with maps between then, which we now make precise.

▶ **Definition 2.39 (Cellular sheaf on a graph).** A set valued *cellular sheaf on a graph* $G$ is a functor $\mathcal{F} \colon \mathbf{Cell}(G) \to \mathbf{Set}$. Explicitly, it is a family of functions defined on the graph that assigns:

- to each vertex $v \in V$, a set $\mathcal{F}(v)$ called the *stalk* at $v$;
- to each edge $e \in E$, a set $\mathcal{F}(e)$ called the *stalk* at $e$;
- to each incidence map $v \to e$ in $\mathbf{Cell}(G)$, a function $\mathcal{F}_{v \to e} \colon \mathcal{F}(v) \to \mathcal{F}(e)$ called the *restriction map*.

The functoriality condition ensures that restriction maps compose correctly and preserve identities. ◀

▶ **Example 2.40 (Set-valued cellular sheaf over a graph).** Consider the graph $G$ with vertices $V = \{v_1, v_2, v_3, v_4\}$ and edges $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_4)\}$. Figure 2.3 shows a set valued cellular sheaf defined on $G$, whose values are taken from $\{x, y, z\}$. The graph below is interpreted as a cell category $\mathbf{Cell}(G)$, with $0$-cells (vertices) shown individually and $1$-cells (edges) shown as horizontal bars. Morphisms are the face incidence inclusions, and are omitted in favor of readability. The sets of values above the vertical bars are the stalks, which are associated to the respective cells below them. The restriction maps are depicted as curved arrows, and two of them shown explicitly. ◀

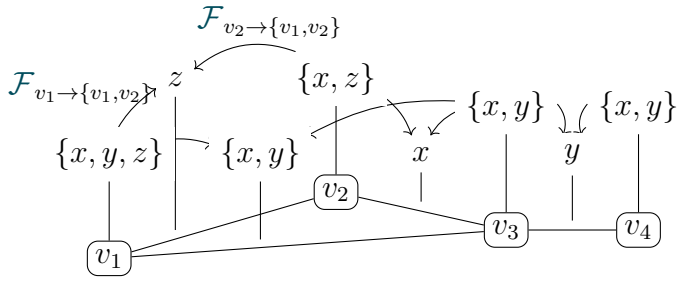



Figure 2.3
A set-valued sheaf over a cell category obtained from a graph.

In networked multi-agent systems specified on a communication graph, stalks at vertices describe the space of possible data of each agent, while stalks on edges define the space in which they may express their local information when communicating. Other types of data have been considered, such as vector spaces [HG21] and lattices [Rie23], whose additional structure is useful when analyzing interaction dynamics.

One of the main benefits of the cellular sheaf model for multi-agent systems is its built-in notion for "consistency" of data, i.e., data that respects every possible restriction map. Sections represent data that can be agreed upon by all agents that can communicate to each other, where agreement is encoded in the definition of the restriction maps.

▶ **Definition 2.41 (Section of a cellular sheaf).** Let $\mathcal{F}\colon \mathbf{Cell}(G) \to \mathbf{Set}$ be a set valued cellular sheaf. A *section* of $\mathcal{F}$ is a choice of elements $s = \{s_c \in \mathcal{F}(c) \mid c \in \mathbf{Cell}(G)\}$, such that for every morphism $\alpha \to \beta$ in $\mathbf{Cell}(G)$, the restriction maps satisfy

$$\mathcal{F}_{\alpha \to \beta}(s_\alpha) = s_\beta \tag{2.8}$$

The set of all sections is denoted $\Gamma(G, \mathcal{F})$. ◀

Hansen and Ghrist [HG19b; Han20] showed that the spectral theory of graphs can be lifted to cellular sheaves, generalizing concepts such as the graph Laplacian (Definition 2.7) that were already used in multi-agent systems to a cleaner framework. Defining the sheaf Laplacian requires additional algebraic structure on the data. We now restrict to cellular sheaves valued in finite-dimensional vector spaces of a field $\mathbb{K}$, where the linear structure enables a notion of "discrepancy" between adjacent data.

▶ **Definition 2.42 (Sheaf Laplacian).** Let $\mathcal{F}\colon \mathbf{Cell}(G) \to \mathbf{Vect}_{\mathbb{K}}$ be a cellular sheaf on a graph $G = (V, E)$ with values in finite-dimensional vector spaces over $\mathbb{K}$. The restriction maps $\mathcal{F}_{v \to e}\colon \mathcal{F}(v) \to \mathcal{F}(e)$ assemble into a *coboundary* linear map

$$\delta\colon \bigoplus_{v \in V} \mathcal{F}(v) \to \bigoplus_{e \in E} \mathcal{F}(e) \tag{2.9}$$

defined component-wise for each edge $e = (v, v') \in E$ by maps $\delta_e(x) = \mathcal{F}_{v \to e}(x_v) - \mathcal{F}_{v' \to e}(x_{v'})$.

The *sheaf Laplacian* is the operator $L_{\mathcal{F}} = \delta^* \delta$, where $\delta^*$ is the adjoint of $\delta$. ◀

We can see through components of the coboundary map $\delta_e(x) = \mathcal{F}_{v \to e}(x_v) - \mathcal{F}_{v' \to e}(x_{v'})$ how the sheaf Laplacian measures the lack of coherence among the values $x_v, x'_v \in \mathcal{F}(v) \oplus \mathcal{F}(v')$ in adjacent vertices. Those are sent to their common edge to be compared, while its adjoint map sends the values back to the vertices' "point of view". For cellular sheaves valued on vector spaces, this algebraic structure enables the spectral analysis of data consistency.

▶ **Proposition 2.43 (Convergence to sections [HG19b]).** Let $\mathcal{F}$ be a cellular sheaf on a connected graph $G$ and $L_{\mathcal{F}}$ be its sheaf Laplacian. The dynamics $\dot{x} = -L_{\mathcal{F}}(x)$ converges asymptotically to its global sections $\Gamma(G, \mathcal{F})$ for any initial assignment $x_0 \in \bigoplus_{v \in V} \mathcal{F}(v)$. ◀

The convergence result of Proposition 2.43 shows one possible generalization of the averaging algorithms used in both networked mobile robotics (Section 2.1) and distributed computing Section 2.2, where iterative updates based on local measurements of discrepancies lead to globally consistent solutions. The diffusion effect of the sheaf Laplacian has been extended by Riess and Ghrist [Rie23; GR22] to work on lattice valued sheaves as a Tarski Laplacian, and for "semantic" diffusion over Kripke frames as a Kripke Laplacian. Ghrist et al. [Ghr+25] generalized it to data in general Quantale-enriched categories, named the Lawvere Laplacian after Lawvere metric spaces. We will employ the framework of cellular sheaves in Chapter 5, where we construct task sheaves as an encoding of distributed task specifications on a space of possible system executions, rather than communication networks.

Table 2.1
Partial glossary of LCM terminology.

| Property | Variant | Description |
|---|---|---|
| Memory | **Oblivious** | No memory, all information is lost between rounds |
| | **Luminous** | Lights can be used to preserve information across rounds |
| Communication | **Silent** | No communication between robots |
| | **Luminous** | Information passed indirectly by retrieving lights during LOOK |
| Visibility | **Myopic** | LOOK retrieves a partial snapshot parametrized by view distance |
| | **Unlimited visibility** | LOOK retrieves a snapshot of the entire environment |
| Identity | **Anonymous** | No unique identifier |
| | **Indistinguishable** | No external identifier |
| Movement | **Rigid** | MOVE is always fully executed |
| | **Non-rigid** | MOVE can be interrupted, parametrized by a minimum distance |
| Dynamics | **Holonomic** | MOVE can be any arbitrary destination |
| | **Non-holonomic** | MOVE has to respect dynamic constraints, e.g., Dubins car |
| Orientation | **Oriented** | There is a global coordinate system that all can access |
| | **Disoriented** | Only different local coordinate systems |
| | **Chiral** | Different local coordinate systems, but with common north |
| Synchronicity | **ASYNC** | Scheduler, no shared notion of time |
| | **k-ASYNC** | Scheduler, the robots are at most $k$ rounds apart |
| | **SSYNC** | Scheduler, a subset of the robots activated at each round |
| | **FSYNC** | Scheduler, all robots are activated at each round |

*"Il y a un réel contraste entre la violence de l'acte de représenter et le calme intérieur de la représentation elle-même."*

"There is a real contrast between the violence of the act of representing and the inner calm of the representation itself."

— Jean-Luc Godard, Le Livre d'image [God18]

# 3

# Models for distributed robotics

Distributed multi-robot systems live at the intersection of mobile robotics and distributed computing, and benefit from the development of different communities tackling complementary problems. This double heritage has as a consequence the plethora of models used to depict similar robotic systems. As shown in Chapter 2, dynamical systems are the prevalent way to capture the physical and dynamical constraints, while the computational capacities are more often modeled as transition systems and their related abstractions. While these are natural choices, as different levels of abstraction allow for focus on different aspects of the robots, their incompatibility requires translation layers with explicit assumptions. In this chapter, we propose such a translation layer, motivated by the desire of bringing distributed computing results to the applied robotics communities, as well as providing more tangible foundations to the theoretical developments found in the latter. This takes the form of two models for multi-robot systems: a switched hybrid dynamical system, where the modes model different possible activations of groups of robots, and compositional automata with an environment, where the physical and adversarial behavior are pushed to the environment and the

computational steps put in focus.

This chapter corresponds to an extended version of Sections 2 and 3 of [Cig+25], a joint work with Giorgio Cignarale, Stephan Felber, Eric Goubault and Hugo Rincon Galeana.

**Related work** The models developed in this chapter are built from converging lines of research across control theory and distributed computing. The main difficulty lies at the different paradigms used by the two communities: dynamical systems theory and a variety of computing models. Pioneer work includes Olfati-Saber, Fax and Murray [OFM07] that showed consensus in a common framework joining control theory and multi-agent systems through graph theory. A similar approach is developed under the mean subsequence-reduced family of algorithms, where fault-tolerant averaging is used for continuous dynamical systems, as reviewed in Ishii, Wang and Feng [IWF22]. Still on the algorithmic side, Alcántara et al. [Alc+19] showed the unification technique of reducing the robot task of (approximate) gathering to the well-understood (approximate) consensus in distributed computing. While we too are interested in transferring methods and results across fields, we develop now a unification on the level of models, so to later reap the benefits of a translation layer between fields.

The time path formalism we introduce in Section 3.1 is inspired by vector clocks, introduced by Fidge [Fid88] and Mattern [Mat88], and hybrid clocks of Kulkarni [Kul+14], alongside the geometric models of concurrency, such as those of Goubault, Mimram and Tasson [GMT18] and Fajstrup et al. [Faj+16]. We simplify this approach to focus specifically in a relationship between local and global time to be used both by a hybrid system and a compositional automata.

The depiction of multi-agent systems as hybrid systems is standard for cyber-physical systems, as has been recently reviewed by Postoyan et al. [Pos+24]. Switched systems theory has been brought to multi-agent systems to model asynchronous modes, as in Lee and Macias [LM22]. We continue from this usage, where our modes follow the block companion form discussed in De et al. [De +20], and we extend the parametrization of the modes with the time path above.

The computational perspective developed in Section 3.3 fits within the LOOK-COMPUTE-MOVE paradigm used for robotics within distributed computing as a luminous model, introduced by Das et al. [Das+16], where robots are endowed

with communication capacities. Our specific instantiation via state machines is largely inspired by the I/O automaton introduced by Lynch [Lyn97], especially for its clear interfaces for composability. It has had several extensions, among which we note the hybrid I/O automaton [LSV03] that provides a modeling of cyber-physical system, with discrete and continuous state variables. While it is more direct at modeling robotic systems, our separation of robot and environment discrete state machines, handling separately the epistemic and ontic states, serves better our goals of unification with more standard distributed computing literature and leveraging the existing logical analysis.

**Chapter organization** This chapter sets the base modeling choices. It starts by defining in Section 3.1 a common notion of global and local time, as well as their relationship. A model for distributed multi-robot systems, equipped with communication and perception capacities, as a switched hybrid dynamical system is provided in Section 3.2. Its counterpart as a distributed state machine with environment is described in Section 3.3. Both are accompanied by their respective notions of trajectory and execution, solution set and run, which are used in Section 3.4 to prove that our change of models preserves epistemic information under reasonable assumptions, allowing us to reason with the more convenient automata for the remainder of this thesis.

## 3.1 TIME

Distributed systems are composed of multiple machines that need to cooperate. This multiplicity is evidenced by separate computational units, individual memory, and physical distance. All of those enforce a notion of locality, as each machine has access only to its own state and must actively acquire information about the rest. Information can be *local*, when it is true only in some portion of the system, or *global*, when it is true for the entire system. For local information to be useful to others, it has to be shared and situated within a common frame of reference. The most fundamental such frame is time. While most characteristics of a system can be made the same by construction, the non-negligible separation of the machines does not allow for a shared notion of time without some synchronizing procedure.

The problem of relating time in distributed systems has been explored since Lamport [Lam78] first observed that the different local times naturally form a partial ordering under the "happened before" relation. Since then, the need to synchronize physical clocks was superseded by the ordering of simpler *logical clocks*, such as counters. The introduction of *vector clocks* by Fidge [Fid88] and Mattern [Mat88] enabled to fully capture causality by maintaining an estimation of the local time by all machines, which was shown asymptotically optimal by Charron-Bost [Cha91]. Gaps in the theory and application of distributed application brought the interest to also use the global physical clock, leading to the *hybrid logical clocks* (HLC) introduced by Kulkarni [Kul+14].

In order to model distributed multi-robot systems, we are interested in tracking both the global execution time of a system and its relation to the local times of the robots, while keeping the full causality relation between these values. Moreover, the usage of physical machines with an inherently continuous dynamics brings the need to use continuous variables in the modeling of the system time. In other words, we are interested in a combination of vector clocks and HLCs with a continuous global clock.

Inspired by these previous approaches and the models of directed topology for concurrency (e.g., [Faj+16]), we introduce time paths as a geometric model of the relation between local and global time. By joining the different local logical clocks as coordinates of a common space $\mathbb{R}^n_+$, any monotonic increasing path rooted in the origin defines a possible execution order of the system.

▶ **Definition 3.1 (Time path).** For a system of $n$ robots, a *time path* is a differentiable map $\rho$, from $[0, T]$ to $\mathbb{R}^n_+$, for some $T \geq 0$, where $t \in [0, T]$ is the global time, that has the following properties:

- **(initialisation)** $\rho(0) = (0, \ldots, 0)$.
- **(*monotonicity*)** For any $t, t' \in [0, T]$ with $t \leq t'$, $\rho_i(t) \leq \rho_i(t')$, for all components $i \in [1, \ldots, n]$.
- **(*rectification*)** For all $t \in [0, T]$, the global time $t$ is the maximum of all local times, i.e., $t = \max(|\rho_1(t) - \rho_1(0)|, |\rho_2(t) - \rho_2(0)|, \ldots, |\rho_n(t) - \rho_n(0)|)$.

◀

Each coordinate $\rho_i(t)$ tracks the local time of robot $i$ at global time $t$. Note that the monotonicity property ensures that no local time decreases as global time progresses. The rectification makes the global time reflect the progress of the fastest robot, which guarantees that at least one robot takes a step at any given time, preventing the entire system to be idle. We can now define a scheduler that describes the possible activation orders of a system.

▶ **Definition 3.2 (Scheduler).** Let $\mathcal{P}$ be the set of all time paths for some $T \geq 0$. For a system of $n$ robots, a *scheduler* $\mathcal{S}$ is a non-empty subset of $\mathcal{P}$, i.e., $\mathcal{S} \in 2^{\mathcal{P}} \setminus \{\varnothing\}$. ◀

We can recover the traditional synchronous, $k$-synchronous and asynchronous schedulers from the distributed computing literature [Lyn97].

▶ **Example 3.3 (Synchronous scheduler).** The *synchronous scheduler*, denoted $\mathcal{S}_{sync}$, is the singleton set containing only the time path where all local times advance in lock-step:

$$\mathcal{S}_{sync} = \{\rho \mid \forall t \in [0, T], \rho(t) = (t, t, \ldots, t)\}$$

◀

▶ **Example 3.4 ($k$-synchronous scheduler).** The *$k$-synchronous scheduler*, denoted $\mathcal{S}_{ssync}$, bounds the difference in local time of the fastest and slowest execution by $k$, i.e.,

$$\mathcal{S}_{ssync} = \{\rho \mid \forall t \in [0, T], \max_{i \in [n]} \rho_i(t) - \min_{i \in [n]} \rho_i(t) \leq k\}$$

◀

▶ **Example 3.5 (Asynchronous scheduler).** The *asynchronous scheduler*, denoted $\mathcal{S}_{async}$, is the set of all possible time paths $\mathcal{P}$. ◀

The asynchronous scheduler is, by definition, the set of all time paths. The scheduler structure becomes meaningful when restricting to subsets of $\mathcal{P}$, as in the synchronous and $k$-synchronous cases.

The time path abstraction allows for the scheduler to capture the precise relationship between local and global times. We can now define the set of active robots according to a time path. This construction will be essential to define the modes of switching dynamics modeling a multi-robot system.

▶ **Definition 3.6 (Activation set).** For a time path $\rho$, the *set of active robots* at time $t$, denoted $\mathrm{active}_\rho(t)$, is the set of robots that advance their local time at global time $t$, i.e.,

$$\mathrm{active}_\rho(t) = \{i \in [n] \mid \exists \varepsilon > 0, \lfloor \rho_i(t+\varepsilon) \rfloor > \lfloor \rho_i(t) \rfloor\}$$

where $\lfloor t \rfloor$ denotes the *floor function*, i.e., the greatest integer less than or equal to $t$. ◀

Equivalently, a robot $i$ is active at $t$ when its discretized local time $\lfloor \rho_i \rfloor$ is about to increment, i.e., when it takes a computational step at time $t$.

## 3.2 ROBOTS, DYNAMICAL SYSTEMS

The modeling of distributed robotic systems encompasses both their physical evolution and computational processes. The distinction becomes evident when we consider that robots operate with partial and uncertain information, regarding both the world and themselves, where their decisions reflect only the information they have available, its belief about the environment. This gap between what is *true* and what is *known* motivates us to extend the dynamical system framework of robots from Section 2.1 by partitioning their state space and dynamics into ontic and epistemic components.

The *ontic* component refers to the physical reality, that of the robot and of the environment it is inserted in. The ontic state consists of the physical state of the robot, containing all the information needed to describe how it is found in the world. It is updated through the usage of actuators: devices that convert some electrical input into mechanical motion, allowing for the robot to move and interact with its environment. Some examples are the traction motors used to spin the wheels of a car, a pump that regulates the water of a submarine ballast and servo motors that positions the joints of a robotic arm.

The *epistemic* component consists of the robot's knowledge, the resources it has available for every computation and decision it takes. The epistemic state consists of the working memory of the robot. It contains the internal estimations that are used during its computational step. Examples are its estimated position, a map of the environment, positions of reference points, the content of messages it may have received, or its current planned trajectory.

▶ **Definition 3.7 (Robot).** A *robot* is a controlled hybrid dynamical system $\mathbf{r} = (O, E, Y, U, f, \varphi, g, h)$, where $O \subseteq \mathbb{R}^{d_o}$ is its *ontic state space*, $E \subseteq \mathbb{R}^{d_e}$ is its *epistemic state space*, $Y \subseteq \mathbb{R}^{d_y}$ is its *observation space*, $U \subseteq \mathbb{R}^{d_u}$ is its *control space*. $f : O \times \mathbb{R}^+ \to O$ is its *ontic dynamics*, $\varphi : E \times \mathbb{N} \to E$ is its *epistemic evolution*, $g : O \to Y$ is its *observation* function, $h : E \times \mathbb{N} \to U$ is its *control* function.
For *ontic state* $o \in O$ and *epistemic state* $e \in E$, its complete dynamics is defined as follows.

$$\dot{o}(t) = f(o(t), u[t]) \qquad \text{(ontic dynamics)} \tag{3.1}$$
$$y(t) = g(o(t)) \qquad \text{(observation)} \tag{3.2}$$
$$e[t+1] = \varphi(e[t], y[t]) \qquad \text{(epistemic evolution)} \tag{3.3}$$
$$u[t] = h(e[t], [t]) \qquad \text{(control)} \tag{3.4}$$

with control input $u \in U$ and observation $y \in Y$. The ontic dynamics and the observation use a continuous time $t$, while the epistemic evolution and the control use discrete time $[t]$, defined by the staircase function $[t] = \lfloor t \rfloor$, meaning that for $s \in (\lfloor t \rfloor, \lfloor t \rfloor + 1)$, $u(s) = h(e(\lfloor t \rfloor), \lfloor t \rfloor)$. ◀

We partition the state space of the dynamical system in Definition 2.1 following the distinction between ontic and epistemic states, the observation and control functions provide information used to steer the robot. The observation function retrieves information made available by the sensors equipped in the robot and is used to update its epistemic state. It may be of different types, such as the image provided by a camera, messages received through some communication channel, or distance measurements to the nearby obstacles. The control function translates its local epistemic state into an input for its actuators. For instance, a waypoint generated by some path planning algorithm would be converted in specific activation values for its wheel motors.

The robot definition has two distinct parts: a physical and a computational. It subsumes models in both domain, as we can retrieve both a standard robot with controller or a (analog) computer by trivializing one of the parts.

▶ **Example 3.8 (Controllers and observers).** A classical robot with a state-feedback controller $h_{\text{ctl}} \colon X \times \mathbb{R}^+ \to U$ from Definition 2.3 can be recovered from Definition 3.7 by setting the epistemic state to track the latest observation, that being the ontic

state. The robot tuple $(O, E, Y, U, f, \varphi, g, h)$ is instantiated as:

$$O = X, \quad E = Y = X, \quad g(o) = o, \quad \varphi(e, y) = y, \quad h(e, [t]) = h_{\text{ctl}}(e, [t])$$

with ontic dynamics unchanged. The epistemic evolution copies each observation into the memory it represents, i.e., $e[t+1] = y[t] = o[t]$. As such, the control $u[t] = h(e[t], [t])$ operates on the most recent ontic state, matching the state-feedback controller signature. Note that this assumes perfect observation at discrete sampling times $[t]$.

More generally, a state-feedback controller may also be implemented by maintaining an estimation of the ontic state $\widehat{o}(t)$ via the epistemic evolution[1], recovering the state observer of Definition 2.5. By setting $e[t] = \widehat{o}[t]$ with

$$e[t+1] = \varphi(e[t], u[t]), \qquad u[t] = h_{\text{ctl}}(e[t], y(t))$$

where $g(o(t))$ is an arbitrary observation function. We obtain then a controller $h$ that uses both the latest estimation $e[t]$ and the observation $y(t)$ to guide the system. ◀

We are interested in robots that are capable of interacting with each other. They do so via two types of interaction: a *perception* function $\eta \colon O \to O$ extracts physical information from the ontic state of another robot, while the *communication* function $\lambda \colon E \to E$ extracts computational information from the epistemic state of another robot.

$$\widehat{o}_{\mathbf{r},\mathbf{r}'}(t) = \eta_{\mathbf{r}}(o_{\mathbf{r}'}(t)) \qquad \text{(perception)} \tag{3.5}$$

$$\widehat{e}_{\mathbf{r},\mathbf{r}'}[t] = \lambda_{\mathbf{r}}(e_{\mathbf{r}'}[t]) \qquad \text{(communication)} \tag{3.6}$$

The distinction between perception and communication is a consequence of the distinction introduced between ontic and epistemic states. Both states may be available to the outsider observer, though through different means. For instance, a robot may be able to perceive another one's velocity or acceleration through its sensors, while the observed robot may only be capable of communicating the relative direction it is pursuing. The perception $\widehat{o}_{\mathbf{r},\mathbf{r}'}$ is the estimation of the ontic

[1]The epistemic evolution operates in discrete time, requiring either to take care of the sampling mechanism or assume a continuous estimation dynamics.

state of robot $\mathbf{r}'$ by robot $\mathbf{r}$. Similarly, The communication function generates the estimation $\widehat{e}_{\mathbf{r},\mathbf{r}'}$ of the epistemic state of robot $\mathbf{r}'$ by robot $\mathbf{r}$. Note that perception is parametrized by a continuous time variable, while communication by a discrete one, matching the nature of the states that they observe.

Communication and perception complement the *observation* step with information about the other robots. These multiple pieces of information are later processed together by the robot's epistemic evolution, matching the intuition of observations as simply inputs to a computing system.

$$y_{\mathbf{r}}(t) = g(o(t), \{\eta_{\mathbf{r}}(o_{\mathbf{r}'}(t))\}, \{\lambda_{\mathbf{r}}(e_{\mathbf{r}'}[t])\}) \quad \text{(multi-robot observation)} \tag{3.7}$$

Moreover, the introduction of multiple robots operating in the system brings forward the notions of local and global execution time. This distinction will be relevant when considering the computational perspective in Section 3.3, and one can assume global time for the remaining of this section.

▶ **Example 3.9 (Distributed averaging).** The epistemic evolution, equipped with data from a multi-robot observation, provides the ideal place for modeling distributed algorithms. Consider a robot participating in an approximate consensus task over a communication network $G$, where the ontic dynamics is trivial, i.e., $O = \{\star\}$, $f = 0$, so to isolate the algorithmic component.

The multi-robot observation implements the retrieval of data by the robot in the distributed process as follows. The perception function (Equation (3.5)) is empty, i.e., $\eta_{\mathbf{r}}(o_{\mathbf{r}'}) = \varnothing$, while the communication function (Equation (3.6)) retrieves the average estimations from the neighbors, i.e., $\lambda_{\mathbf{r}}(e_{\mathbf{r}'}) = e_{\mathbf{r}'}[t]$. Observation (Equation (3.7)) becomes then:

$$y_{\mathbf{r}}[t] = g(\star, \varnothing, \{e_{\mathbf{r}'}[t] \mid \mathbf{r}' \in \mathcal{N}_G(\mathbf{r})\})$$

The epistemic evolution implements the discrete time averaging algorithm:

$$\varphi(e_{\mathbf{r}}[t], y_{\mathbf{r}}[t]) = e_{\mathbf{r}}[t] + \alpha \sum_{\mathbf{r}' \in \mathcal{N}_G(\mathbf{r})} (e_{\mathbf{r}'}[t] - e_{\mathbf{r}}[t])$$

where $0 < \alpha < 1/d_{max}$ is a parameter ensuring convergence, for $d_{max}$ the maximum degree in $G$.

Note that the structure of the epistemic update depends on ontic dynamics' complexity. For instance, second-order dynamics require additional velocity damping terms [DI17] and resilience to malicious agents demands a filtering step known as mean subsequence-reduced [LeB+13]. Other practical variations can be seen in Kia et al. [Kia+19] The ontic and epistemic separation makes these dependencies explicit, showing which modifications address sensing (observation), computation (epistemic evolution), or actuation (control). ◀

The intuition of a distributed system of interacting mobile robots is formalized by the definition of a distributed multi-robot system as a switched hybrid dynamical system. Multiple robots obeying similar ontic dynamics and epistemic update rules are amalgamated in a single system description. A time path defines the execution order and provides with a single global execution time. The modes of the switched system correspond to the different sets of active robots at a given moment, reflecting on which robots can observe, move or compute. The description of the global system is simplified by combining the state variables and observations as follows.

▶ **Definition 3.10 (Hybrid multi-robot system).** A *multi-robot system* composed of $n$ robots is a switched hybrid system $\mathbf{S} = (\mathbf{O}, \mathbf{E}, \mathbf{Y}, \mathbf{U}, F, \Phi, G, H, \mathcal{S}, \mathbf{O}_0, \mathbf{E}_0)$, composed of a *distributed ontic state space* $\mathbf{O} \subseteq \mathbb{R}^{d_o \times n}$, a *distributed epistemic state space* $\mathbf{E} \subseteq \mathbb{R}^{d_e \times n}$, a *distributed observation space* $\mathbf{Y} \subseteq \mathbb{R}^{d_y \times n}$, a *distributed control space* $\mathbf{U} \subseteq \mathbb{R}^{d_u \times n}$, a hybrid switching *distributed ontic dynamics* $F_\rho \colon \mathbf{O} \times \mathbf{U} \times \rho \to \mathbf{O}$, a discrete *distributed epistemic evolution* $\Phi \colon \mathbf{E} \times \mathbf{Y} \to \mathbf{E}$, a multi-robot observation function $G \colon \mathbf{O} \times \mathbf{E} \to \mathbf{Y}$, a global control function $H \colon \mathbf{E} \times \mathbb{R}^+ \to \mathbf{U}$, a scheduler $\mathcal{S}$, with initial states $\mathbf{o}_0 \in \mathbf{O}_0 \subseteq \mathbf{O}$ and $\mathbf{e}_0 \in \mathbf{E}_0 \subseteq \mathbf{E}$.
The complete dynamics is parametrized by a time path $\rho \in \mathcal{S}$ as follows.

$$\dot{\mathbf{o}}(t) = F_\rho(\mathbf{o}(t), \mathbf{u}[t]) \qquad \text{(distributed ontic dynamics)} \tag{3.8}$$

$$\mathbf{y}(t) = G(\mathbf{o}(t), \eta(\mathbf{o}(t)), \lambda(\mathbf{e}[t])) \qquad \text{(multi-robot observation)} \tag{3.9}$$

$$\mathbf{e}[t+1] = \Phi(\mathbf{e}[t], \mathbf{y}(t)) \qquad \text{(distributed epistemic evolution)} \tag{3.10}$$

$$\mathbf{u}[t] = H(\mathbf{e}[t], t) \qquad \text{(global control)} \tag{3.11}$$

Where the global states relate to the local states via the time path: if robot $\mathbf{r}$ is at local time $\tau_\mathbf{r} = \rho_\mathbf{r}(t)$, then $o(\tau_\mathbf{r})$ is the contribution of $\mathbf{r}$ to the *global ontic state* $\mathbf{o}(t)$.
◀

This definition is a consequence of the evolution of individual robots, as seen in Definition 3.7. Changing variables in Equation (3.1), so to consider the states of each robot at global time $t$ for some time path $\rho$, $\mathbf{o}(t) = o(p_i(t))$, we get:

$$\dot{\mathbf{o}}(t) = \langle \dot{\rho}, f(\mathbf{o}(t), u[t]) \rangle$$

where $\langle ., . \rangle$ denotes the scalar product in $\mathbb{R}^n$. Setting $F_\rho(\mathbf{o}(t), \mathbf{u}[t]) = \langle \dot{\rho}, f(o(t), u[t]) \rangle$ gives the global ontic dynamics, Equation (3.8) of Definition 3.10.

The observation and epistemic evolution immediately generalize with $G = g$ and $\Phi = \varphi$. Note that the control must now use the local time variable as a parameter, i.e., $H(\mathbf{e}[t], t) = h(\mathbf{e}[t], [t]_\mathbf{r})$.

▶ **Remark 3.11.** We only consider homogeneous multi-robot systems, equipped with identical capacities and behavior, but note that the described model does not limit to such case. ◀

The hybrid multi-robot system, as a switched system parametrized by a scheduler, has a notion of solution, which will be used to define the runs of the system. As there are multiple possible sets of active robots, depending on the choice of time path, the mode at each instant is not uniquely determined, leading us to cast the system as a differential inclusion.

▶ **Definition 3.12 (Solutions to differential inclusions [AC84]).** Consider the general differential inclusion $\dot{x} \in \mathbb{F}(x)$ where $\mathbb{F}$ is a map from $\mathbb{R}^n$ to $2^{\mathbb{R}^n}$, the set of subsets of $\mathbb{R}^n$. A function $x(\cdot) \colon \mathbb{R}^+ \to \mathbb{R}^n$ is a *solution* to the inclusion if $x$ is an absolutely continuous function and satisfies for almost all $t \in \mathbb{R}$, $\dot{x}(t) \in \mathbb{F}(x(t))$. ◀

We can now define the trajectory of a hybrid multi-robot system as one of its possible solutions.

▶ **Definition 3.13 (Hybrid multi-robot system trajectory).** The *trajectory* of a hybrid multi-robot system $\mathbf{S} = (\mathbf{O}, \mathbf{E}, \mathbf{Y}, \mathbf{U}, F, \Phi, G, H, \mathcal{S}, \mathbf{O}_0, \mathbf{E}_0)$ under a time path $\rho \in \mathcal{S}$ is a pair $\xi = (\mathbf{o}(\cdot), \mathbf{e}[\cdot])$, where

- $\mathbf{o}(\cdot) : [0, T] \to \mathbf{O}$ is an absolutely continuous function satisfying $\dot{\mathbf{o}}(t) = F_\rho(\mathbf{o}(t), \mathbf{u}[t])$ for almost all $t \in [0, T]$, with $\mathbf{o}(0) \in \mathbf{O}_0$;
- $\mathbf{e}[\cdot] : \mathbb{N} \to \mathbf{E}$ is a discrete sequence satisfying $\mathbf{e}[t + 1] = \Phi(\mathbf{e}[t], \mathbf{y}[t])$ with $\mathbf{e}[0] \in \mathbf{E}_0$.

◀

From the definition of a system, multiple trajectories may be possible for a given initial state. We encapsulate them in a solution set.

▶ **Definition 3.14 (Solution set).** For a hybrid system $\mathbf{S}$, the *$n$-solution set* is $\Xi^n_{\mathbf{S}} = \{\xi : [0,T] \to \mathbf{O} \times \mathbf{E} \mid \xi \text{ is a trajectory under some } \rho \in \mathcal{S}, T \leq n\}$ ◀

We will see in Section 3.4 that the solution set here introduced matches the notion of system runs for the computational alternative of the multi-robot system.

## 3.3 ROBOTS, TRANSITION SYSTEMS

We complement now the continuous perspective of Section 3.2 with a discrete computational view that is mode suitable for the analysis of distributed protocols. The dynamical model captures the physical constraints inherent to a multi-robot system, and we now develop a model that focuses on the discrete decision points where computational steps occur, packaging the continuous dynamics as environmental steps. This view is compatible with the digital controllers that sample sensors and activate actuators at discrete intervals, which naturally introduce a discretization of the continuous dynamics. We use the result of this discretization as the stage of a distributed computing perspective of robotic systems, which we formalize via state machines with notions of executions and runs that are counterparts of the trajectories and solution sets seen before. This abstraction process will be made precise in Section 3.4.

We now model a distributed multi-robot system as a composition of I/O automata: robot automata handle the local epistemic computations, while an environment automaton mediates the combined ontic dynamics, scheduling of activations and possible external influences to the execution. This separation exposes the nature of robots operating beliefs based on uncertain observations, not being able to directly control the physical outcomes.

We provide an instantiation of the LOOK-COMPUTE-MOVE cycle, the standard approach to robotics in the distributed computing literature, as seen in Section 2.2, which captures the discretization by separating observation, computation and actuation in distinct phases. This change of perspective will allow us to isolate

in Chapter 4 the information available during the computational steps of a robot, the main resource needed to design and verify distributed robotic systems and protocols. We are largely inspired by the I/O automaton of Lynch [Lyn97], which we now recall.

▶ **Definition 3.15 (I/O automaton).** An *I/O automaton* $\mathcal{A} = \langle \mathbf{S}, \mathbf{I}, \mathbf{A}, \tau \rangle$ consists of:

- $\mathbf{S}$, a (possibly infinite) set of states;
- $\mathbf{I} \subseteq \mathbf{S}$, a finite set of initial states;
- $\mathbf{A}$, a signature consisting of three disjoint sets:
  - $\mathbf{A}^{in}$ of *input actions*,
  - $\mathbf{A}^{out}$ of *output actions*,
  - $\mathbf{A}^{int}$ of *internal actions*;
- $\tau \subseteq \mathbf{S} \times \mathbf{A} \times \mathbf{S}$, a transition relation where:
  - An *action* $\mathbf{a}$ is *enabled* by a state $s$ if there is a transition $(s, \mathbf{a}, s')$, denoted $\mathbf{a} \in \text{enabled}(s)$
  - Input actions $\mathbf{a} \in \mathbf{A}^{in}$ are enabled for every state $s \in \mathbf{S}$.

◀

By separating the physical (ontic) and computational (epistemic) dynamics, it becomes evident that the *epistemic evolution* matches the role of a protocol: the epistemic state is the memory being operated on. We now instantiate the protocol executed by a robot as the states of an I/O automaton.

▶ **Definition 3.16 (Robot state machine).** A *robot* $\mathbf{r}$ executes the state machine $\mathcal{A}_\mathbf{r} = \langle \mathbf{E}_\mathbf{r}, \mathbf{I}_\mathbf{r}, \mathbf{A}_\mathbf{r}, \tau_\mathbf{r} \rangle$ where

- $\mathbf{E}_\mathbf{r}$ is a possibly infinite set of *local epistemic states*;
- $\mathbf{I}_\mathbf{r} \subseteq \mathbf{E}_\mathbf{r}$ is the set of initial states;
- $\mathbf{A}_\mathbf{r} = \mathbf{A}_\mathbf{r}^{obs} \sqcup \mathbf{A}_\mathbf{r}^{ctl} \sqcup \mathbf{A}_\mathbf{r}^{epi}$ is the set actions, divided in three types, each containing at least the empty action $\perp$:
  - $\mathbf{A}_\mathbf{r}^{obs}$ are input *observation actions*, to be received from the environment,
  - $\mathbf{A}_\mathbf{r}^{ctl}$ are output *control actions*, to be sent to the environment,
  - $\mathbf{A}_\mathbf{r}^{epi}$ are internal *epistemic actions*, internal to the robot;
- $\tau_\mathbf{r} \subseteq \mathbf{E}_\mathbf{r} \times \mathbf{A}_\mathbf{r} \times \mathbf{E}_\mathbf{r}$ is a *transition relation* that associates to each pair of epistemic state and action another epistemic state.

◀

The epistemic state $\mathbf{E_r}$ corresponds to the memory of the machine. It contains all of the information that is available to a robot in order to operate. Importantly, it contains no physical information, but only processed information. For instance, it may contain its current and past estimated positions (instead of its actual position), the last observation it received and its current control decision.

Three types of actions may be performed, which always lead to an update in the epistemic state of the robot: observations, control and epistemic. An observation action is external to the robot, it is received from the environment. An example is the usage of a camera, which demands visual information that will be mediated by the environment. A control action is generated by the robot and sent to the environment. Those are the commands given to physical machine, such as to move towards some position or decide a value. Again, those are mediated by the environment that may faithfully execute the command or apply some (possibly non-deterministic) influence. The epistemic actions are internal to the robot, they represent computations and decisions that are performed using the current epistemic state.

▶ **Example 3.17 (Robot with limited visibility).** Consider a system of robots operating in a graph[2] $G_X = (V, E)$, where robots are positioned in vertices $v \in V$ and move between them when there exists an edge $(v, v') \in E$. Robots operate under limited visibility, capable of observing vertices at most some distance $r > 0$, inducing a communication graph $Gr = (V, \{(v, v') \mid v, v' \in V, \mathrm{dist}_{G_X}(v, v') \leqslant r\})$. We show now how visibility constraints are encoded in a robot automaton $\mathcal{A}_\mathbf{r} = \langle \mathbf{E_r}, \mathbf{I_r}, \mathbf{A_r}, \tau_\mathbf{r} \rangle$, as per Definition 3.16.

Each local epistemic state $e_\mathbf{r} \in \mathbf{E_r}$ tracks a set of observed robots $\mathcal{N}_{Gr}(\mathbf{r})$ and their positions, i.e., $\{(\mathbf{r}', v_{\mathbf{r}'}) \mid \mathbf{r}' \in \mathcal{N}(\mathbf{r})\}$, including itself. Its actions available are:

- $\mathbf{A}_\mathbf{r}^{obs} = \{\mathrm{observe}(A) \mid A \subseteq \mathcal{R} \times V, \forall (\mathbf{r}', v_{\mathbf{r}'}) \in A : (v_\mathbf{r}, v_{\mathbf{r}'}) \in Gr\}$ for receiving position data of robots within the visibility radius,
- $\mathbf{A}_\mathbf{r}^{ctl} = \{\mathrm{move}(v') \mid v' \in V, (v_\mathbf{r}, v') \in E\}$ for sending control commands to move to adjacent vertices,

[2] We use a graph as space for simplicity, but an arbitrary subspace of $\mathbb{R}^n$ could be used with a suitable discretization.

- $\mathbf{A}_{\mathbf{r}}^{epi}$ for internal computational steps, such as updating position estimates and local planning.

The transition $\tau_{\mathbf{r}} \subseteq \mathbf{E}_{\mathbf{r}} \times \mathbf{A}_{\mathbf{r}} \times \mathbf{E}_{\mathbf{r}}$ updates the set of neighbors $\mathcal{N}(\mathbf{r})$ based on received observations and executed controls. Note that the epistemic state contains no factual physical information, only processed position data. When two robots $\mathbf{r}, \mathbf{r}'$ are at distance greater than the visibility radius, neither appears in each other's observation actions. This limitation will be shown in Chapter 4 to be a source of uncertainty in the system, increasing the sets of possible states that a robot cannot distinguish. ◀

▶ **Remark 3.18 (Decision outputs).** Note that the decisions taken with respect to a distributed task are implemented as internal epistemic actions. This implementation is separate from the protocol execution, as it is commonly done with I/O automata [Lyn97]. ◀

The class of robots that we are interested in have no bounds in their memories, and preserve all information across time steps. This is formalized as the perfect recall property, which will be important in Chapter 4 to define the knowledge of a system of robot.

▶ **Assumption 3.19 (Perfect recall).** A robot $\mathcal{A}_{\mathbf{r}} = \langle \mathbf{E}_{\mathbf{r}}, \mathbf{I}_{\mathbf{r}}, \mathbf{A}_{\mathbf{r}}, \tau_{\mathbf{r}} \rangle$ has *perfect recall* if its epistemic states $\mathbf{e} \in \mathbf{E}$ are finite sequences of all past epistemic states, and the transition relation strictly extends that sequence, i.e., if $(\mathbf{e}, \mathbf{a}, \mathbf{e}') \in \tau_{\mathbf{r}}$, then $\mathbf{e}$ is a *proper prefix* of $\mathbf{e}'$, denoted $\mathbf{e} \subsetneq \mathbf{e}'$. Two epistemic states are equal if and only if they are identical sequences. ◀

▶ **Remark 3.20 (Perfect recall creates different states).** Observe that, for a robot with perfect recall, every pair of states $\mathbf{e}, \mathbf{e}' \in \mathbf{E}$, such that $(\mathbf{e}, \mathbf{a}, \mathbf{e}') \in \tau_{\mathbf{r}}$, is necessarily different, i.e., $\mathbf{e} \neq \mathbf{e}'$. As the transition relation of such a system appends a new element to the sequence forming its local epistemic state, it emulates a local clock by counting the length of such sequence. ◀

As we apply the LOOK-COMPUTE-MOVE cycle, the control actions implicitly request subsequent observation actions, simplifying the protocol. The environment,

which mediates all physical and inter-robot interaction, determines what observations result from each control action. We now define this environment automaton, which serves as the single source of truth for all *ontic states* and constraints.

▶ **Definition 3.21 (Environment state machine).** For a set of robots $\mathcal{R}$, the *environment* is a state machine $\mathcal{E} = \langle \mathbf{O}_\mathcal{E}, \mathbf{I}_\mathcal{E}, \mathbf{A}_\mathcal{E}, \tau_\mathcal{E} \rangle$ where:

- $\mathbf{O}_\mathcal{E}$ is a set of possible *environment states*.
- $\mathbf{I}_\mathcal{E} \subseteq \mathbf{O}_\mathcal{E}$ is a set of initial states;
- $\mathbf{A}_\mathcal{E} = \mathbf{A}_\mathcal{E}^{ctl} \sqcup \mathbf{A}_\mathcal{E}^{obs} \sqcup \mathbf{A}_\mathcal{E}^{onc}$ is a set of three types of actions, each containing at least the empty action $\perp$:
  - $\mathbf{A}_\mathcal{E}^{ctl} = \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{A}_\mathbf{r}^{ctl}$ is a set of input control actions,
  - $\mathbf{A}_\mathcal{E}^{obs} = \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{A}_\mathbf{r}^{obs}$ is a set of output observation actions,
  - $\mathbf{A}_\mathcal{E}^{onc}$ is a set of internal *ontic actions*;
- $\tau_\mathcal{E} \subseteq \mathbf{O}_\mathcal{E} \times \mathbf{A}_\mathcal{E} \times \mathbf{O}_\mathcal{E}$ is a transition relation, where an environment state and action are associated to another environment state.

◀

The environment interacts with the robots via its input and output actions, control commands and observations, respectively. Its role extends beyond physical dynamics, though. It can model, through its internal actions, communication failures (dropping messages), sensor limitations (restricting observations), actuator faults (rejecting control commands). Most importantly, it models the scheduler, determining the activation order. For instance, limited visibility will make observations only contain information within some geometric constraint, while message loss becomes an environment choice to deliver an empty message $\perp$, instead of the actual one. As an I/O automaton, it evolves its state at each action, and as such encapsulate all the *ontic dynamics* of the robots. The time of the environment is the one global time, while those of the robots are only local ones.

The I/O automata above can be composed as product automata, as shown in [Lyn97], whenever they respect some basic conditions making their activations not overlap in inconvenient manners, and when every input action (respectively output) is matched by some output action (respectively input). We recall now its definition.

▶ **Definition 3.22 (Automata composition).** The *composition* of the state machines of robots $\mathbf{r}_i \in \mathcal{R}$ with the environment $\mathcal{E}$, denoted $\mathbf{r}_1||\mathbf{r}_2||\ldots||\mathbf{r}_n||\mathcal{E}$, is the I/O automaton $\mathbf{S}_{\mathcal{R},\mathcal{E}} = \langle \mathbf{S}, \mathbf{I}, \mathbf{A}, \tau \rangle$, where:

- $\mathbf{S} = \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{E}_\mathbf{r} \times \mathbf{O}_\mathcal{E}$
- $\mathbf{I} = \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{I}_\mathbf{r} \times \mathbf{I}_\mathcal{E}$
- $\mathbf{A} = \bigcup_{\mathbf{r}\in\mathcal{R}} \mathbf{A}_\mathbf{r} \cup \mathbf{A}_\mathcal{E}$ where the components are compatible, i.e., $\mathbf{A}_\mathcal{E}^{obs} = \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{A}_\mathbf{r}^{obs}$ and $\mathbf{A}_\mathcal{E}^{ctl} = \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{A}_\mathbf{r}^{ctl}$
- $\tau \subseteq \mathbf{S} \times \mathbf{A} \times \mathbf{S}$, where a transition $(s, \mathbf{a}, s') \in \tau$ if the following holds for every component $c \in \{\mathcal{E}\} \cup \{\mathcal{A}_\mathbf{r}\}_{\mathbf{r}\in\mathcal{R}}$:
  - If $\mathbf{a} \in \mathbf{A}_c$, then $(\pi_c(s), \mathbf{a}, \pi_c(s')) \in \tau_c$;
  - If $\mathbf{a} \notin \mathbf{A}_c$, then $\pi_c(s') = \pi_c(s)$.

◀

▶ **Remark 3.23 (Synchronization).** Note that the automata composition enforces that shared actions are executed synchronously for all components. This corresponds to matching observation actions from the environment to the robot that are executed simultaneously, and similarly for control actions from the robots to the environment. Furthermore, by Definition 3.15, input actions are always enabled, guaranteeing that both robots process observations and the environment receives controls. ◀

We can now define the *compositional multi-robot system* as an I/O automaton $\mathbf{S}_{\mathcal{R},\mathcal{E}}$, obtained by the composition of multiple robots automata $\mathcal{R}$ and one environment automaton $\mathcal{E}$. Through the composition of robots and environment, we can enforce that all interaction between the robots happens through the environment. This captures the reality of distributed robotic systems, where the physical channels are subject to interference. Most importantly, it ensures that the results and tools from distributed computing transfer to our model.

Unsurprisingly, a pure distributed computing model arises as a special case of the automata composition. When the environment states are possible message buffers and its ontic actions are message delivery or loss, we recover the standard message-passing system. The robots' local epistemic states track the messages that were sent and received, while the environment implements the message adversary. As such, our framework extends the classical distributed computing models.

▶ **Example 3.24 (Message-passing system).** Recall from Definition 2.12 that a message-passing system consists of $n$ processes $\Pi = \{p_1, \ldots, p_n\}$ in a communication network $G$. Each process $p_i$ can send a message $m \in \mathcal{M}$, for $\mathcal{M}$ finite, via $\text{send}_{i,j}(m)$, which places it in the respective channel, and receive via $\text{receive}_{j,i}$, retrieving either a message or $\perp$ from the channel. We show that this model from distributed computing fits naturally within our framework, as the message-passing model is captured in the robot and environment automata, while physical robot dynamics could be added to the same structure.

Each process becomes a robot automaton $\mathbf{r} = \langle \mathbf{E}_\mathbf{r}, \mathbf{I}_\mathbf{r}, \mathbf{A}_\mathbf{r}, \tau_\mathbf{r} \rangle$ as per Definition 3.16. The local epistemic state $\mathbf{E}_\mathbf{r}$ tracks message history (sent, received and local computation state). The robot has the following actions:

- $\mathbf{A}_\mathbf{r}^{obs} = \{\text{receive}_{j,i}(m) \mid j \in \mathcal{R}, m \in \mathcal{M} \cup \{\perp\}\}$ for receiving messages,
- $\mathbf{A}_\mathbf{r}^{ctl} = \{\text{send}_{i,j}(m) \mid j \in \mathcal{R}, m \in \mathcal{M}\}$ for sending messages,
- $\mathbf{A}_\mathbf{r}^{epi}$ for local computation steps.

The transition relation $\tau_\mathbf{r} \subseteq \mathbf{E}_\mathbf{r} \times \mathbf{A}_\mathbf{r} \times \mathbf{E}_\mathbf{r}$ updates the message history accordingly. Note that the epistemic state contains no physical information, only processed data.

The environment automaton $\mathcal{E} = \langle \mathbf{O}_\mathcal{E}, \mathbf{I}_\mathcal{E}, \mathbf{A}_\mathcal{E}, \tau_\mathcal{E} \rangle$, as per Definition 3.21, implements the communication channels[3] with message buffers and a message adversary. States $\mathbf{O}_\mathcal{E} = (\mathcal{M} \cup \{\perp\})^{n \times n}$ represent the content of the message channels, with actions $\mathbf{A}_\mathcal{E}^{obs} = \prod_{\mathbf{r} \in \mathcal{R}} \mathbf{A}_\mathbf{r}^{ctl}$ and $\mathbf{A}_\mathcal{E}^{ctl} = \prod_{\mathbf{r} \in \mathcal{R}} \mathbf{A}_\mathbf{r}^{obs}$, matching those of the robots. The ontic actions

$$\mathbf{A}_\mathcal{E}^{onc} = \{\text{deliver}_{ij}, \text{drop}_{ij}, \text{delay}_{ij} \mid i, j \in \mathcal{R}\}$$

model the adversary, e.g., reliable channels only deliver, lossy ones may drop. The transition relation $\tau_\mathcal{E} \subseteq \mathbf{O}_\mathcal{E} \times \mathbf{A}_\mathcal{E} \times \mathbf{O}_\mathcal{E}$ places messages in buffers on $\text{send}_{i,j}(m)$ and retrieves them on $\text{receive}_{j,i}$, with ontic actions determining delivery behavior.

The compositional system $\mathbf{S}_{\mathcal{R},\mathcal{E}} = \mathbf{r}_1 \| \cdots \| \mathbf{r}_n \| \mathcal{E}$ is then obtained by Definition 3.22. The scheduler encodes the activation order of the robots by a time path $\rho$: at each global time $t$, the set $\text{active}_\rho(t)$ determines which robots perform

[3] In [Lyn97] it is suggested to implement each channel as one I/O automaton, which is equally valid as those could be composed to provide the current definition.

transitions, while inactive ones remain unchanged (more precisely defined in Definition 3.25). As such, message-passing models from distributed computing are special cases of multi-robot systems. In this instance, the environment merely manages message buffers as data structures, but the same framework accommodates arbitrary physical robot dynamics alongside the communication protocol. ◀

We look now at the execution structure of a compositional system $\mathbf{S}_{\mathcal{R},\mathcal{E}}$. First, one should note that at a given global time $t$, a local state can refer to either the local epistemic states of the composing robots $\mathbf{r} \in \mathcal{R}$ or the ontic state of the environment $\mathcal{E}$. The *global state* is then a tuple $\mathbf{c} = \langle t, o, \overrightarrow{e} \rangle$, where $\overrightarrow{e}$ is a vector of one local epistemic state per robot. The global states $\mathbf{C}$ are snapshots of the system, which we now use to define the big-step *operational semantics*, where each transition advances one complete look-compute-move cycle.

▶ **Definition 3.25 (Operational Semantics).** For a system $\mathbf{S}_{\mathcal{R},\mathcal{E}}$ with a time path $\rho$, a transition between two global states, denoted $\langle t, o, \vec{e} \rangle \Rightarrow \langle t', o', \vec{e}' \rangle$ holds if and only if

- $\exists \vec{y} \in \mathbf{A}_{\mathcal{E}}^{obs}$ such that $(o, \vec{y}, o^*) \in \tau_{\mathcal{E}}$ for some $o^*$ **(environment observation)**
- $\forall \mathbf{r} \in \text{active}_\rho(t), \exists e_{\mathbf{r}}^*, e_{\mathbf{r}}^{**}$, such that:
  - $(e_{\mathbf{r}}, y_{\mathbf{r}}, e_{\mathbf{r}}^*) \in \tau_{\mathbf{r}}$ **(robot observation)**
  - $\exists \mathbf{a} \in \mathbf{A}_{\mathbf{r}}^{epi}, (e_{\mathbf{r}}^*, \mathbf{a}, e_{\mathbf{r}}^{**}) \in \tau_{\mathbf{r}}$ **(epistemic transition)**
  - $\exists u_{\mathbf{r}}, (e_{\mathbf{r}}^{**}, u_{\mathbf{r}}, e_{\mathbf{r}}^{'}) \in \tau_{\mathbf{r}}$ **(robot control)**
  - $[t']_{\mathbf{r}} = [t]_{\mathbf{r}} + 1$ **(local time step)**
- $\forall \mathbf{r} \notin \text{active}_\rho(t)$, such that:
  - $e_{\mathbf{r}}^{'} = e_{\mathbf{r}}$ **(no inactive computation)**
  - $[t']_{\mathbf{r}} = [t]_{\mathbf{r}}$ **(no inactive local time step)**
- $(o^*, \vec{u}, o^{'}) \in \tau_{\mathcal{E}}$ **(environment control)**
- $t' = min\{s > t \text{ such that } \rho(s) \neq \rho(t)\}$ **(global time step)**

where $y_{\mathbf{r}}$ and $u_{\mathbf{r}}$ are robot $\mathbf{r}$'s component of $\vec{y}$ and $\vec{u}$, respectively, and $[t]_{\mathbf{r}} = \lfloor \rho_{\mathbf{r}} \rfloor$ is $\mathbf{r}$'s local time. ◀

Given the transition rule above, we can define now an execution: a chain of transitions of global states that forms a single history of the system. We use the

fact that the global and local times associated to a time path correspond precisely to the clock of the environment and of the robots, respectively.

▶ **Definition 3.26 (System execution).** An *execution* of a system under time path $\rho$ is a possibly infinite sequence $\sigma = (\mathbf{c}_i)_{i\in\mathbb{N}}$ where:

- $\mathbf{c}_0 = \langle 0, o, \vec{e}\rangle$ is an initial state with $o_{\mathcal{E}} \in \mathbf{I}_{\mathcal{E}} \subseteq \mathbf{O}_{\mathcal{E}}$ and $e_{\mathbf{r}} \in \mathbf{I}_{\mathbf{r}} \subseteq \mathbf{E}_{\mathbf{r}}$ for all $\mathbf{r} \in \mathcal{R}$;
- Each $\mathbf{c}_i \Rightarrow \mathbf{c}_{i+1}$ is a valid transition by Definition 3.25;

◀

While an execution is a trace according to one time path, an environment is usually allowed to follow a more general scheduler definition, which contains a class of possible time paths, as seen in Section 3.1. A system run corresponds to this generalization, containing all possible histories associated to a compositional system.

▶ **Definition 3.27 (System run).** For compositional system $\mathbf{S}$ and $n \in \mathbb{N}$, its *$n$-run* is $\Sigma^n_{\mathbf{S}} = \{\sigma = (\mathbf{c}_i)_{i\leq k} \mid \mathbf{c}_0 \in \mathbf{I}_{\mathbf{S}}, \mathbf{c}_i \Rightarrow \mathbf{c}_{i+1}, k \leq n\}$. ◀

The run captures all possible transitions of up to $n$ steps from any possible initial state of the compositional multi-robot system in Definition 3.22, any sequence of transitions allowed by the transition rule in Definition 3.25, and any non-deterministic choice within the environment and robot transition relations. We will see in Chapter 5 that an inclusion relation on runs can be defined from the inclusion of different definitions of the system, such as the class of allowed time paths, the initial states, the set of robots, among others. We denote by $\Sigma_{\mathbf{S}}$ the maximal run of system $\mathbf{S}$.

## 3.4 CORRESPONDENCE BETWEEN THE TWO VIEWS

We have presented models for distributed multi-robot system: a switched hybrid dynamical system that naturally captures the physical dynamics (Section 3.2), and a compositional state machine that focuses on computations and information flow (Section 3.3). The two perspectives share a common behavior: a robot makes decisions based entirely on their epistemic states, as it is the only information available to their computational process. Both models capture the same epistemic

evolution, either through epistemic jumps amidst the continuous evolution of the hybrid system, or by generating discrete global states as a compositional system. We formalize now under which conditions the computational model is a faithful abstraction of the hybrid system.

We establish now a system abstraction that maps between state spaces of both models, preserving their structure while connecting continuous dynamics to discrete computations, keeping the same epistemic content. As shown below, this system abstraction induces a simulation relation between global states of both models, which will serve to show that task solvability analysis and protocol design can be done meaningfully in both, justifying the usage of the simpler compositional model in the subsequent chapters.

▶ **Definition 3.28 (System abstraction).** Let $\mathbf{S}_H = (\mathbf{O}, \mathbf{E}, \mathbf{Y}, \mathbf{U}, F, \Phi, G, H, \mathcal{S}, \mathbf{o}_0, \mathbf{e}_0)$ be a multi-robot system and $\mathbf{S}_C = \langle \mathbf{S}, \mathbf{I}, \mathbf{A}, \tau \rangle$ be a compositional multi-robot system. A *system abstraction* $\Psi : \mathbf{S}_H \to \mathbf{S}_C$ is a tuple of surjective functions $\Psi = (\Psi_{\mathbf{O}} : \mathbf{O} \to \mathbf{O}_{\mathcal{E}}, \Psi_{\mathbf{E}} : \mathbf{E} \to \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{E}_{\mathbf{r}}, \Psi_{\mathbf{Y}} : \mathrm{codom}(G) \to \mathbf{A}^{obs}_{\mathcal{E}}, \Psi_{\mathbf{U}} : \mathrm{codom}(H) \to \mathbf{A}^{ctl}_{\mathcal{E}})$, respecting the following compatibility conditions for any time path $\rho \in \mathcal{S}$:

- **(initial states)** For any $\mathbf{o}_0 \in \mathbf{O}_0$ and $\mathbf{e}_0 \in \mathbf{E}_0$, $\Psi_{\mathbf{O}}(\mathbf{o}_0) \subseteq \mathbf{I}_{\mathcal{E}}$ and $\Psi_{\mathbf{E}}(\mathbf{e}_0) \subseteq \prod_{\mathbf{r}\in\mathcal{R}} \mathbf{I}_{\mathbf{r}}$.
- **(local epistemic states)**. For any $\mathbf{e}, \mathbf{e}' \in \mathbf{E}$ and robot $\mathbf{r} \in \mathcal{R}$, if $\pi_{\mathbf{r}}(\mathbf{e}) = \pi_{\mathbf{r}}(\mathbf{e}')$, then $\pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e})) = \pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e}'))$.
- **(epistemic evolution).** For any $\mathbf{e} \in \mathbf{E}$ and $\mathbf{y} \in \mathbf{Y}$, if $\mathbf{e}' = \Phi(\mathbf{e}, \mathbf{y})$, then for each $\mathbf{r} \in \mathcal{R}$, where $e_{\mathbf{r}} = \pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e}))$, $y_{\mathbf{r}} = \pi_{\mathbf{r}}(\Psi_{\mathbf{Y}}(\mathbf{y}))$, and $e'_{\mathbf{r}} = \pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e}'))$, there exist $e^*_{\mathbf{r}}, e^{**}_{\mathbf{r}} \in \mathbf{E}_{\mathbf{r}}$, $\mathbf{a}_{\mathbf{r}} \in \mathbf{A}^{epi}_{\mathbf{r}}$, and $u_{\mathbf{r}} \in \mathbf{A}^{ctl}_{\mathbf{r}}$ such that $(e_{\mathbf{r}}, y_{\mathbf{r}}, e^*_{\mathbf{r}}), (e^*_{\mathbf{r}}, \mathbf{a}_{\mathbf{r}}, e^{**}_{\mathbf{r}}), (e^{**}_{\mathbf{r}}, u_{\mathbf{r}}, e'_{\mathbf{r}}) \in \tau_{\mathbf{r}}$.
- **(control generation).** For any $\mathbf{e} \in \mathbf{E}$, the control action $\Psi_{\mathbf{U}}(H(\mathbf{e})) \in \mathrm{enabled}(\Psi_{\mathbf{E}}(\mathbf{e}))$.
- **(ontic-environment evolution).** Let $o \in \mathbf{o}$ and $u \in \mathrm{codom}(H)$. Let $t < t'$ be two consecutive times when epistemic updates occur, i.e., $\exists \mathbf{r} \lfloor \rho_{\mathbf{r}}(t') \rfloor > \lfloor \rho_{\mathbf{r}}(t) \rfloor$. If $o'$ is the ontic state reached after integrating $\dot{\mathbf{o}} = F_\rho(\mathbf{o}, u)$ from $o$ over $[t, t']$, then $(\Psi_{\mathbf{O}}(o), \Psi_{\mathbf{U}}(u), \Psi_{\mathbf{O}}(o')) \in \tau_{\mathcal{E}}$.

◀

Note that the epistemic evolution condition ensures the correspondence between the hybrid and the compositional epistemic states, the observations and observation actions, as well as that all transitions are valid in the compositional system. The control generation condition ensures that control commands and control actions match, as well as their generation. The ontic-environment evolution conditions guarantee that the global ontic states match the environment state, while ensuring that all transitions of the switched system are valid after the abstraction, as it takes into consideration the time path that parametrizes the jumps between sets of active robots. As such, the system abstraction induces a simulation relation between the global states of a hybrid system and those of a compositional system.

▶ **Remark 3.29 (Epistemic updates in global time).** Note that the rectification property of time paths ensures that the global time advances only when at least one local robot time advances. Hence, the set of active robots $\mathrm{active}_\rho(t)$ is never empty, and every global state transition involves at least one epistemic update. ◀

From the system abstraction, we can formalize the relationship between executions in both systems. We use a simulation relation that connects global states at discrete points when the epistemic updates happen.

▶ **Definition 3.30 (State simulation relation).** Let $(\mathbf{o}_H, \mathbf{e}_H) \in \mathbf{O} \times \mathbf{E}$ be a global state of a hybrid system $\mathbf{S}_H$ at time $t_H \in \mathbb{R}_+$, and let $\langle t_C, o_C, \vec{e_C} \rangle \in \mathbf{C}$ be a global state of a compositional system. A *simulation relation* on global states is a set $\leqslant \subseteq (\mathbf{O} \times \mathbf{E} \times \mathbb{R}_+) \times \mathbf{C}$ such that

$$(\mathbf{o}_H, \mathbf{e}_H, t_H) \leqslant \langle t_C, o_C, \vec{e_C} \rangle \iff t_H = t_C \text{ and } \Psi_{\mathbf{O}}(\mathbf{o}_H) = o_C \text{ and } \Psi_{\mathbf{E}}(\mathbf{e}_H) = \vec{e_C}$$

◀

We note that the simulation relation is preserved under the system evolution of abstracted systems.

▶ **Lemma 3.31 (Simulations are stable under system evolution).** Let $\mathbf{S}_H$ be a hybrid system and $\mathbf{S}_C$ be a compositional system, such that there exists a compatible system

abstraction $\Psi : \mathbf{S}_H \to \mathbf{S}_C$. For any trajectory $\xi = (\mathbf{o}(.), \mathbf{e}[.]) \in \Xi_{\mathbf{S}_H}$, and for any pair of consecutive epistemic update times $t_i, t_{i+1}$ in the update sequence $T_\xi$, if a compositional state $\mathbf{c}_i \in \mathbf{C}$ satisfies $(\mathbf{o}(t_i), \mathbf{e}[t_i], t_i) \leqslant \mathbf{c}_i$, then there is a compositional state $\mathbf{c}_{i+1} \in \mathbf{C}$ such that the transition $\mathbf{c}_i \Rightarrow \mathbf{c}_{i+1}$ is valid in $\mathbf{S}_C$ and the simulation holds, i.e., $(\mathbf{o}(t_{i+1}), \mathbf{e}[t_{i+1}], t_{i+1}) \leqslant \mathbf{c}_{i+1}$. ◀

*Proof.* Let $(\mathbf{o}(t_i), \mathbf{e}[t_i], t_i) \leqslant \mathbf{c}_i$, where $\mathbf{c}_i = \langle t_i, o_i, \vec{e}_i \rangle$, and $t_i, t_{i+1}$ be two consecutive epistemic update times.

We construct the new state $\mathbf{c}_{i+1}$ by applying the system abstraction at $(\mathbf{o}(t_{i+1}), \mathbf{e}[t_{i+1}], t_{i+1})$. We obtain $\mathbf{c}_{i+1} = \langle t_{i+1}, \Psi_{\mathbf{O}}(\mathbf{o}(t_{i+1})), \Psi_{\mathbf{E}}(\mathbf{e}[t_{i+1}]) \rangle$, which satisfies $(\mathbf{o}(t_{i+1}), \mathbf{e}[t_{i+1}], t_{i+1}) \leqslant \mathbf{c}_{i+1}$ by Definition 3.30.

For $\mathbf{S}_C$, let $\mathcal{R}$ be its set of robots and $\mathcal{E}$ be its environment. We verify now that the transition $\mathbf{c}_i \Rightarrow \mathbf{c}_{i+1}$ is valid according to Definition 3.25 using the compatibility conditions of the system abstraction in Definition 3.28:

- **(environment observation).** At time $t_i$, $\mathbf{S}_H$ generates observation $\mathbf{y}(t_i) = G(\mathbf{o}(t_i), \eta(\mathbf{o}[t_i]), \lambda(\mathbf{e}[t_i]))$, let $y = \Psi_{\mathbf{Y}}(\mathbf{y}(t_i))$ be the corresponding action $y \in \mathbf{A}_{\mathcal{E}}^{obs}$, where, by the compatibility condition of the abstraction $\Psi$, there exists $o^*$ such that $(o_i, y, o^*) \in \tau_{\mathcal{E}}$. This step has no ontic update and $o^* = o_i$.

- **(robot observation and epistemic transition).** Let $\vec{e} = \Psi_{\mathbf{E}}(\mathbf{e}[t_i])$. For each active robot $\mathbf{r} \in \text{active}_\rho(t_i)$, let $e_{\mathbf{r}} = \pi_{\mathbf{r}}(\vec{e})$, $y_{\mathcal{E}} = \Psi_{\mathbf{Y}}(\mathbf{y}(t_i))$. Per Definition 3.22, each observation action $y_{\mathcal{E}} \in \mathbf{A}_{\mathcal{E}}^{obs}$ has a matching $y_{\mathbf{r}} \in \mathbf{A}_{\mathbf{r}}^{obs}$. By the compatibility of the epistemic evolution, there exists $(e_{\mathbf{r}}, y_{\mathbf{r}}, e_{\mathbf{r}}^*) \in \tau_{\mathbf{r}}$. Similarly, for $\mathbf{e}[t_{i+1}] = \Phi(\mathbf{e}[t_i], \mathbf{y}(t_i))$, it is ensured that $(e_{\mathbf{r}}^*, \mathbf{a}_{\mathbf{r}}, e_{\mathbf{r}}^{**}) \in \tau_{\mathbf{r}}$, where $e_{\mathbf{r}}^{**} = \pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e}[t_{i+1}]))$.

- **(robot control).** At time $t_i$, $\mathbf{S}_H$ generates control $\mathbf{u}[t_{i+1}] = H(\mathbf{e}[t_i])$. Let $u_{\mathbf{r}} = \pi_{\mathbf{r}}(\Psi_{\mathbf{U}}(\mathbf{u}[t_{i+1}]))$, which is enabled in $\Psi_{\mathbf{E}}(e^{**})$ by the control generation compatibility condition. Hence, there exists $(e_{\mathbf{r}}^{**}, u, e_{\mathbf{r}}^{'}) \in \tau_{\mathbf{r}}$, and we obtain $\vec{e}_{i+1} = (e_{\mathbf{r}}^{'})_{\mathbf{r} \in \mathcal{R}}$.

- **(local time step).** For active robots $\mathbf{r} \in \text{active}_\rho(t_i)$, time advances at the epistemic computation and $[t']_{\mathbf{r}} = [t]_{\mathbf{r}} + 1$. For inactive robots $\mathbf{r} \notin \text{active}_\rho(t_i)$, local time stays the same, i.e., $[t']_{\mathbf{r}} = [t]_{\mathbf{r}}$.

- **(no inactive computation).** For robot $\mathbf{r} \notin \text{active}_\rho(t_i)$, $\mathbf{e}[t_{i+1}] = \mathbf{e}[t_i]$. By the local epistemic state compatibility, $\pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e}[t_{i+1}])) = \pi_{\mathbf{r}}(\Psi_{\mathbf{E}}(\mathbf{e}[t_i]))$.
- **(environment control).** The system $\mathbf{S}_H$ arrives at the ontic state $\mathbf{o}(t_{i+1})$ from $\mathbf{o}(t_i)$ by integrating over $t_i, t_{i+1}$ using control $\mathbf{u}[t_i] = H(\mathbf{e}[t_i])$. The ontic-environment compatibility condition guarantees that we have $(\Psi_{\mathbf{O}}(\mathbf{o}(t_i)), \Psi_{\mathbf{U}}(\mathbf{u}[t_{i+1}]), \Psi_{\mathbf{O}}(\mathbf{o}(t_{i+1}))) \in \tau_\mathcal{E}$, where $o_{i+1} = \Psi_{\mathbf{O}}(\mathbf{o}(t_{i+1}))$
- **(global time step).** By assumption, the global time advances from $t_i$ to $t_{i+1}$, the next epistemic update.

■

Note that the proof of Lemma 3.31 make evident the delay between the epistemic and ontic updates of the system, as the robot first processes an observation and computes a new control, updating its epistemic state, before any change in the environment, and its ontic state is done. All of those steps are mediated by the several functions defining the multi-robot system, hybrid or compositional. We can state now how the simulation defined on global states can be lifted to relate trajectories to runs.

▶ **Definition 3.32 (Execution simulation).** Let $\xi = (\mathbf{o}(.), \mathbf{e}[.]) \in \Xi_{\mathbf{S}_H}$ be a trajectory of a hybrid system $\mathbf{S}_H$, and let $\sigma = (\mathbf{c}_0, \mathbf{c}_1, \dots) \in \Sigma_{\mathbf{S}_C}$ be a run of a compositional system $\mathbf{S}_C$. $\xi$ simulates $\sigma$, denoted $\xi \preceq \sigma$, if and only if the state-wise simulation $\leqslant$ holds for every time $t_i$ in the sequence of epistemic transitions $T_\xi$. ◀

We can now state the first theorem: whenever a compositional system is a compatible abstraction of a hybrid system, the executions of the former precisely match the trajectories of the latter.

▶ **Theorem 3.33 (Existence and uniqueness of abstracted trajectories).** Let $\mathbf{S}_H$ be a hybrid system and $\mathbf{S}_C$ be a compositional system. Let $\Psi : \mathbf{S}_H \to \mathbf{S}_C$ be a compatible abstraction between them. For every trajectory $\xi \in \Xi_{\mathbf{S}_H}$, there exists a unique execution $\sigma \in \Sigma_{\mathbf{S}_C}$ such that the simulation $\xi \preceq \sigma$ holds.

*Proof.* We proceed by induction on the sequence of epistemic transitions $T_\xi = (t_0, t_1, \dots)$. The base case $t_0$ holds by the compatibility condition of initial states in Definition 3.28. Lemma 3.31 provides that, if a simulation holds at time $t_i$, then a

valid transition exists for the corresponding state $\mathbf{c}_{i+1}$, for which the simulation holds at $t_{i+1}$.

Uniqueness is obtained from the abstraction $\Psi$ being composed of functions. For any trajectory $\xi$ the application of the abstraction at each time $t \in T_\xi$ produces a unique global state. As every state $\mathbf{c}_i$ of the execution is constructed deterministically, the resulting execution is unique. ■

Theorem 3.33 allows us to lift the system abstraction to a simulation map, from a solution set to a system run.

▶ **Definition 3.34 (Simulation map).** Let $\Psi : \mathbf{S}_H \to \mathbf{S}_C$ be an abstraction from a hybrid system $\mathbf{S}_H$ to a compositional system $\mathbf{S}_C$. The map $\Psi$ can be lifted to a *simulation map* $\Psi^* : \Xi_{\mathbf{S}_H} \to \Sigma_{\mathbf{S}_C}$, which maps each trajectory $\xi \in \Xi_{\mathbf{S}_H}$ to the unique execution $\sigma \in \Sigma_{\mathbf{S}_C}$ that simulates it, i.e., $\xi \preceq \sigma$. ◀

We have shown that the hybrid multi-robot system can be simulated into a compositional system counterpart such that the content of their trajectories is preserved. When considering the inverse case, we note that not all robotic computational systems should be expected to have a dynamical version, as one may impose an unfeasible behavior violating its physical constraints. Hence, we restrict to consider only those systems that are physically realizable.

▶ **Assumption 3.35 (Physical realizability).** A compositional multi-robot system $\mathbf{S}_C$ is *physically realizable* by a hybrid system if the simulation map $\Psi^* : \Xi_{\mathbf{S}_H} \to \Sigma_{\mathbf{S}_C}$ is surjective. That is, for every execution of the compositional system, there is at least one corresponding physically realizable trajectory. ◀

From the simulation relation and assuming physical realizability, we can now state the main equivalence result that will justify our future choice of models.

▶ **Corollary 3.36 (Epistemic equivalence).** Let $\mathbf{S}_H$ be a hybrid system, and $\mathbf{S}_C$ be a compositional system with a compatible system abstraction $\Psi : \mathbf{S}_H \to \mathbf{S}_C$. If $\mathbf{S}_C$ is physically realizable, according to Assumption 3.35, then epistemic evolution of both systems are equivalent. That is, for every $n \in \mathbb{N}$:

$$\Psi_{\mathbf{E}}(\pi_{\mathbf{E}}(\Xi^n_{\mathbf{S}_H})) = \pi_{\mathbf{E}}(\Sigma^n_{\mathbf{S}_C})$$

where $\pi_{\mathbf{E}}$ projects the trajectories and executions to their sequence of epistemic states. ◀

*Proof.* We proceed by showing the double inclusion implying the equality:

- $\Psi_{\mathbf{E}}(\pi_{\mathbf{E}}(\Xi^n_{\mathbf{S}_H})) \subseteq \pi_{\mathbf{E}}(\Sigma^n_{\mathbf{S}_C})$. Let $\overline{e} \in \Psi_{\mathbf{E}}(\pi_{\mathbf{E}}(\Xi^n_{\mathbf{S}_H}))$, where $\overline{e} = \Psi_{\mathbf{E}}(e)$ for some epistemic sequence $e$ within a trajectory $\xi \in \Xi^n_{\mathbf{S}_H}$. By Theorem 3.33, there exists a unique execution $\sigma \in \Sigma^n_{\mathbf{S}_C}$ such that $\xi \preceq \sigma$. From the simulation map (Definition 3.34), we have $\Psi_{\mathbf{E}}(\mathbf{e}[t_i]) = \overline{e}_i$, where $\overline{e}_i$ is the $i$th component of the sequence $\overline{e}$, for each epistemic update time $t_i$. Therefore, $\overline{e} = (\overline{e}_0, \overline{e}_1, \dots)$ is the epistemic sequence of $\sigma$, hence $\overline{e} \in \pi_{\mathbf{E}}(\Sigma^n_{\mathbf{S}_C})$.

- $\pi_{\mathbf{E}}(\Sigma^n_{\mathbf{S}_C}) \subseteq \Psi_{\mathbf{E}}(\pi_{\mathbf{E}}(\Xi^n_{\mathbf{S}_H}))$. Let $\overline{e} \in \pi_{\mathbf{E}}(\Sigma^n_{\mathbf{S}_C})$ be an epistemic sequence from some execution $\sigma \in \Sigma^n_{\mathbf{S}_C}$. By Assumption 3.35, the map $\Psi^* : \Xi_{\mathbf{S}_H} \to \Sigma_{\mathbf{S}_C}$ is surjective, hence there exists a trajectory $\xi \in \Xi_{\mathbf{S}_H}$ such that $\Psi^*(\xi) = \sigma$. By Theorem 3.33, $\sigma$ is the unique execution where $\xi \preceq \sigma$. The simulation map gives us that $\Psi_{\mathbf{E}}(\mathbf{e}[t_i]) = \overline{e}_i$ for every epistemic update time $t_i$. As $\Psi_{\mathbf{E}}$ is surjective by Definition 3.28, there exists an epistemic sequence $e$ with $\Psi_{\mathbf{E}}(e) = \overline{e}$ and $e \in \pi_{\mathbf{E}}(\Xi^n_{\mathbf{S}_H})$. Thus, $\overline{e} \in \Psi_{\mathbf{E}}(\pi_{\mathbf{E}}(\Xi^n_{\mathbf{S}_H}))$.

■

The key observation is that the two system representations, hybrid or compositional, when connected by a system abstraction, have the same epistemic content. Such is the information available to a robot, and Corollary 3.36 attests one can meaningfully reason about the computational problems present in a multi-robot systems from either perspective. For the remainder of this thesis, we will adopt the compositional perspective to analyze and reason about the distributed systems, distributed tasks and their protocols.

## 3.5 CONCLUSION

This chapter establishes a unified framework for modeling distributed multi-robot systems through two complementary perspectives: a hybrid (Definition 3.10) and a compositional model (Definition 3.22). While the switching hybrid model captures the continuous physical dynamics that is present in real-life, the compositional

I/O automata isolates the computational information that is essential for tackling the distributed computing facet of multi-robot system.

The main contribution is the explicit separation of a robot state into its ontic (physical) and epistemic (computational) components within a unified framework for distributed multi-robot systems. This separation allows us to model the physical constraints and dynamics that are naturally present in a hybrid system, while abstracting away such details when focusing on the computational process that use them. The two components are interfaced through observation and control functions, which provide physical information to the computations and steer the physical control through its decisions.

We introduce time paths (Definition 3.1) as a geometric model relating global and local time, which we use to generalize schedulers and provide a unified notion of time for the multi-robot systems. This formalism is used in common through the execution structure of both models. The hybrid system trajectories (Definition 3.13) describe the continuous evolution of the ontic state across discrete jumps of the epistemic state, which are collected into solution sets (Definition 3.14) with all possible behaviors of the system. Meanwhile, the compositional system produces analogous sequences of global system states as executions (Definition 3.26), those collected into system runs (Definition 3.27).

The relationship between the two models is made precise as a system abstraction (Definition 3.28). The abstraction procedure is lifted to their execution structures in Theorem 3.33. Provided an extra assumption of physical realizability of the compositional system, we state the main result in Corollary 3.36: the information available to the decision-making process of the robots is model invariant. This invariance allows us to reason about distributed protocols, task solvability and the computational evolution in a simpler model fitting our tools in such a way that it is still relevant to the original hybrid system.

This chapter serves as a foundation for the analysis of robot systems done in the following ones, where the computational equivalence justifies our adoption of the compositional model. Namely, we will make use of the execution information to apply epistemic logic to reason about robot tasks in Chapter 4. The execution structure will then be used as the parametrizing space of the task data of a sheaf in Chapter 5. The system evolution through distributed protocols will then be put into a categorical perspective in Chapter 6.

We note that several limitations deserve attention in future works. While our physical realizability assumption grounds our theoretical developments, further work is needed to systematically verify it in practice. Additionally, while we handle scheduling with time paths, we do not delve into delays in the steps. We consider only homogeneous systems, with identical sensing, processing and acting capacities. Such restriction is not enforced by the modeling, and the introduction of different dynamics and observation deserves attention for its current usage in robotic missions. Beyond the possible choices of ontic and epistemic evolution, it would be interesting to formalize the different limitations in information gathering through variations of the observation functions, such as faulty sensors.

*"You have eyes, yet cannot see without light. If you are on the floor of a valley, you cannot see beyond your valley."*

— Frank Herbert, Dune [Her65]

# 4

# KNOWLEDGE IN MULTI-ROBOT SYSTEMS

As we saw in Chapter 3, distributed robotic systems can be understood as both dynamical systems and state machines. It was also shown that we can meaningfully reason about the execution properties in whichever abstraction is the most convenient, as their translation preserves epistemic information. In this chapter, we adopt the computational perspective and introduce the system frames, a variation of the runs and systems framework of Fagin et al. [Fag+95]. It forms a geometrical representation of the knowledge over time in a system, which models exactly when it is beyond the capacities of a robot to distinguish between possible epistemic states. As a compositional system provides the possible states following a given one, we relate them via a causal, temporal relation. At the same time, the environment dynamics and constraints in the robot model limit how much information can be gained or discerned by the robots. This uncertainty is captured by indistinguishability relations over these global states, providing the structure for reasoning about distributed coordination.

We use system frames to characterize knowledge requirements for task solvability, abstracting the protocol implementation to focus on the properties that it must satisfy to guarantee distributed coordination. We formalize such prop-

erties of systems, protocols and tasks in a temporal-epistemic logic with spatial propositions. These can then be verified by attaching such spatial information to the system frame, obtaining an interpreted system fitting the task. We identify the task-relevant information for three fundamental problems: explored regions of space for exploration, visited regions in space-time for surveillance, and robot positions for gathering. We then derive liveness properties that must be present in the implementation of a robotic system so that it is capable of completing such tasks.

This chapter corresponds to an extended version of Section 4 in [Cig+25], a joint work with Giorgio Cignarale, Stephan Felber, Eric Goubault and Hugo Rincon Galeana.

**Related work** Epistemic logic [Hin62] is a modal framework for reasoning about the uncertainty in systems, regarding both facts and beliefs. It is particularly suitable for modeling multi-agent systems, having been adapted to distributed systems by Halpern and Moses [HM90]. It has been applied to reasoning about information flow in public announcements [BMS16; BvDG22], to epistemic planning formalisms [Fab+21], and cleanly captures features of communication, such as the information present in the lack of communication [FKS21; GM20]. Temporal logic enables the specification of properties along executions, orthogonal to the static view of epistemic logic. Linear Temporal Logic (LTL) has been vastly used for verification and analysis of distributed protocols [Pnu77], including more recently robotic protocols [DBO17; DBO18], while the branching-time variant, such as Computational Tree Logic (CTL) allows for quantification over non-deterministic and asynchronous executions [Cla+18]. We favor the latter and employ throughout this chapter a mix of epistemic and branching-time operators, capturing the non-determinism that is naturally present in multi-robot systems in a compact manner that is compatible with its further usage in Chapter 5. Our epistemic operators follow the $S5$ for knowledge and distributed knowledge. Our temporal operators correspond to $AG$ (invariance), $EF$ (reachability), and $AF$ (liveness). This choice provides the necessary language to reason about multi-robot systems including asynchronicity and non-determinism. Our spatial propositions relate to the framework of spatial modal logics [vBB07], where modal operators can be interpreted over open sets of a topological space as interior and closure. This

approach also connects to the topological semantics for evidence in knowledge and belief [Bal+22]. In our setting, the regions of space (the mission space) serve as atomic propositions, and while we exploit their topological properties, our semantics remains interpreted on frames.

The composition of epistemic and temporal modalities enables expressing properties that must be satisfied, and be known, throughout an execution, without the need to focus on specific communication mechanisms. We work at this intersection, where the runs and systems framework of Fagin et al. [Fag+95] provides the foundations to our framework, which we extend with explicit causality relations. This approach to modeling knowledge in distributed system has been extensively studied, with particular attention to the role of common knowledge in coordination [Mos16]. Recent applications include characterization of two agent approximate agreement [ARL20] and sufficient epistemic conditions for stabilizing consensus [CFR24]. While it is well established in distributed computing, this is, to the best of our knowledge, the first systematic application of temporal-epistemic logics with spatial propositions derived from the mission space topology to distributed robotic tasks, which are encoded in the underlying model. Alternative approaches in distributed computing include Knight [Kni13], who develops the epistemic view of concurrency theory, and recent algebraic topological approaches by Goubault et al. [Gou+23], who model distributed knowledge on semi-simplicial structures. The latter provides a complementary perspective to the reasoning of knowledge over time that will be leveraged in Chapter 6.

**Chapter organization** Section 4.1 introduces system frames (Definition 4.1) as a geometric representation of knowledge evolution through time constructed from compositional multi-robot systems (Lemma 4.2). The structure is illustrated by the effects of limited visibility and asynchronicity (Examples 4.4 and 4.5).

Section 4.2 develops a temporal-epistemic logic with spatial propositions derived from the topology of the mission space (Definition 4.7) interpreted on a system frame (Definition 4.9). The main result establishes the persistence of knowledge regarding spatial facts under a system with perfect recall (Theorem 4.13), showing that once a robot learns a positive fact about the world, that knowledge is retained throughout the subsequent execution steps.

Section 4.3 applies this framework to three robot tasks. For exploration, we formalize the task and three different termination conditions (Definitions 4.14 to 4.16), and show that exploration liveness is both necessary and sufficient for parallel termination (Theorems 4.18 and 4.19), and that combined with communication liveness it can guarantee cooperative termination (Theorem 4.21). For surveillance, we show that the analogous temporal cut liveness as sufficient for cooperative termination on whether a mission space contains an intruder (Theorem 4.30). For gathering, we establish a correspondence with stabilizing agreement (Theorem 4.36), reducing spatial coordination to a form of consensus with regions as values.

## 4.1 SYSTEM FRAMES AND KNOWLEDGE

The compositional multi-robot system developed in Chapter 3 provides a framework where the physical dynamics and other environment constraints are encapsulated as an "external influence" applied in-between computational steps. We maintain this perspective when modeling the knowledge of a distributed multi-robot system. The possible global computational states throughout the execution of a distributed protocol are used to construct a system frame, which captures when it is beyond the capacities of a robot to distinguish between a pair of possible states. This indistinguishability relation abstracts the uncertainty in the environment dynamics and physical limitations. It includes limited visibility, asynchronicity and system malfunctions, all of which limit how much information can be gained or discerned from observations or reasoning.

▶ **Definition 4.1 (System frame).** A *system frame* over a set of agents $\mathcal{R}$ is a triple $\mathcal{K} = (\mathbf{C}, \leq, \sim)$ where

- $\mathbf{C}$ is a set of global states;
- $\leq \subseteq \mathbf{C} \times \mathbf{C}$ is a *causality* preorder on $\mathbf{C}$;
- $\sim : 2^{\mathcal{R}} \to 2^{\mathbf{C}\times\mathbf{C}}$ is an *indistinguishability* function assigning to each $A \subseteq \mathcal{R}$ an equivalence relation $\sim_A$ on $\mathbf{C}$, such that $\sim_{A\cup B} = \sim_A \cap \sim_B$ for all $A, B \subseteq \mathcal{R}$.

The shorthand $\sim_{\mathbf{r}}$ is used for $\sim_{\{r\}}$ when $A = \{\mathbf{r}\}$ is a singleton. ◀

The system frame is a variation of the runs and systems framework of Fagin et al. [Fag+95] in two ways: we explicitly model the causality relation across global

states of the system, and we parametrize the indistinguishability relations by subsets of robots, allowing us to reason about group knowledge and distributed information directly[1]. The abstract structure of a system frame captures the features required for reasoning about the knowledge in sequences of global states without the need to choose a particular computational model. The application of this framework to robotic systems is motivated by the fact that, unlike purely computational distributed systems, robots operate in a physical environment whose spatial constraints directly affect the epistemic structure.

▶ **Lemma 4.2 (System frame from a compositional system).** Let $\mathbf{S}_{\mathcal{R},\mathcal{E}}$ be a compositional multi-robot system and $\Sigma^n_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}$ its $n$-run. The triple $\mathcal{K}^n_{\mathbf{S}_{\mathcal{R},\mathcal{E}}} = (\mathbf{C}, \leq, \sim)$ defined by

- $\mathbf{C} = \{\mathbf{c} \mid \exists \sigma \in \Sigma^n_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}, \mathbf{c} \in \sigma\}$,
- $\mathbf{c} \leq_{\mathbf{r}} \mathbf{c}' \iff \exists \sigma \in \Sigma^n_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}$ with transition $\mathbf{c} \overset{\tau_{\mathbf{r}}}{\Rightarrow} \mathbf{c}'$, and $\leq = \bigcup_{\mathbf{r}\in\mathcal{R}} \leq_{\mathbf{r}}$,
- $\mathbf{c} \sim_A \mathbf{c}' \iff \pi_{\mathbf{r}}(\mathbf{c}) = \pi_{\mathbf{r}}(\mathbf{c}')$ for all $\mathbf{r} \in A$,

is a system frame over $\mathcal{R}$. ◀

*Proof.* We verify that construction yields a system frame, that is, that the triple $\mathbf{C}, \leq, \sim$ satisfies the conditions of Definition 4.1. Observe first that $\mathbf{C}$ is well-defined as the union of global states appearing in all executions of $\Sigma^n_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}$.

In order to assess that $\leq$ is a preorder, note that reflexivity holds by the definition allowing $\mathbf{c} = \mathbf{c}'$, as the causality relation captures reachability along executions, including over zero steps. For transitivity, let $\mathbf{c} \leq \mathbf{c}'$ and $\mathbf{c}' \leq \mathbf{c}''$. By Definition 3.25, transition relations can be concatenated, and paths from $\mathbf{c}$ to $\mathbf{c}'$ and from $\mathbf{c}'$ to $\mathbf{c}''$ concatenate to witness $\mathbf{c} \leq \mathbf{c}''$.

By construction, $\mathbf{c} \sim_A \mathbf{c}'$ holds if and only if $\pi_{\mathbf{r}}(\mathbf{c}) = \pi_{\mathbf{r}}(\mathbf{c}')$ for all $\mathbf{r} \in A$. Reflexivity, symmetry and transitivity of $\sim_A$ follow from the corresponding properties of equality of local epistemic states, hence the relation $\sim_A$ is an equivalence relation.

The property that $\sim_{A\cup B} = \sim_A \cap \sim_B$ also follows from construction. Observe that, by definition, $\mathbf{c} \sim_{A\cup B} \mathbf{c}'$ holds if and only if $\pi_{\mathbf{r}}(\mathbf{c}) = \pi_{\mathbf{r}}(\mathbf{c}')$ for all $\mathbf{r} \in A \cup B$.

[1] These modifications are motivated by the categorical machinery which will be employed on top of a system frame in Chapter 5, while retaining the expressiveness that is sufficient for the present chapter.

This is equivalent to requiring equality for all $\mathbf{r} \in A$ and for all $\mathbf{r} \in B$ separately, i.e., $\mathbf{c} \sim_A \mathbf{c}'$ and $\mathbf{c} \sim_B \mathbf{c}'$. ■

The sequences of states naturally induce a preorder relation that reflects their causality relation. Furthermore, this relation can be seen robot-wise, depending on which transition relation led to the update. Observe that most robots may remain unaltered across state transitions, as not all of them are active simultaneously unless the scheduler is synchronous. A robot may remain inactive for long or simply be unable to observe the change due to the environment. The set of global states where the same local state is found for some robot forms an equivalence class, all of which form a robot-wise indistinguishability relation. This is precisely the condition imposed on transition relations for an automata composition in Definition 3.22.

▶ **Example 4.3 (Lossy link).** Consider a compositional system $\mathbf{S}$ with two robots $a$ and $b$ that communicate synchronously. We model a scenario where the communication between $a$ and $b$ may fail in only one direction at each round. The system frame $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \sim, \leq)$ associated to the initial states and one step is depicted in Figure 4.1. The set of $\mathbf{C}$ contains the four initial configurations composed of binary inputs for each robot and the subsequent reachable states, where an arrow indicates in which direction communication has been successful. For instance, $(0, 0 \rightarrow)$ describes the first step after global state $(0, 0)$ where only robot $b$ received information from robot $a$.
The solid red and blue lines are the indistinguishability relations for $a$ and $b$, respectively, encoding the evolution of knowledge. In one case, the indistinguishability edge $\sim_a$ connects the state where $b$ has input $0$ to the state where $b$ has input $1$, as $a$ cannot see $b$'s local state directly without successful communication. The dotted arrows represent the causality relations $\leq$ and indicate the passage of local time steps. For instance, robot $a$ cannot distinguish the state $(0, 1, \rightarrow)$, where $b$ successfully received the message, from the state $(0, 1)$, where the message was lost. ◀

For the remainder of this chapter, we assume robots to have perfect recall (Assumption 3.19), that is, information is never lost from their local epistemic states. This assumption ensures that robots accumulate information monotoni-

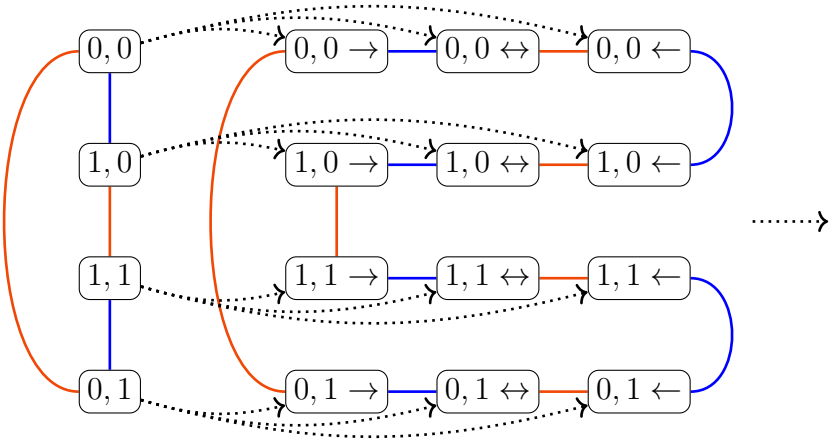



Figure 4.1
Depiction of a system frame one step of a lossy communication protocol. The red lines represent pairs of global states related by $\sim_a$, while blue lines are related by $\sim_b$. The dotted lines depict global states that are causally related by $\leq$. The system frame grows to the right as $\sigma$ progresses.

cally, a property that we will use later in Section 4.2 to reason about knowledge persistence.

The structure of the system frames depends on the model of the environment. Constraints in the robot updates and observations modify the possible global states and relations across them. This presents a form of "external influence" to the purely computational robot automata. For instance, when the observation of a robot is limited in distance, larger indistinguishability classes are created across all updates that happen outside its range. Depending on the communication model, the possible message loss increases the indistinguishability relations, as robots are incapable of differentiating between scenarios where messages were sent but lost and where they were not sent at all. As a scheduler defines directly the possible execution orders, a larger set of possible executions increases the possible sequences global states that cannot be distinguished. A sensor with some noise in its readings will provide observations that may fit different possible environment states. We can see explicitly the effect on the causality and indistinguishability relations of the system frame in the following examples.

▶ **Example 4.4 (Limited visibility system frame).** Consider two robots $a$ and $b$ in $2D$

space with observation radius $r$, and that they start in a configuration at distance $d > 2r$ apart. Because the distance $d$ is greater than $a$'s observation radius $r$, $a$'s local epistemic state is independent of $b$'s. In that case, for any two global states $\mathbf{c} = (e_a, e_b)$ and $\mathbf{c}' = (e_a, e'_b)$, it holds that $\mathbf{c} \sim_a \mathbf{c}'$. ◀

▶ **Example 4.5 (Asynchronous vs synchronous system frames).** In a synchronous system all robots are active simultaneously, as noted in Example 3.3, and therefore the causality relation $\leq$ forms a total order on the global states within an execution. In an asynchronous system, branchings in the execution order are created, as robots may activate independently, resulting in partially ordered global states. As such, two global states where different robots updated at the same time, i.e., in different branches, are incomparable under $\leq$. The partial order reflects the lack of synchronization. ◀

## 4.2 A TEMPORAL EPISTEMIC LOGIC OF SPACE

We introduce now a formal language to reason about robot tasks. While Chapter 3 defines an environment model to capture the physical system's dynamics, we reason about a mission space, the stage where the robots exist within this environment. We assume that robots can observe portions of a shared environment, which we define as the mission space.

▶ **Definition 4.6 (Mission space and space propositions).** The *mission space*, the physical environment where the robots operate, is a compact topological space $(\mathcal{X}, \tau)$. For every open set $U \in \tau$, we associate a unique atomic *space proposition* $\mathsf{s}(\mathsf{U})$. The set of all spatial propositions is then $\mathsf{S}_{\mathcal{X}} = \{\mathsf{s}(\mathsf{U}) \mid U \in \tau\}$. The spatial propositions are closed under union, satisfying $\mathsf{s}(\mathsf{U}) \wedge \mathsf{s}(\mathsf{V}) \Rightarrow \mathsf{s}(\mathsf{U} \cup \mathsf{V})$, and *contravariant*, satisfying $\mathsf{s}(\mathsf{V}) \Rightarrow \mathsf{s}(\mathsf{U})$ whenever $U \subseteq V$, for any $U, V \in \tau$ ◀

The spatial propositions $\mathsf{s}(\mathsf{U})$ capture the cumulative coverage of regions, rather than the mere visitation of points within them. The contravariance property reflects that observing a region entails observing any of its subregions, while the closure by conjunctions formalizes that observing both $U$ and $V$ implies observing $U \cup V$. Note that these propositions are global, instead of indexed by robots, since for exploration and surveillance the relevant question is whether a region has been

observed, not by whom. This situation will change for gathering, when individual positions become central. The mission space is modeled as a compact topological space so that we can tackle termination properties of robot mission. Compactness ensures that any open cover admits a finite subcover, allowing us to declare a space explored after visiting finitely many regions.

We define the following language to express our robot system and protocol properties.

▶ **Definition 4.7 (Language of $\mathbf{L}_{space}$).** Let $\mathsf{S}_{\mathcal{X}}$ be a set of spatial propositions. The *language* $\mathbf{L}_{space}$ is defined by the following grammar:

$$\varphi \coloneqq p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_{\mathbf{r}}\varphi \mid D_A\varphi \mid \Diamond\varphi \mid \Diamond!\varphi \mid \Box\varphi$$

where $p \in \mathsf{S}_{\mathcal{X}}$, $\mathbf{r} \in \mathcal{R}$, and $A \subseteq \mathcal{R}$. ◀

The language $\mathbf{L}_{space}$ defines a temporal epistemic logic. The operator $K_{\mathbf{r}}$ represents the individual *knowledge* of a robot $\mathbf{r}$, used for reasoning about local information and decisions, while $D_A$ captures the *distributed knowledge* of what is known collectively by a group. As it will be shown for the tasks in Section 4.3, this is essential for distributed coordination. The two temporal operators $\Diamond$ and $\Box$ represent persistent properties, which are either *eventually* or *always* true. Their combination allows us to express properties such as "robot $\mathbf{r}$ will eventually know that the entire space is explored", i.e., $\Diamond K_{\mathbf{r}}(\mathcal{X})$, or "if a region in space is explored, then it remains explored forever", i.e., $\mathsf{s}(\mathsf{U}) \Rightarrow \Box\mathsf{s}(\mathsf{U})$.

The atomic propositions are used to capture facts about the world that we reason about. The nature of facts can be either static or dynamic, depending on their behavior over time. Static facts are persistent in the sense that, once they become true, they remain true for the rest of an execution. For instance, it is convenient to consider that a region of space, once explored, remains explored forever. Dynamic facts may change its truth value over time, such as the position of a robot throughout a mission.

The indistinguishability relation is an equivalence relation, and consequently the epistemic operators satisfy the $S5$ axiom system. Note that this reflects only the epistemic state of the robot, as it reasons using its belief about the world. More specifically, for any robot $\mathbf{r} \in \mathcal{R}$ and formula $\varphi$:

- *Truth* ($T$): $K_{\mathbf{r}}\varphi \Rightarrow \varphi$

- *Positive introspection* (4): $K_{\mathbf{r}}\varphi \Rightarrow K_{\mathbf{r}}K_{\mathbf{r}}\varphi$
- *Negative introspection* (5): $\neg K_{\mathbf{r}}\varphi \Rightarrow K_{\mathbf{r}}\neg K_{\mathbf{r}}\varphi$
- *Distribution* (K): $K_{\mathbf{r}}(\varphi \Rightarrow \psi) \Rightarrow (K_{\mathbf{r}}\varphi \Rightarrow K_{\mathbf{r}}\psi)$

We now formalize the model for our logic. We use a variation of the interpreted system from the runs and systems framework of Fagin et al. [Fag+95], in which we employ an explicit causality relation.

▶ **Definition 4.8 (Interpreted system).** An *interpreted system* is a pair $M = (\mathcal{K}_{\mathbf{S}}, \ell)$, where $\mathcal{K}_{\mathbf{S}}$ is a system frame and $\ell \colon \mathbf{C} \to 2^{\mathsf{At}}$ is a *valuation function* assigning a set of true spatial propositions to each global state. ◀

▶ **Definition 4.9 (Semantics of $\mathbf{L}_{space}$).** Let $M = (\mathcal{K}_{\mathbf{S}}, \ell)$ be an interpreted system, where $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \leq, \sim)$ is a system frame. For a global state $\mathbf{c} \in \mathbf{C}$, the *semantics* is defined as follows:

$$
\begin{array}{lll}
M, \mathbf{c} \models p & \text{iff} & p \in \ell(\mathbf{c}) \\
M, \mathbf{c} \models \neg\varphi & \text{iff} & M, \mathbf{c} \not\models \varphi \\
M, \mathbf{c} \models (\varphi \wedge \psi) & \text{iff} & M, \mathbf{c} \models \varphi \text{ and } M, \mathbf{c} \models \psi \\
M, \mathbf{c} \models K_{\mathbf{r}}\varphi & \text{iff} & \forall \mathbf{c}' \in \mathbf{C} \text{ s.t. } \mathbf{c} \underset{\mathbf{r}}{\sim} \mathbf{c}', M, \mathbf{c}' \models \varphi \\
M, \mathbf{c} \models D_A\varphi & \text{iff} & \forall \mathbf{c}' \in \mathbf{C} \text{ s.t. } \mathbf{c} \underset{A}{\sim} \mathbf{c}', M, \mathbf{c}' \models \varphi \\
M, \mathbf{c} \models \Diamond\varphi & \text{iff} & \exists \mathbf{c}' \in \mathbf{C} \text{ s.t. } \mathbf{c} \leq \mathbf{c}', M, \mathbf{c}' \models \varphi \\
M, \mathbf{c} \models \Box\varphi & \text{iff} & \forall \mathbf{c}' \in \mathbf{C} \text{ s.t. } \mathbf{c} \leq \mathbf{c}', M, \mathbf{c}' \models \varphi \\
M, \mathbf{c} \models \Diamond!\varphi & \text{iff} & \forall (\mathbf{c}_i)_{i\geq 0}, \text{ where } \mathbf{c}_0 = \mathbf{c}, \mathbf{c}_i \leq \mathbf{c}_{i+1}, \exists \mathbf{c}' \in (\mathbf{c}_i)_{i\geq 0} \text{ s.t. } M, \mathbf{c}' \models \varphi
\end{array}
$$

The derived operator of *mutual knowledge* is defined as $E_A\varphi := \bigwedge_{\mathbf{r}\in A} K_{\mathbf{r}}\varphi$. A formula $\varphi$ is valid in $M$ if, for all $\mathbf{c} \in \mathbf{C}$, $M, \mathbf{c} \models \varphi$. In this case, it is abbreviated to $M \models \varphi$. ◀

▶ **Remark 4.10 (Temporal operators).** We employ three temporal modalities that quantify over execution paths from a given global state.

- Eventually ($\Diamond\varphi$): $\varphi$ holds in at least one execution path;
- *Inevitability* ($\Diamond!\varphi$): $\varphi$ holds in every execution path;

- Always ($\Box\varphi$): $\varphi$ holds in all future states of every execution path.

These operators express reachability, liveness and invariance properties, respectively. The inevitability operator $\Diamond!$ is particularly important for multi-robot coordination, as tasks require termination properties that hold regardless of execution choices. These correspond to the CTL operators $EF$, $AF$, and $AG$ [Cla+18].
◀

▶ **Lemma 4.11 (Relation of temporal modalities).** Let $M$ be an interpreted system in mission space $\mathcal{X}$. For any global state $\mathbf{c}$ and formula $\varphi \in \mathbf{L}_{space}$, it holds that:

$$M, \mathbf{c} \models \Box\varphi \Rightarrow M, \mathbf{c} \models \Diamond!\varphi \Rightarrow M, \mathbf{c} \models \Diamond\varphi$$

◀

*Proof.* Let $M$ be an interpreted system and let $\mathbf{c}$ be any state in the system frame of $M$.

> $M, \mathbf{c} \models \Box\varphi \Rightarrow M, \mathbf{c} \models \Diamond!\varphi$. Assume $M, \mathbf{c} \models \Box\varphi$. By the semantics of $\Box$, for every state $\mathbf{c}'$ such that $\mathbf{c} \leq \mathbf{c}'$, $M, \mathbf{c}' \models \varphi$ is satisfied. Let $e = (\mathbf{c}_0, \mathbf{c}_1, \dots)$ be any sequence of states where $\mathbf{c}_0 = \mathbf{c}$ and $\mathbf{c}_i \leq \mathbf{c}_{i+1}$. By reflexivity, $\mathbf{c}_0 = \mathbf{c}$ is a future of $\mathbf{c}$ and hence $M, \mathbf{c}_0 \models \varphi$. Therefore, along $e$ there exists a state $\mathbf{c}'$ where $M, \mathbf{c}' \models \varphi$. That matches the definition for $\Diamond!$, i.e., $M, \mathbf{c} \models \Diamond!\varphi$.
>
> $M, \mathbf{c} \models \Diamond!\varphi \Rightarrow M, \mathbf{c} \models \Diamond\varphi$. Assume $M, \mathbf{c} \models \Diamond!\varphi$. Let $e = (\mathbf{c}_0, \mathbf{c}_1, \dots)$ be a sequence such that $\mathbf{c}_0 = \mathbf{c}$ and $\mathbf{c}_i \leq \mathbf{c}_{i+1}$. By the semantics of $\Diamond!$, there exists $j \geq 0$ such that $M, \mathbf{c}_j \models \varphi$. By transitivity of $\leq$, $\mathbf{c} \leq \mathbf{c}_j$. This property is the definition of $\Diamond$, i.e., $M, \mathbf{c} \models \Diamond\varphi$.

■

We can now formalize the model assumption of certain atomic propositions to be *static*[2].

[2]This persistence property is similar to the knowledge gain property proven categorically in Theorem 6.29, where it emerges from morphisms of (semi-simplicial) models rather than as an axiom.

▶ **Assumption 4.12 (Persistence of spatial propositions).** Let $M = (\mathcal{K}_{\mathbf{S}}, \ell)$ be an interpreted system in mission space $\mathcal{X}$. Let $\mathsf{s}(\mathsf{U}) \in \mathsf{S}_{\mathcal{X}}$ be a spatial proposition in $\mathbf{L}_{space}$ and $\mathbf{c} \in \mathbf{C}$ a global state. We assume that all such *spatial propositions are static*:

$$M, \mathbf{c} \models \mathsf{s}(\mathsf{U}) \Rightarrow M, \mathbf{c} \models \Box \mathsf{s}(\mathsf{U})$$

◀

We lift now the persistence of static facts to the knowledge operator. This property connects the knowledge and temporal operators in our semantics, showing that knowledge about persistent facts is also persistent.

▶ **Theorem 4.13 (Spatial knowledge persistence).** Let $M = (\mathcal{K}_{\mathbf{S}}, \ell)$ be an interpreted system over an induced system frame (Lemma 4.2) satisfying perfect recall (Assumption 3.19) and atomic proposition persistence (Assumption 4.12). For any robot $\mathbf{r}$ and positive epistemic formula, i.e., built using only $(\wedge, \vee, K, D)$ without negation, $\varphi$ from static atomic propositions At, we have *knowledge persistence*:

$$M \models K_{\mathbf{r}}\varphi \Rightarrow \Box K_{\mathbf{r}}\varphi \tag{4.1}$$

*Proof.* Let $\mathbf{c} \in \mathbf{C}$ be a global state such that $M, \mathbf{c} \models K_{\mathbf{r}}\varphi$. We want to show that, for an arbitrary robot $\mathbf{r} \in \mathcal{R}$, and for any global state $\mathbf{c}' \in \mathbf{C}$ such that $\mathbf{c} \leq \mathbf{c}'$, we have $M, \mathbf{c}' \models K_{\mathbf{r}}\varphi$. Equivalently, by the semantics of $\mathbf{L}_{space}$ (Definition 4.9), that is to show that for all $\mathbf{c}'' \in \mathbf{C}$ with $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$ we have $M, \mathbf{c}'' \models \varphi$. We proceed by structural induction on the formula $\varphi$, assuming $\varphi$ is a positive formula build using only $\wedge, \vee, K_{\mathbf{r}}, D_A$ from static atomic propositions $p \in \mathsf{At}$ (Assumption 4.12).

> **Base case.** Let $\varphi = p$, where $p \in \mathsf{At}$. We assume $M, \mathbf{c} \models K_{\mathbf{r}}p$, which means that $M, \mathbf{c}_* \models p$ for all $\mathbf{c} \sim_{\mathbf{r}} \mathbf{c}_*$. Let $\mathbf{c}', \mathbf{c}'' \in \mathbf{C}$ such that $\mathbf{c} \leq \mathbf{c}'$ and $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$. By perfect recall, since $\mathbf{c} \leq_{\mathbf{r}} \mathbf{c}'$, the local epistemic states are related as prefixes, i.e., $\pi_{\mathbf{r}}(\mathbf{c}) \subseteq \pi_{\mathbf{r}}(\mathbf{c}')$. Similarly, as $\pi_{\mathbf{r}}(\mathbf{c}') = \pi_{\mathbf{r}}(\mathbf{c}'')$, we also have that $\pi_{\mathbf{r}}(\mathbf{c}) \subseteq \pi_{\mathbf{r}}(\mathbf{c}'')$. Furthermore, global epistemic states are unique within executions, as perfect recall specifies that local epistemic states encode the entire history of updates leading up to it, together with the monotonicity of the time path (Definition 3.1), the scheduler always has at least one robot active at every transition, making every global state differ by at least one local state. As such, there must exist a unique predecessor of $\mathbf{c}_*$ in the

execution leading to $\mathbf{c}''$ such that $\mathbf{c}_* \leq \mathbf{c}''$ and $\pi_{\mathbf{r}}(\mathbf{c}_*) = \pi_{\mathbf{r}}(\mathbf{c})$, i.e., $\mathbf{c}_* \sim_{\mathbf{r}} \mathbf{c}$. Consequently, the initial assumption that $M, \mathbf{c} \models K_{\mathbf{r}} p$ implies $M, \mathbf{c}_* \models p$. By the persistence of static facts (Assumption 4.12), as $p$ is static and $\mathbf{c}_* \leq \mathbf{c}''$, we have that $M, \mathbf{c}'' \models p$. As this holds for all $\mathbf{c}''$ such that $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$, we have that $M, \mathbf{c}' \models K_{\mathbf{r}} p$. Because our choice of $\mathbf{c}'$ was arbitrary for $\mathbf{c} \leq \mathbf{c}'$, it holds that $M, \mathbf{c} \models \Box K_{\mathbf{r}} p$.

**Inductive step.** Assume the theorem is true for positive epistemic formulas $\psi_1$ and $\psi_2$.

- Let $\varphi = \psi_1 \wedge \psi_2$. We assume $M, \mathbf{c} \models K_{\mathbf{r}}(\psi_1 \wedge \psi_2)$. From the semantics of knowledge, for all indistinguishable states $\mathbf{c}_*$ where $\mathbf{c} \sim_{\mathbf{r}} \mathbf{c}_*$, it holds that $M, \mathbf{c}_* \models \psi_1 \wedge \psi_2$, and individually $M, \mathbf{c}_* \models \psi_1$ and $M, \mathbf{c}_* \models \psi_2$. As those are all indistinguishable states of $\mathbf{c}$, by the same semantics, we have that $M, \mathbf{c} \models K_{\mathbf{r}} \psi_1$ and $M, \mathbf{c} \models K_{\mathbf{r}} \psi_2$. By the induction hypothesis, $M, \mathbf{c} \models \Box K_{\mathbf{r}} \psi_1$ and $M, \mathbf{c} \models \Box K_{\mathbf{r}} \psi_2$. Therefore, for any $\mathbf{c}'$ with $\mathbf{c} \leq \mathbf{c}'$, we have $M, \mathbf{c}' \models K_{\mathbf{r}} \psi_1$ and $M, \mathbf{c}' \models K_{\mathbf{r}} \psi_2$, giving us $M, \mathbf{c}' \models K_{\mathbf{r}}(\psi_1 \wedge \psi_2)$. As $\mathbf{c}'$ was arbitrary, we obtain $M, \mathbf{c} \models \Box K_{\mathbf{r}}(\psi_1 \wedge \psi_2)$.
- Let $\varphi = \psi_1 \vee \psi_2$. This case is analogous to the conjunction case.
- Let $\varphi = K_{\mathbf{r}'} \psi$, where $\mathbf{r}' \in \mathcal{R}$. We assume $M, \mathbf{c} \models K_{\mathbf{r}} K_{\mathbf{r}'} \psi$. Let $\mathbf{c}'$ be a global state such that $\mathbf{c} \leq \mathbf{c}'$, and $\mathbf{c}''$ be another one such that $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$. We want to show that $M, \mathbf{c}'' \models K_{\mathbf{r}'} \psi$, and for that we take an arbitrary $\mathbf{c}_z$ such that $\mathbf{c}'' \sim_{\mathbf{r}'} \mathbf{c}_z$ and show that $M, \mathbf{c}_z \models \psi$. By perfect recall, since $\mathbf{c} \leq \mathbf{c}'$ and $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$, there exists a unique predecessor state $\mathbf{c}_p$ such that $\mathbf{c}_p \leq \mathbf{c}''$ and $\mathbf{c}_p \sim_{\mathbf{r}} \mathbf{c}$. By the initial assumption that $M, \mathbf{c} \models K_{\mathbf{r}} K_{\mathbf{r}'} \psi$ and the fact that $\mathbf{c}_p \sim_{\mathbf{r}} \mathbf{c}$, we have that $M, \mathbf{c}_p \models K_{\mathbf{r}'} \psi$. Consider now $\mathbf{c}_z$, where $\mathbf{c}'' \sim_{\mathbf{r}'} \mathbf{c}_z$. As $\mathbf{c}_p \leq \mathbf{c}''$, we have by perfect recall of robot $\mathbf{r}'$ that $\mathbf{c}_p$ is a prefix of $\mathbf{c}''$ i.e., $\pi_{\mathbf{r}'}(\mathbf{c}_p) \subseteq \pi_{\mathbf{r}'}(\mathbf{c}'')$. As $\mathbf{c}'' \sim_{\mathbf{r}'} \mathbf{c}_z$, i.e., $\pi_{\mathbf{r}'}(\mathbf{c}'') = \pi_{\mathbf{r}'}(\mathbf{c}_z)$, we also have that $\pi_{\mathbf{r}'}(\mathbf{c}_p) \subseteq \pi_{\mathbf{r}'}(\mathbf{c}_z)$. Again by perfect recall, there exists a unique predecessor state $\mathbf{c}_q$ of $\mathbf{c}_z$ such that $\mathbf{c}_q \leq \mathbf{c}_z$ and $\mathbf{c}_q \sim_{\mathbf{r}'} \mathbf{c}_p$. Since $M, \mathbf{c}_p \models K_{\mathbf{r}'} \psi$ and $\mathbf{c}_q \sim_{\mathbf{r}'} \mathbf{c}_p$, we have that $M, \mathbf{c}_q \models \psi$. As $\psi$ is a positive epistemic formula, it persists through causal steps by persistence of static facts. As $\mathbf{c}_q \leq \mathbf{c}_z$, we obtain that $M, \mathbf{c}_z \models \psi$. Since $\mathbf{c}_z$ was an arbitrary choice for $\mathbf{c}'' \sim_{\mathbf{r}'} \mathbf{c}_z$, by the semantics of knowledge, we have that $M, \mathbf{c}'' \models K_{\mathbf{r}'} \psi$. Similarly, as $\mathbf{c}''$ was also arbitrary for $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$, we

have that $M, \mathbf{c}' \models K_{\mathbf{r}} K_{\mathbf{r}'} \psi$. Finally, as $\mathbf{c}'$ was also arbitrary for $\mathbf{c} \leq \mathbf{c}'$, by the semantics of always, we obtain that $M, \mathbf{c} \models \Box K_{\mathbf{r}} K_{\mathbf{r}'} \psi$.

– Let $\varphi = D_A \varphi$, where $A \subseteq \mathcal{R}$. We assume $M, \mathbf{c} \models K_{\mathbf{r}} D_A \varphi$. Let $\mathbf{c}'$ be an arbitrary global state such that $\mathbf{c} \leq \mathbf{c}'$, and $\mathbf{c}''$ such that $\mathbf{c}' \sim_{\mathbf{r}} \mathbf{c}''$. We need to show that $M, \mathbf{c}'' \models D_A \psi$. The argument follows the same structure as for $\varphi = K_{\mathbf{r}'} \psi$, with all usages of $\sim_{\mathbf{r}'}$ replaced by $\sim_A$ throughout the inner steps. Specifically, those are:

  (i) We find the unique predecessor $\mathbf{c}_p$ of $\mathbf{c}''$ such that $\mathbf{c}_p \leq \mathbf{c}''$ and $\mathbf{c}_p \sim_{\mathbf{r}} \mathbf{c}$ by the perfect recall property. From the initial assumption we get that $M, \mathbf{c}_p \models D_A \psi$.

  (ii) For an arbitrary global state $\mathbf{c}_z$ such that $\mathbf{c}'' \sim_A \mathbf{c}_z$, we have the unique predecessor $\mathbf{c}_q$ such that $\mathbf{c}_q \leq \mathbf{c}_z$ and $\mathbf{c}_q \sim_A \mathbf{c}_z$.

  (iii) From the previously obtained fact that $M, \mathbf{c}_p \models D_A \psi$ and that $\mathbf{c}_q \sim_A \mathbf{c}_p$, we can now derive by the semantics of distributed knowledge that $M, \mathbf{c}_q \models \psi$.

  (iv) As $\psi$ persists along causal paths, now through $\mathbf{c}_q \leq \mathbf{c}_z$, we have that $M, \mathbf{c}_z \models \psi$.

  As $\mathbf{c}_z$ was arbitrary for $\mathbf{c}'' \sim_A \mathbf{c}_z$, we have that $M, \mathbf{c}'' \models D_A \psi$. Since $\mathbf{c}''$ and $\mathbf{c}'$ were also arbitrary, we obtain that $M, \mathbf{c} \models \Box K_{\mathbf{r}} D_A \psi$.

As the property Equation (4.1) holds for atomic propositions and is preserved under all constructions of positive epistemic formulas, we have that $M, \mathbf{c} \models \Box K_{\mathbf{r}} \varphi$. ■

Theorem 4.13 captures the intuitive property that once a robot learns something, expressed as a positive fact about the world, that knowledge is retained forever.

This temporal-epistemic logic provides the formal language for specifying robot system properties and task requirements. The spatial propositions $\mathsf{S}_{\mathcal{X}}$ capture the geometry present in the robot tasks, while the modal operators describe the knowledge and its evolution, both needed for reasoning about distributed coordination.

## 4.3 MODELING ROBOT TASKS

Having established system frames as collecting the possible executions in Section 4.1, and a temporal-epistemic logic with spatial propositions from the mission space in Section 4.2, we now turn analyzing robot tasks there expressed. We start by considering the information which is attributed to the space, in order to capture exactly the task-relevant properties that are required to reason about robot tasks. Note that, while the explored zone is central to the exploration task, which our base language of space is shown sufficient to capture, the position of the robots is more relevant for gathering. This requirement will warrant an extension of the language to include dynamic propositions. Similarly, the explored zone will require a temporal aspect in the surveillance task, requiring a further extension.

### 4.3.1 EXPLORATION

The exploration task requires a group of robots to collectively visit all parts of a mission space. This task is usually accompanied by some notion of "observation range", which roughly translates to which part of the space are observed for each possible ontic state of the robot. We abstract these concepts as part of the observation function inside the robot model. We assume that the valuation function of the system model correctly attributes to each global state of a system, throughout an execution, which parts of the space $\mathcal{X}$ are visited. That is, an atomic spatial proposition $\mathsf{s}(\mathsf{U}) \in \mathsf{S}_{\mathcal{X}}$ now describes the information that the region of space $U$ is within the observation range of some robot. Note that both the logic and the semantics remain unchanged. Provided this interpretation of the language of space defined in Section 4.2, we define now the set of properties that must be respected by any interpreted system so that it can be used to reason about the exploration task.

▶ **Definition 4.14 (Simple exploration).** An interpreted system $M = (\mathcal{K}_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}, \ell)$ satisfies *simple exploration* if, for any group of robots $A \subseteq \mathcal{R}$ and any region of its mission space $U \subset X$, the following holds:

- *Exploration agency*: $M \models \mathsf{s}(\mathsf{U}) \Rightarrow D_{\mathcal{R}}\mathsf{s}(\mathsf{U})$, i.e., if a region $U$ has been explored, there is distributed knowledge among robots that it has been explored;

- *Exploration independence*: $M \models D_A\mathsf{s}(\mathsf{U}) \Rightarrow \bigwedge_{\mathbf{r}\in A} K_{\mathbf{r}}\mathsf{s}(\mathsf{V}_{\mathbf{r}})$ for some family $(V_{\mathbf{r}})_{\mathbf{r}\in A}$ such that $\bigcup_{\mathbf{r}\in A} V_{\mathbf{r}} = U$, i.e., any spatial proposition known distributively by robots in $A$ is given by the union of the spatial propositions known by robots in $A$;
- *Stable space propositions*: $M \models \bigwedge_{U\subseteq\mathcal{X}\text{ open}}(\mathsf{s}(\mathsf{U}) \Rightarrow \Box\mathsf{s}(\mathsf{U}))$, i.e., once a region has been explored, it is forever explored;
- *Stable exploration knowledge*: $M \models K_{\mathbf{r}}\mathsf{s}(\mathsf{U}) \Rightarrow (\Box K_{\mathbf{r}}\mathsf{s}(\mathsf{U}))$, i.e., once a spatial proposition is known by a robot, it will be known forever by that robot.

◀

The conditions for simple exploration set how a system should behave in order to reason about exploration in our framework. Exploration agency is a coherence check so that a region can only be explored by a robot. Exploration independence ensures that the distributed knowledge of a group arises only from the individual knowledge of its members. Stable space propositions and stable exploration knowledge guarantee progress in the exploration, formalizing that regions remain explored, and this knowledge is preserved. They are provided by Assumption 4.12 and Theorem 4.13, respectively.

We now define that for a system to (physically) solve the exploration task, it must inevitably visit the entire mission space, along all possible executions.

▶ **Definition 4.15 (Exploration task).** An interpreted system $M$ on mission space $\mathcal{X}$ satisfies the *exploration task* if it satisfies $M \models \Diamond!\mathsf{s}(\mathcal{X})$. ◀

A system is required to both satisfy the exploration task and terminate. Termination can be understood at different levels of coordination, which now involve the actual knowledge of the robots, beyond the ontic execution of the task. We distinguish three cases: centralized knowledge (parallel termination), individual knowledge (selfish termination) and collective knowledge (cooperative termination).

▶ **Definition 4.16 (Termination conditions for exploration).** Let $M = (\mathcal{K}_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}, \ell)$ be an interpreted system consistent with simple exploration and on mission space $\mathcal{X}$. The system $M$ satisfies a *termination condition* if it is consistent with one of the following:

- *Parallel termination*: $M \models \Diamond! D_{\mathcal{R}} \mathsf{s}(\mathcal{X})$, i.e., no single robot needs to know that the whole space is explored;
- *Selfish termination*: $M \models \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$, i.e., each robot needs to know that the whole space is explored;
- *Cooperative termination*: $M \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$, i.e., each robot needs to know that the whole space is explored and that other robots know that the whole space is explored.

◀

Parallel termination is the weakest condition, requiring only for the mission space to be collectively explored, however no single robot needs to know that this has been done. Selfish termination now imposes some coordination, as all robots must know that the entire space has been explored. This condition is the weakest to allow for robots are to terminate independently. Cooperative termination offers the possibility of a coordinated cessation of the mission, as robots are required to further know that every other robot knows that the space has been explored. This prevents situations in which some robot continues to execute unaware that the mission is complete.

In order to achieve some termination condition, we need to ensure that the robots actually visit all regions of the space. Note the distinction between the actual behavior, represented by a true space proposition, and the knowledge of it. We state now the property that the physical behavior must be correct, so to then reason about the knowledge of the system under that precondition.

▶ **Definition 4.17 (Exploration liveness).** An interpreted system $M$ on mission space $\mathcal{X}$ satisfies *exploration liveness* if there exists an open cover of $\mathcal{X}$, i.e., a collection of open sets $\mathcal{U}$ with $\bigcup_{U \in \mathcal{U}} U = \mathcal{X}$, such that $M \models \bigwedge_{U \in \mathcal{U}} \Diamond! \mathsf{s}(U)$. ◀

Exploration liveness captures the requirement that a group of robots must visit all region of the mission space in all of their possible executions. We now show that this property is equivalent to parallel termination under the simple exploration conditions. First by establishing that achieving parallel termination requires exploration liveness.

▶ **Theorem 4.18 (Exploration liveness necessity).** Let $M$ be an interpreted system consistent with simple exploration. If $M$ satisfies parallel termination, then $M$ must satisfy exploration liveness.

*Proof.* Let $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \leq, \sim)$ be the system frame of $M$. We proceed by contraposition. Suppose that $M$ does not satisfy exploration liveness (Definition 4.17). Then, for every open cover $\mathcal{U}$ of $\mathcal{X}$, there exists some region $U \in \mathcal{U}$ and global state $\mathbf{c}_0 \in \mathbf{C}$, such that $M, \mathbf{c}_0 \not\models \Diamond!\mathsf{s}(\mathsf{U})$. Consider an arbitrary open cover $\mathcal{U}$ of $\mathcal{X}$. By the negation of exploration liveness applied to this cover, there exists some $U \in \mathcal{U}$ and $\mathbf{c}_0 \in \mathbf{C}$ such that $M, \mathbf{c}_0 \not\models \Diamond!\mathsf{s}(\mathsf{U})$. By the semantics of $\Diamond!$ (Definition 4.9), this means that there exists an execution sequence $\sigma = (\mathbf{c}_i)_{i\geq 0}$ with $\mathbf{c}_i \leq \mathbf{c}_{i+1}$ such that $M, \mathbf{c}_i \not\models \mathsf{s}(\mathsf{U})$ for every $i \geq 0$.

Note that $U \subseteq \mathcal{X}$, and that by the contravariance of spatial propositions (Definition 4.6) $\mathsf{s}(\mathcal{X}) \Rightarrow \mathsf{s}(\mathsf{U})$. By the contrapositive, i.e., $\neg\mathsf{s}(\mathsf{U}) \Rightarrow \neg\mathsf{s}(\mathcal{X})$, we have that $M, \mathbf{c}_i \not\models \mathsf{s}(\mathcal{X})$ for every $\mathbf{c}_i \in \sigma$. By exploration agency (Definition 4.14), $\mathsf{s}(\mathcal{X})$ holds if and only if $D_{\mathbf{r}}\mathsf{s}(\mathcal{X})$. Thus, $M, \mathbf{c}_i \not\models D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$, for every $\mathbf{c}_i \in \sigma$. By the semantics of $\Diamond!$, since there exists an execution sequence $\sigma$ where $D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$ never holds, we have that $M, \mathbf{c}_0 \not\models \Diamond!D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. As there exists a global state from which parallel termination is not satisfied, the system $M$ does not satisfy parallel termination. Thus, exploration liveness is necessary for parallel termination. ■

In other words, Theorem 4.18 formalizes the intuition that if a system should know that it solves exploration, it is necessary for the multi-robot system to be capable of visiting every part of the space in every possible execution. Exploration is not possible unless the entire space is always reachable, as some environment constraint or physical limitation in the system model could prevent it.

▶ **Theorem 4.19 (Exploration liveness sufficiency).** Let $M$ be an interpreted system with a mission space $\mathcal{X}$ consistent with simple exploration. If $M$ satisfies exploration liveness, then it also satisfies parallel termination, i.e., $M \models \Diamond!D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$.

*Proof.* Let $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \leq, \sim)$ be the system frame of $M$. Consider an arbitrary global state $\mathbf{c}_0 \in \mathbf{C}$ and any sequence $\sigma = (\mathbf{c}_i)_{i\geq 0}$ from $\mathbf{c}_0$ with $\mathbf{c}_i \leq \mathbf{c}_{i+1}$. By exploration liveness (Definition 4.17), there exists an open cover $\mathcal{U}$, i.e., $\bigcup_{U\in\mathcal{U}} U = \mathcal{X}$, such that for each $U \in \mathcal{U}$, it holds that $M, \mathbf{c}_0 \models \Diamond!\mathsf{s}(\mathsf{U})$. By the semantics of $\Diamond!$ (Definition 4.9), this means that for each $U \in \mathcal{U}$, there exists some $\mathbf{c}' \in \sigma$ where $M, \mathbf{c}' \models \mathsf{s}(\mathsf{U})$.

By Definition 4.6, the space $\mathcal{X}$ is compact, hence there exists a finite cover $\mathcal{U}' = \{V_1, \ldots, V_k\} \subseteq \mathcal{U}$ with $\bigcup_{i=1}^{k} V_i = \mathcal{X}$. For each $V_i \in \mathcal{U}'$, there exists an index $[t_i]$ such that $M, \mathbf{c}_{[t_i]} \models \mathsf{s}(\mathsf{V_i})$. Let $[t_{\max}] = \max\{[t_i] \mid i \in [1,k]\}$. By the persistence of spatial propositions (Assumption 4.12), once $\mathsf{s}(\mathsf{V_i})$ becomes true, it remains true. Therefore, $M, \mathbf{c}_{[t_{\max}]} \models \mathsf{s}(\mathsf{V_i})$ for all $i \in [1,k]$. As such, $M, \mathbf{c}_{[t_{\max}]} \models \bigwedge_{i=1}^{k} \mathsf{s}(\mathsf{V_i})$. As $\bigcup_{i=1}^{k} V_i = \mathcal{X}$, we have that $M, \mathbf{c}_{[t_{\max}]} \models \mathsf{s}(\mathcal{X})$. By exploration agency (Definition 4.14), $\mathsf{s}(\mathcal{X}) \Rightarrow D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. Thus, $M, \mathbf{c}_{[t_{\max}]} \models D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$.

Since the sequence $(\mathbf{c}_i)_{i\geq 0}$ from $\mathbf{c}_0$ was arbitrary, every such sequence eventually reaches a state where $D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$ holds. By the semantics of $\Diamond!$ and the fact that $\mathbf{c}_0$ was also arbitrary, we have that $M \models \Diamond! D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. That is, $M$ satisfies parallel termination. ■

Theorem 4.19 guarantees that if every part of the mission space can be visited (exploration liveness), then the entirety of it will eventually be explored (selfish termination). The argument depends on the compactness of the mission space $\mathcal{X}$, which ensures that any open cover admits a finite subcover. As such, the open cover $\mathcal{U}$ from exploration liveness can be reduced to finitely many regions $\{U_1, \ldots, U_k\}$ whose union is $\mathcal{X}$. Since the spatial propositions are persistent, the robots only need to verify finitely many facts $\mathsf{s}(\mathsf{U_1}), \ldots, \mathsf{s}(\mathsf{U_k})$ to conclude that $\mathsf{s}(\mathcal{X})$ holds, which can be achieved in finite time.

We established with exploration liveness the condition under which the robots will physically visit the entire mission space over time. In order for this exploration to be coordinated, we also need to establish that the information individually obtained by a robot is shared with the rest of the group. The communication liveness property serves this purpose, as it states that anything learned by an individual robot is propagated to all other robots.

▶ **Definition 4.20 (Communication liveness).** An interpreted system $M$ satisfies *communication liveness* if for any robot $\mathbf{r} \in \mathcal{R}$ and region of space $\mathsf{s}(\mathsf{U})$, it holds that $M \models K_{\mathbf{r}}\mathsf{s}(\mathsf{U}) \Rightarrow \Diamond! E_{\mathcal{R}}\mathsf{s}(\mathsf{U})$. ◀

Communication liveness is a property of the protocol within the robot model. It translated the requirement of the communication to be sufficiently efficient, so that robots actively share all of their knowledge with everyone else. As such, if it

is possible for one robot to learn a fact $\mathsf{s}(\mathsf{U})$, then it is also possible for every robot to learn $\mathsf{s}(\mathsf{U})$.

We can now show that a system providing exploration liveness and communication liveness, each attesting to it satisfies the required physical and communication behaviors, can also satisfy the exploration task with a cooperative termination, the strongest variant.

▶ **Theorem 4.21 (Exploration and communication liveness imply cooperative termination).** Let $M$ be an interpreted system with mission space $\mathcal{X}$ consistent with simple exploration, satisfying exploration liveness, whose robots satisfy communication liveness. Then $M$ satisfies the cooperative termination property, i.e., $M \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$.

*Proof.* Let $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \leq, \sim)$ be the system frame of $M$. By exploration liveness (Definition 4.17) and Theorem 4.19, we have that $M \models \Diamond! D_{\mathcal{R}} \mathsf{s}(\mathcal{X})$. Consider an arbitrary global state $\mathbf{c}_0 \in \mathbf{C}$ and any sequence $\sigma = (\mathbf{c}_i)_{i \geq 0}$ from $\mathbf{c}_0$ where $\mathbf{c}_i \leq \mathbf{c}_{i+1}$. By the semantics of $\Diamond!$ (Definition 4.9), there exists $j \geq 0$ such that $M, \mathbf{c}_j \models D_{\mathcal{R}} \mathsf{s}(\mathcal{X})$. By exploration independence (Definition 4.14), there exists a family $(U_{\mathbf{r}})_{\mathbf{r} \in \mathcal{R}}$ with $\bigcup_{\mathbf{r} \in \mathcal{R}} U_{\mathbf{r}} = \mathcal{X}$ such that $M, \mathbf{c}_j \models K_{\mathbf{r}} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$ for each $\mathbf{r} \in \mathcal{R}$. By knowledge persistence (Theorem 4.13), we have that $M, \mathbf{c}_j \models \Box K_{\mathbf{r}} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$. More precisely, by Lemma 4.11, we obtain $M, \mathbf{c}_j \models \Diamond! K_{\mathbf{r}} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$, allowing for the assumption of communication liveness (Definition 4.20) to provide that $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$ for each $\mathbf{r} \in \mathcal{R}$.

By the semantics of $\Diamond!$, any sequence $(\mathbf{c}_i)_{i \geq j}$ from $\mathbf{c}_j$ eventually reaches a global state $\mathbf{c}_k$ where $M, \mathbf{c}_k \models E_{\mathcal{R}} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$. By the persistence of spatial propositions (Assumption 4.12), any such sequence reaches a global state $\mathbf{c}_l$ where $M, \mathbf{c}_l \models \bigwedge_{\mathbf{r} \in \mathcal{R}} E_{\mathcal{R}} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$. That is, at $\mathbf{c}_l$, we have that $M, \mathbf{c}_l \models \bigwedge_{\mathbf{r} \in \mathcal{R}} \bigwedge_{\mathbf{r}' \in \mathcal{R}} K_{\mathbf{r}'} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$. Since the family $\bigcup_{\mathbf{r} \in \mathcal{R}} U_{\mathbf{r}} = \mathcal{X}$, by the fact that the spatial propositions are closed under union (Definition 4.6) we have that $\bigwedge_{\mathbf{r} \in \mathcal{R}} \mathsf{s}(\mathsf{U}_{\mathbf{r}}) \Rightarrow \mathsf{s}(\mathcal{X})$. By commutativity of conjunction, we can organize the formula as $M, \mathbf{c}_l \models \bigwedge_{\mathbf{r}' \in \mathcal{R}} \bigwedge_{\mathbf{r} \in \mathcal{R}} K_{\mathbf{r}'} \mathsf{s}(\mathsf{U}_{\mathbf{r}})$. By applying the K axiom to each $\mathbf{r}' \in \mathcal{R}$, we obtain $M, \mathbf{c}_l \models \bigwedge_{\mathbf{r}' \in \mathcal{R}} K_{\mathbf{r}'} \left( \bigwedge_{\mathbf{r} \in \mathcal{R}} \mathsf{s}(\mathsf{U}_{\mathbf{r}}) \right)$. With the previous conclusion regarding the union of the propositions, we get that $M, \mathbf{c}_l \models \bigwedge_{\mathbf{r}' \in \mathcal{R}} K_{\mathbf{r}'} \mathsf{s}(\mathcal{X})$, i.e., $M, \mathbf{c}_l \models E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$.

As every sequence from $\mathbf{c}_j$ reaches such $\mathbf{c}_l$, we have that $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$. As any sequence from $\mathbf{c}_0$ reaches such $\mathbf{c}_j$, we have that $M, \mathbf{c}_0 \models \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$. As $\mathbf{c}_0$ was arbitrary, we have that $M \models \Diamond! E_{\mathcal{R}} \mathsf{s}(\mathcal{X})$. Consider now any global state

$\mathbf{c}_\ell$ such that $M, \mathbf{c}_\ell \models E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. For any $\mathbf{r}' \in \mathcal{R}$, consider $\mathbf{c}'$ such that $\mathbf{c}_\ell \sim_{\mathbf{r}'} \mathbf{c}'$. As $M \models \Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$, we have that $M, \mathbf{c}' \models \Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$ for any such $\mathbf{c}'$. By the semantics of knowledge, $M, \mathbf{c}_\ell \models K_{\mathbf{r}'}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. As $\mathbf{r}'$ was arbitrary, we have that $M, \mathbf{c}_\ell \models E_{\mathcal{R}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. Since every sequence from $\mathbf{c}_0$ reaches such $\mathbf{c}_\ell$, we obtain $M, \mathbf{c}_0 \models \Diamond! E_{\mathcal{R}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. As $\mathbf{c}_0$ was arbitrary, we conclude that $M \models \Diamond! E_{\mathcal{R}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. ■

Note that, since exploration liveness already guarantees that every part of the mission space *can* be visited, the remaining question is *whether* the computational process employed by the robots can know when this is accomplished. The main implication of Theorem 4.21 is the decoupling of the exploration problem. A system designer can focus on providing, independently, a protocol guaranteeing the coverage of the space and the efficient communication, knowing that these two properties together are sufficient to solve the more complex task of exploration. Furthermore, that such a protocol will enable all robots to terminate only when it is mutually understood as complete, a fundamental requirement in a decentralized system.

▶ **Corollary 4.22.** Let $M$ be an interpreted system consistent with simple exploration, satisfying exploration liveness and communication liveness. Then, cooperative termination, selfish termination and parallel termination are equivalent. ◀

*Proof.* Let $M$ be an interpreted system satisfying exploration liveness (Definition 4.17) and communication liveness (Definition 4.20). By Theorem 4.19, we have that $M \models \Diamond! D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$, i.e., it satisfies parallel termination (Definition 4.16). By Theorem 4.21, we have that $M \models \Diamond! E_{\mathcal{R}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$, i.e., it satisfies cooperative termination.

We show now that the three termination conditions form a chain of implications. Consider an arbitrary global state $\mathbf{c} \in \mathbf{C}$. Assume that $M, \mathbf{c} \models \Diamond! E_{\mathcal{R}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. By the semantics of $\Diamond!$, there exists a global state $\mathbf{c}'$ where $M, \mathbf{c}' \models E_{\mathcal{R}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. That is, at $\mathbf{c}'$, we have that $M, \mathbf{c}' \models K_{\mathbf{r}}\Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$ for each $\mathbf{r} \in \mathcal{R}$. By the T axiom, we have $M, \mathbf{c}' \models \Diamond! E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$, i.e., selfish termination holds at $\mathbf{c}'$. By the semantics of $\Diamond!$, every sequence $(\mathbf{c}_i)_{i\geq 0}$ from $\mathbf{c}'$ has some state $\mathbf{c}_k$ where $M, \mathbf{c}_k \models E_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. By the definition of $E_{\mathcal{R}}$ (Definition 4.7), $E_{\mathcal{R}}\mathsf{s}(\mathcal{X}) \Rightarrow D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. Thus, $M, \mathbf{c}' \models \Diamond! D_{\mathcal{R}}\mathsf{s}(\mathcal{X})$. Since every sequence from $\mathbf{c}$ reaches $\mathbf{c}'$, we have that

it satisfies parallel termination. As such, cooperative termination implies selfish termination, which in turn implies parallel termination. As $M$ satisfies both cooperative and parallel termination, all three termination conditions are equivalent. ■

Corollary 4.22 shows that the three notions of termination all collapse during the design of robot systems under exploration liveness and protocols under communication liveness. The usage of such assumptions would allow targeting the verification of the weakest property, parallel termination, while still obtaining the coordination level of cooperative termination.

### 4.3.2 SURVEILLANCE

We now consider the surveillance task, an extension of exploration where the robots must now cover a space over time, making sure that there are no intruders inside. An *intruder* is another robot that may exist in the same mission space. We assume that such an intruder can be detected by a member of the multi-robot system if it is found within one's perception range. The *surveillance task* demands for the system to decide either that an intruder exists if it is found, or that there are none. In order to formalize the surveillance task, we extend the mission space to incorporate a temporal dimension.

▶ **Definition 4.23 (Mission space-time and space-time propositions).** The *mission space-time* is the cylinder $\mathcal{X}_{[0,1]} = \mathcal{X} \times [0,1]$, between the mission space $\mathcal{X}$ and the normalized time interval of the mission. The set of *space-time propositions* associated to space-time $\mathcal{X}_{[0,1]}$ is $\mathsf{ST}_{\mathcal{X}_{[0,1]}} = \{\mathsf{st}(W)\}_{W \subseteq \mathcal{X}_{[0,1]}} \cup \{\text{FOUND}, \text{SECURE}\}$, where an atomic space proposition is assigned to each open set $W \subseteq \mathcal{X}_{[0,1]}$, in addition to two propositions FOUND and SECURE. ◀

The mission space-time represents every point of the mission space at every time. Note that the path, or *worldline*, followed by each robot within that space-time is a continuous curve within this cylinder. The robots' task is then to explore the mission space-time, in such a way that they can guarantee that the combined observation regions will intersect the worldline of any possible intruder. The space-time propositions $\mathsf{ST}_{\mathcal{X}}$ of a new space-time $\mathcal{X}$ are then the analogous space propositions defined in the original language, extended with two new values,

FOUND and SECURE, that capture the possible local decisions. As such, we consider that the conditions for simple exploration (Definition 4.14) are satisfied with respect to $\mathcal{X}_{[0,1]}$. Most remarkably, since we express time as an extra dimension, the persistence of the validity space-time propositions still holds.

The challenge now is to ensure that the robots' paths in space-time form a "barrier" through which any possible intruder is caught. This leads to the notion of a *temporal cut* of the mission space-time $\mathcal{X}_{[0,1]} = \mathcal{X} \times [0,1]$: the graph of a continuous function $f : \mathcal{X} \to [0,1]$, i.e., $\mathrm{gr}(f) = \{(x, f(x)) \mid x \in \mathcal{X}\}$. By verifying an open region $W$ of space-time that contains this graph, i.e., $\mathrm{gr}(f) \subseteq W$, the robots satisfy a proposition $\mathrm{st}(W)$ that creates a barrier that intersects possible worldline of an intruder. We state now the geometric meaning of that strategy. Note that, while we focus on a continuous barrier for simplicity, any set of observations whose union is guaranteed to intersect any possible intruder worldline would suffice.

▶ **Lemma 4.24 (Curve catching lemma).** Let $\alpha\colon [0,1] \to \mathcal{X}$ be an arbitrary curve in $\mathcal{X}$, and $f\colon \mathcal{X} \to [0,1]$ a continuous function. Then, the graph of $\alpha$, $\mathrm{gr}(\alpha) \coloneqq \{(\alpha(t), t) \mid t \in [0,1]\}$ and the graph of $f$, $\mathrm{gr}(f) \coloneqq \{(x, f(x)) \mid x \in \mathcal{X}\}$ intersect. ◀

*Proof.* Define the function $h\colon [0,1] \to [0,1]$ as the composition $h(t) = f(\alpha(t))$. The function $h$ is continuous as it is the composition of two continuous functions. By Brouwer's fixed-point theorem, there exists $t^* \in [0,1]$ such that $h(t^*) = t^*$, i.e., $f(\alpha(t^*)) = t^*$. As such, we have that $(\alpha(t^*), t^*) = (\alpha(t^*), f(\alpha(t^*))) \in \mathrm{gr}(\alpha) \cap \mathrm{gr}(f)$. ■

Lemma 4.24 provides the basis of our approach to surveillance, as any continuous barrier ($f\colon \mathcal{X} \to [0,1]$) across the space-time cylinder ($\mathcal{X}_{[0,1]} = \mathcal{X} \times [0,1]$) will intercept any possible intruder ($\alpha\colon [0,1] \to \mathcal{X}$). The surveillance task can be solved by constructing such a barrier through the robots' observations, specifying when each region of the mission space is observed, as we define now.

▶ **Definition 4.25 (Found and secure conditions).** An interpreted system $M$ satisfies *found and secure conditions* with mission space-time $\mathcal{X}_{[0,1]} = \mathcal{X} \times [0,1]$ if the following holds:

- *Detection agency*: For any global state $\mathbf{c} \in \mathbf{C}$, $\mathrm{FOUND} \in \ell(\mathbf{c})$ if and only if the underlying ontic state represents that an intruder has been detected in $\mathbf{c}$, or there exists $\mathbf{c}' \leq \mathbf{c}$ such that $\mathrm{FOUND} \in \ell(\mathbf{c}')$. Furthermore:

(a) *Persistence*: $M \models \text{FOUND} \Rightarrow \Box\text{FOUND}$, i.e., once an intruder is found, this fact does not change,

(b) *Distributed knowledge*: $M \models \text{FOUND} \Rightarrow D_{\mathcal{R}}\text{FOUND}$, i.e., the robots should collectively know when an intruder is found,

(c) *Individual knowledge*: $M \models \text{FOUND} \Rightarrow \bigvee_{\mathbf{r}\in\mathcal{R}} K_{\mathbf{r}}\text{FOUND}$, i.e., if an intruder is found, at least one robot knows it.

- *Observation correctness*: For any global state $\mathbf{c} \in \mathbf{C}$ and any region $W \subseteq \mathcal{X}_{[0,1]}$, if $\mathbf{c}$ is a state where the intruder's worldline intersects $W$, then the system cannot assert that the region is clear, i.e., $M, \mathbf{c} \not\models \mathsf{st}(\mathsf{W})$;
- *Securement*: For any continuous function $f : \mathcal{X} \to [0, 1]$ and any region of the mission space-time $W \subseteq \mathcal{X}_{[0,1]}$ such that $\mathsf{gr}(f) \subseteq W$, then $M \models \mathsf{st}(\mathsf{W}) \Rightarrow$ SECURE;
- *Completion*: $M \models \neg(\text{SECURE} \wedge \text{FOUND})$, i.e., the decisions are mutually exclusive.

◀

These conditions connect the intended behavior of a system in the surveillance task to its logical structure. Detection agency ensures that the FOUND proposition is only true if an intruder has been detected by a robot and that this fact is permanent. Observation correctness maintains consistency between the presence of an intruder and the robots' observations. Securement captures the idea of Lemma 4.24 as a barrier being sufficient to declare a space secure. Completion ensures a definitive outcome of either SECURE or FOUND and their relationship. Using the conditions above we can now define the meaning of the surveillance task in our model.

▶ **Definition 4.26 (Surveillance).** An interpreted system $M$ with mission space-time $\mathcal{X}_{[0,1]}$ *solves* the surveillance task if $M \models \Diamond!(\text{FOUND} \vee \text{SECURE})$. ◀

As with exploration, we distinguish three levels of coordination among the robots for termination. The cases represent again the knowledge of the system regarding the task: centralized (parallel termination), individual (selfish termination) and collective (cooperative termination). Note that now these conditions are

adapted for the binary outcome of surveillance, as either an intruder is found or the space is confirmed secure.

▶ **Definition 4.27 (Termination conditions for surveillance).** An interpreted system $M = (\mathcal{K}_{\mathbf{S}_{\mathcal{R},\mathcal{E}}}, \ell)$ in mission space-time $\mathcal{X}_{[0,1]}$ satisfies *termination* if one of the following conditions is satisfied:

- *Parallel termination* $M \models \Diamond! D_{\mathcal{R}}\text{FOUND} \vee \Diamond! D_{\mathcal{R}}\text{SECURE}$
- *Selfish termination* $M \models \Diamond! E_{\mathcal{R}}\text{FOUND} \vee \Diamond! E_{\mathcal{R}}\text{SECURE}$
- *Cooperative termination* $M \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{FOUND} \vee \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{SECURE}$

◀

We adapt now the liveness conditions to the geometric intuition from Lemma 4.24, constraining the robots to construct a temporal cut.

▶ **Definition 4.28 (Temporal cut liveness).** An interpreted system $M$ with mission space-time $\mathcal{X}_{[0,1]}$ satisfies *temporal cut liveness* if there exists a continuous function $f : \mathcal{X} \to [0,1]$ and a finite open cover $\mathcal{W} = \{W_1, \ldots, W_n\}$ with $\text{gr}(f) \subseteq \bigcup_{W_i \in \mathcal{W}} W_i$ such that $M \models \bigwedge_{W_i \in \mathcal{W}} \Diamond! \mathsf{st}(\mathsf{W_i})$. ◀

Under temporal cut liveness, every possible intruder is guaranteed to be caught by the temporal cut barrier eventually formed by the multi-robot system.

▶ **Theorem 4.29 (Temporal cut liveness is sufficient for surveillance).** Let $M$ be an interpreted system in mission space-time $\mathcal{X}_{[0,1]}$ satisfying the simple exploration conditions, the found and secure conditions, and temporal cut liveness. Then, $M$ satisfies surveillance with parallel termination, i.e., $M \models \Diamond! D_{\mathcal{R}}\text{FOUND} \vee \Diamond! D_{\mathcal{R}}\text{SECURE}$.

*Proof.* Consider an arbitrary global state $\mathbf{c}_0 \in \mathbf{C}$ and any sequence $\sigma = (\mathbf{c}_i)_{i \geq 0}$ from $\mathbf{c}_0$ with $\mathbf{c}_i \leq \mathbf{c}_{i+1}$. We proceed by analyzing the two possible cases of either an intruder is detected along $\sigma$ or it is not.

(i) The intruder is detected. Suppose there exists a global state $\mathbf{c}_j \in \sigma$ for $j \geq 0$ such that $M, \mathbf{c}_j \models \text{FOUND}$. By detection agency (Definition 4.25), we have that $M, \mathbf{c}_j \models D_{\mathcal{R}}\text{FOUND}$. As $\mathbf{c}_j \in \sigma$, we have that along the sequence $\sigma$ from $\mathbf{c}_0$ there exists a global state (namely $\mathbf{c}_j$) where $D_{\mathcal{R}}\text{FOUND}$ holds.

(ii) The intruder is not detected. Suppose that for all $i \geq 0$, $M, \mathbf{c}_i \not\models \text{FOUND}$. By temporal cut liveness (Definition 4.28), there exists a continuous function $f\colon \mathcal{X} \to [0,1]$ and a finite open cover $\mathcal{W} = \{W_1, \ldots, W_n\}$ with $\text{gr}(f) \subseteq \bigcup_{i=1}^{n} W_i$ such that $M \models \bigwedge_{i=1}^{n} \Diamond! W_i$. By the semantics of $\Diamond!$, this means that, for each $W_i \in \mathcal{W}$, there exists some $\mathbf{c}_{t_i} \in \sigma$ where $M, \mathbf{c}_{t_i} \models \mathsf{st}(\mathsf{W_i})$. Let $t_{\max} = \max\{t_i \mid i \in [1,n]\}$.

By the persistence of space-time propositions (Definition 4.14 applied to $\mathcal{X}_{[0,1]}$), once $\mathsf{st}(\mathsf{W_i})$ becomes true, it remains true. As such, $M, \mathbf{c}_{t_{\max}} \models \bigwedge_{i=1}^{n} \mathsf{st}(\mathsf{W_i})$. Let $W = \bigcup_{i=1}^{n} W_i$, where $W$ is the union from the temporal cut liveness cover. As space-time propositions are closed under union (Definition 4.23), we have that $\bigwedge_{i=1}^{n} \mathsf{st}(\mathsf{W_i}) \Rightarrow \mathsf{st}(\mathsf{W})$. As such, $M, \mathbf{c}_{t_{\max}} \models \mathsf{st}(\mathsf{W})$. By the space-time observation agency (from simple exploration conditions applied to the space-time $\mathcal{X}_{[0,1]}$), we have that $M \models \mathsf{st}(\mathsf{W}) \Rightarrow D_{\mathcal{R}}\mathsf{st}(\mathsf{W})$, and hence $M, \mathbf{c}_{t_{\max}} \models D_{\mathcal{R}}\mathsf{st}(\mathsf{W})$.

Since $W$ is defined to contain the temporal cut, i.e., $\text{gr}(f) \subseteq W$, which Lemma 4.24 shows to intersect any intruder, and by securement (Definition 4.25), we have that $M \models \mathsf{st}(\mathsf{W}) \Rightarrow \text{SECURE}$. As this holds at all global states, we have that $M \models D_{\mathcal{R}}(\mathsf{st}(\mathsf{W}) \Rightarrow \text{SECURE})$. By the K axiom, $D_{\mathcal{R}}(\mathsf{st}(\mathsf{W}) \Rightarrow \text{SECURE}) \Rightarrow (D_{\mathcal{R}}\mathsf{st}(\mathsf{W}) \Rightarrow D_{\mathcal{R}}\text{SECURE})$. Since $M, \mathbf{c}_{t_{\max}} \models D_{\mathcal{R}}\mathsf{st}(\mathsf{W})$, we obtain that $M, \mathbf{c}_{t_{\max}} \models D_{\mathcal{R}}\text{SECURE}$. This means that along $\sigma$ there exists a state where $D_{\mathcal{R}}\text{SECURE}$ holds.

In either case, along the arbitrary sequence $\sigma$ from the arbitrary global state $\mathbf{c}_0$, there exists a global state where $D_{\mathcal{R}}\text{FOUND} \vee D_{\mathcal{R}}\text{SECURE}$ holds. Since both $\sigma$ and $\mathbf{c}_0$ were arbitrary, by the semantics $\Diamond!$ (Definition 4.9), we have that $M \models \Diamond! D_{\mathcal{R}}\text{FOUND} \vee \Diamond! D_{\mathcal{R}}\text{SECURE}$. That is, $M$ satisfies parallel termination for surveillance. ■

While Theorem 4.29 guarantees that surveillance is solved, we show now that combined with the communication properties of exploration can again be used to ensure that all robots reach mutual knowledge of this information, enabling cooperative termination.

▶ **Theorem 4.30 (Sufficient conditions for cooperative surveillance).** Let $M$ be an interpreted system with mission space-time $\mathcal{X}_{[0,1]}$ satisfying the simple exploration

conditions, the found and secure conditions, temporal cut liveness, and communication liveness. Then, $M$ satisfies cooperative termination for surveillance, i.e., $M \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}} \text{FOUND} \vee \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}} \text{SECURE}$.

*Proof.* Note first that by Theorem 4.29, we have that $M \models \Diamond!(D_{\mathcal{R}}\text{FOUND} \vee D_{\mathcal{R}}\text{SECURE})$. Let $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \leq, \sim)$ be the system frame of $M$. Consider an arbitrary global state $\mathbf{c}_0 \in \mathbf{C}$ and any sequence $\sigma = (\mathbf{c}_i)_{i \geq 0}$ from $\mathbf{c}_0$ with $\mathbf{c}_i \leq \mathbf{c}_{i+1}$. By the semantics of $\Diamond!$, there exists $j \geq 0$ such that $M, \mathbf{c}_j \models D_{\mathcal{R}}\text{FOUND} \vee D_{\mathcal{R}}\text{SECURE}$. We consider each case now.

(i) $M, \mathbf{c}_j \models D_{\mathcal{R}}\text{FOUND}$. By individual knowledge from detection agency (Definition 4.25), we have that $M, \mathbf{c}_j \models \bigvee_{\mathbf{r} \in \mathcal{R}} K_{\mathbf{r}}\text{FOUND}$, meaning that for at least one robot $\mathbf{r}_i \in \mathcal{R}$, $M, \mathbf{c}_j \models K_{\mathbf{r}_i}\text{FOUND}$. By knowledge persistence (Theorem 4.13), since FOUND is a static atomic proposition (by persistence of detection), we have $M, \mathbf{c}_j \models \Box K_{\mathbf{r}_i}\text{FOUND}$. By Lemma 4.11, we obtain $M, \mathbf{c}_j \models \Diamond! K_{\mathbf{r}_i}\text{FOUND}$. By communication liveness (Definition 4.20), the information is guaranteed to spread, and we obtain $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}}\text{FOUND}$.

Consider any global state $\mathbf{c}' \in \mathbf{C}$ with $\mathbf{c}_j \sim_{\mathbf{r}_i} \mathbf{c}'$. Since $M, \mathbf{c}_j \models K_{\mathbf{r}_i}\text{FOUND}$, by the semantics of knowledge, we have $M, \mathbf{c}' \models \text{FOUND}$. By the individual knowledge from detection agency (Definition 4.25), we have that $M, \mathbf{c}' \models \bigvee_{\mathbf{r} \in \mathcal{R}} K_{\mathbf{r}}\text{FOUND}$. As such, there exists $\mathbf{r}_k \in \mathcal{R}$ with $M, \mathbf{c}' \models K_{\mathbf{r}_k}\text{FOUND}$. Again, by knowledge persistence (Theorem 4.13) and Lemma 4.11, $M, \mathbf{c}' \models \Diamond! K_{\mathbf{r}_k}\text{FOUND}$, and by communication liveness we have that $M, \mathbf{c}' \models \Diamond! E_{\mathcal{R}}\text{FOUND}$. Since this holds for all $\mathbf{c}'$ with $\mathbf{c}_j \sim_{\mathbf{r}_i} \mathbf{c}'$, by the semantics of knowledge, we obtain that $M, \mathbf{c}_j \models K_{\mathbf{r}_i} \Diamond! E_{\mathcal{R}}\text{FOUND}$.

Consider now any sequence $(\mathbf{c}_k)_{k \geq j}$. By $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}}\text{FOUND}$, we know there exists $\mathbf{c}_\ell$ in this sequence where $M, \mathbf{c}_\ell \models E_{\mathcal{R}}\text{FOUND}$. That is, for each $\mathbf{r} \in \mathcal{R}$, we have $M, \mathbf{c}_\ell \models K_{\mathbf{r}}\text{FOUND}$. By applying the same reasoning as before to each $\mathbf{r} \in \mathcal{R}$, using knowledge persistence and communication liveness, we obtain that $M, \mathbf{c}_\ell \models K_{\mathbf{r}} \Diamond! E_{\mathcal{R}}\text{FOUND}$ for all $\mathbf{r} \in \mathcal{R}$. By the definition of $E$ (Definition 4.7), this means $M, \mathbf{c}_\ell \models E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{FOUND}$. Since this holds for any sequence from $\mathbf{c}_j$, by the semantics of $\Diamond!$, we obtain that $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{FOUND}$.

(ii) $M, \mathbf{c}_j \models D_{\mathcal{R}}\text{SECURE}$. By exploration independence (Definition 4.14) applied to the mission space-time $\mathcal{X}_{[0,1]}$, we have that $M, \mathbf{c}_j \models D_{\mathcal{R}}\text{SECURE}$ implies the existence of a family $(W_{\mathbf{r}})_{\mathbf{r}\in\mathcal{R}}$ of regions with $\bigcup_{\mathbf{r}\in\mathcal{R}} W_{\mathbf{r}} \supseteq \mathsf{gr}(f)$, where $f$ is the continuous function from temporal cut liveness (Definition 4.28), such that $M, \mathbf{c}_j \models K_{\mathbf{r}}\mathsf{st}(\mathsf{W}_{\mathbf{r}})$ for each $\mathbf{r} \in \mathcal{R}$.

By the persistence of space-time propositions, from simple exploration conditions applied to $\mathcal{X}_{[0,1]}$ and knowledge persistence (Theorem 4.13), we have that $M, \mathbf{c}_j \models \Box K_{\mathbf{r}}\mathsf{st}(\mathsf{W}_{\mathbf{r}})$ for each $\mathbf{r} \in \mathcal{R}$. By Lemma 4.11, we obtain that $M, \mathbf{c}_j \models \Diamond! K_{\mathbf{r}}\mathsf{st}(\mathsf{W}_{\mathbf{r}})$. By communication liveness, we now have that $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}}\mathsf{st}(\mathsf{W}_{\mathbf{r}})$ for each $\mathbf{r} \in \mathcal{R}$.

Consider any sequence from $\mathbf{c}_j$. By the semantics of $\Diamond!$ and the persistence of space-time propositions, there exists $\mathbf{c}_m$ with $m \geq j$ in this sequence where $M, \mathbf{c}_m \models E_{\mathcal{R}}\mathsf{st}(\mathsf{W}_{\mathbf{r}})$ for all $\mathbf{r} \in \mathcal{R}$. Since space-time propositions are closed under union, we have that $\bigwedge_{\mathbf{r}\in\mathcal{R}} \mathsf{st}(\mathsf{W}_{\mathbf{r}}) \Rightarrow \mathsf{st}(\mathsf{W})$, where $W = \bigcup_{\mathbf{r}\in\mathcal{R}} W_{\mathbf{r}}$. By the K axiom applied to each $\mathbf{r} \in \mathcal{R}$ after opening $E_{\mathcal{R}}$ by its definition, we obtain that $M, \mathbf{c}_m \models E_{\mathcal{R}}\mathsf{st}(\mathsf{W})$. By securement (Definition 4.25), since $\mathsf{gr}(f) \subseteq W$, we have that $M \models \mathsf{st}(\mathsf{W}) \Rightarrow \text{SECURE}$. By the K axiom, $M, \mathbf{c}_m \models E_{\mathcal{R}}\text{SECURE}$. Using the same reasoning from case (i) with SECURE in the place of FOUND, noting that SECURE is static by the same reasons, we obtain $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{SECURE}$.

In either case, along the arbitrary sequence $\sigma$ from the arbitrary global state $\mathbf{c}_0$, we reach a state $\mathbf{c}_j$ where $M, \mathbf{c}_j \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{FOUND} \vee \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{SECURE}$. Since $\sigma$ and $\mathbf{c}_0$ were arbitrary, by the semantics of $\Diamond!$, we obtain $M \models \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{FOUND} \vee \Diamond! E_{\mathcal{R}} \Diamond! E_{\mathcal{R}}\text{SECURE}$. That is, $M$ satisfies cooperative termination for surveillance. ■

Theorem 4.30 shows that surveillance, similarly to exploration, can be reduced to verifying two independent properties: forming a temporal cut and enabling sufficient communication so that information is diffused. The geometric guarantee from Lemma 4.24 ensures that any continuous barrier through space-time must intersect the worldline of any intruder, where this barrier is created through the collective observations over time. The temporal cut liveness condition formalizes this requirement, while communication liveness ensures that the outcome,

whether FOUND or SECURE, eventually reaches all robots. Note that the found and secure conditions connect the detection of intruders by the physical robot to the logical framework, which in turn provides conditions for information obtained through their computations to be propagated until they cooperatively terminate.

### 4.3.3 GATHERING

So far, we analyzed robot tasks whose desired behavior demanded knowledge about the coverage of a space, either via the exploration of a region or its surveillance over time. We turn now to the gathering task, where we seek instead for the robots' positions in the mission space (or their estimations thereof) to converge to some gathering region. While the space is known in advance to all robots, the location of gathering must be coordinated throughout the mission. As such, gathering corresponds to the spatial counterpart to the consensus task in distributed computing, where robots express their agreement through their physical states instead of through communicating values.

We consider the approximate gathering relaxation of gathering, which demands that robots should eventually be contained within the same "small" region of the mission space, rather than an exact point convergence that would require infinite precision. This relaxation allows us to maintain the spatial language developed for exploration and surveillance, where atomic propositions are defined from regions of space.

▶ **Definition 4.31 (Position propositions).** Let $\mathcal{X}$ be a mission space and $\mathcal{R}$ a set of robots. For each open region $U \subseteq \mathcal{X}$ and robot $\mathbf{r} \in \mathcal{R}$, it is associated a *position proposition* $\mathsf{p}_{\mathbf{r}}(U)$ and an *initial position proposition* $\mathsf{i}_{\mathbf{r}}(U)$. The set of positional propositions is then $\mathrm{P}_{\mathcal{X}} = \{\mathsf{p}_{\mathbf{r}}(U), \mathsf{i}_{\mathbf{r}}(U) \mid \mathbf{r} \in \mathcal{R}, U \subseteq \mathcal{X} \text{ open}\}$. ◀

For a mission space $\mathcal{X}$ and open region $U \subseteq X$, we use propositions $\mathsf{p}_{\mathsf{r}}(U)$ to mark that a robot $\mathbf{r}$ is currently within $U$, a *dynamic fact* that contrasts with the static spatial propositions $\mathsf{s}(U)$, used to represent the fact that a region was forever explored. As validity conditions are often related to the initial positions of the robots, we also introduce static initial position propositions $\mathsf{i}_{\mathbf{r}}(U)$.

The language used for gathering is then the language of space with the extended set of positional propositions. The semantics of $\mathbf{L}_{gather}$ are the same as in

Definition 4.9, namely, that $M, \mathbf{c} \models p$ iff $p \in \ell(\mathbf{c})$ for any atomic proposition $p \in \mathsf{At}$. As such, it suffices to specify when the new positional propositions belong to the valuation of a global state $\mathbf{c}$.

$$\mathsf{p}_{\mathbf{r}}(\mathsf{U}) \in \ell(\mathbf{c}) \quad \text{iff} \quad o_{\mathbf{r}}(\mathbf{c}) \in U$$
$$\mathsf{i}_{\mathbf{r}}(\mathsf{U}) \in \ell(\mathbf{c}) \quad \text{iff} \quad o_{\mathbf{r}}(\mathbf{c}_{\mathbf{I}}) \in U$$

where $\mathbf{c}_{\mathbf{I}} \in \mathbf{I}$ is the unique initial state in the execution containing $\mathbf{c}$, and $o_{\mathbf{r}}(\mathbf{c})$ is the ontic position of robot $\mathbf{r}$ extracted from the ontic component of $\mathbf{c}$ (recall the compositional multi-robot system from Section 3.3).

The gathering task is then reduced to finding a gathering region $U$ in the mission space such that eventually all robots simultaneously satisfy $\mathsf{p}_{\mathbf{r}}(\mathsf{U})$ and remain there forever.

▶ **Definition 4.32 (Approximate gathering).** An interpreted system $M$ in mission space $\mathcal{X}$ is consistent with *approximate gathering* if there exists a collection of *gathering regions* $\mathsf{gather}(\mathcal{X}) \subseteq \{U \subseteq \mathcal{X} \mid U \text{ is open}\}$ such that:

1. $\bigcup_{U \in \mathsf{gather}(\mathcal{X})} U = \mathcal{X}$, i.e., they cover the mission space,
2. for all $U \neq V \in \mathsf{gather}(\mathcal{X})$, $U \cap V = \varnothing$, i.e., a partition is formed,
3. each $U \in \mathsf{gather}(\mathcal{X})$ has bounded diameter,

and the following is satisfied.

- *Eventual gathering*: $M \models \bigvee_{U \in \mathsf{gather}(\mathcal{X})} \bigwedge_{\mathbf{r} \in \mathcal{R}} \Diamond! \Box \mathsf{p}_{\mathbf{r}}(\mathsf{U})$, i.e., in every sequence of global states, there exists a region $U \in \mathsf{gather}(\mathcal{X})$ such that all robots inevitably meet there forever;
- *Starting validity*: $M \models \bigwedge_{U \in \mathsf{gather}(\mathcal{X})} \left( \bigwedge_{\mathbf{r} \in \mathcal{R}} \mathsf{i}_{\mathbf{r}}(\mathsf{U}) \Rightarrow \bigwedge_{\mathbf{r} \in \mathcal{R}} \Diamond! \Box \mathsf{p}_{\mathbf{r}}(\mathsf{U}) \right)$, i.e., if all robots start in the same gathering region, they should end in that region.

◀

▶ **Remark 4.33 (Gathering regions).** For a mission space $\mathcal{X} = (\mathcal{X}, \tau)$, the collection of gathering regions $\mathsf{gather}(\mathcal{X})$ defines what constitutes "close enough" for a gathering task. Different choices provide different definitions of gathering, which we abstract away. For example, $\mathsf{gather}(\mathcal{X}) = \{B_\varepsilon(x) \mid x \in \mathcal{X}\}$ recovers the classical

$\varepsilon$-gathering. Note that the bounded diameter property make the approximation physically meaningful, ensuring that the gathering regions are within finite distance. ◀

We now exploit the known similarities between distributed computing tasks and robot tasks, as reviewed in Section 2.2.3. Following the work of Alcántara et al. [Alc+19], where the borders for impossibility of exact gathering were derived from the analogous consensus problem, we reduce approximate gathering to the stabilizing agreement problem. In doing so, we provide a novel logical perspective to this correspondence.

▶ **Definition 4.34 (Stabilizing agreement, adapted from [CFR24]).** Let $\mathcal{V}$ be a set of possible decision values and $\Pi$ a set of agents[3]. Let $\mathsf{choose}_a(v)$ denote agent $a$'s *decision* to stabilize on value $v$, and $\mathsf{init}_b(v)$ denote that $v$ was agent $b$'s *initial value*. An interpreted system $M$ satisfies *stabilizing agreement* if the following are satisfied.

- *Agreement*: $M \models \bigvee_{v\in\mathcal{V}} \bigwedge_{a\in\Pi} \Diamond!\Box \mathsf{choose}_a(v)$, i.e., there is a value $v \in \mathcal{V}$ such that every agent decides on that value,
- *Validity*: $M \models \bigwedge_{v\in\mathcal{V}} \bigwedge_{a\in\Pi} \Big(\mathsf{choose}_a(v) \Rightarrow K_a(\bigvee_{b\in\Pi} \mathsf{init}_b(v))\Big)$, i.e., an agent $a \in \Pi$ can only choose a known initial value of some agent.

◀

The correspondence between approximate gathering and stabilizing agreement requires translating between their respective logical frameworks, as the former employs system frames with branching-time logic while the latter uses the runs and systems framework with a linear-time logic.

▶ **Remark 4.35 (Translating from runs and systems to system frames).** The stabilizing agreement from Cignarale et al. [CFR24] uses the runs and systems framework [Fag+95], where a Linear Temporal Logic quantifies over individual execution traces, while our framework with system frames employs branching-time operators over all possible executions (Definition 4.9). The translation is straightforward,

[3] We identify $\Pi$ with $\mathcal{R}$ in the robot setting.

as each maximal sequence of global states connected by the causality relation ($\leq$) of a system frame (Definition 4.1) corresponds to one execution trace in a runs and systems frame.

The temporal operators translate as follows. LTL's always operator $\Box\varphi$, signifying that $\varphi$ holds at all times in the respective trace, corresponds to our invariance operator $\Box\varphi$, which requires for $\varphi$ to hold in all future states along all paths. LTL's eventually operator $\Diamond\varphi$, interpreted as $\varphi$ holding at some time in the trace, corresponds to our liveness operator $\Diamond!\varphi$, which analogously requires $\varphi$ to eventually hold in every branching. For the stabilization property, LTL's $\Diamond\Box\varphi$ ("eventually always" $\varphi$), translates to our $\Diamond!\Box\varphi$, expressing that along every possible execution, the system eventually reaches and maintains $\varphi$ as true. This captures the AGREEMENT condition in Definition 4.34. ◀

By viewing each gathering region as a decision value, position propositions as *choices*, and initial positions to initial values, we establish that approximate gathering corresponds to a spatial variant of the stabilizing agreement problem. The physical position of the robots encode the implicit decisions for a gathering location, where the spatial convergence manifests the distributed agreement on a value. As such, the epistemic conditions for stabilizing agreement, shown in Cignarale et al. [CFR24], translate directly to sufficient conditions for approximate gathering.

▶ **Theorem 4.36 (Stabilizing agreement implies approximate gathering).** Let $M$ be an interpreted system in mission space $\mathcal{X}$ that satisfies stabilizing agreement over the set of gathering regions $V = \mathsf{gather}(\mathcal{X})$, with

$$\mathsf{choose}_{\mathbf{r}}(U) \iff \mathsf{p}_{\mathbf{r}}(U) \qquad \text{and} \qquad \mathsf{init}_{\mathbf{r}}(U) \iff \mathsf{i}_{\mathbf{r}}(U),$$

for each robot $\mathbf{r} \in \mathcal{R}$ and region $U \in \mathsf{gather}(\mathcal{X})$. Then $M$ satisfies approximate gathering.

*Proof.* We identify the set of robots $\mathcal{R}$ with the set of agents $\Pi$, and the set of gathering regions $\mathsf{gather}(\mathcal{X})$ with the decision values $\mathcal{V}$, from stabilizing agreement (Definition 4.34). We proceed by showing that the stabilizing agreement properties imply the approximate gathering properties (Definition 4.32) under the correspondence $\mathsf{choose}_{\mathbf{r}}(U) \iff \mathsf{p}_{\mathbf{r}}(U)$ and $\mathsf{init}_{\mathbf{r}}(U) \iff \mathsf{i}_{\mathbf{r}}(U)$.

(i) **Agreement implies eventual gathering**. Assume that $M$ satisfies the agreement property, i.e., $M \models \bigvee_{v \in \mathcal{V}} \bigwedge_{a \in \Pi} \Diamond! \Box \mathsf{choose}_{\mathsf{a}}(\mathsf{v})$. By the correspondence, this becomes

$$M \models \bigvee_{U \in \mathsf{gather}(\mathcal{X})} \bigwedge_{\mathbf{r} \in \mathcal{R}} \Diamond! \Box \mathsf{p}_{\mathbf{r}}(\mathsf{U}),$$

which is precisely the eventual gathering property of Definition 4.32.

(ii) **Validity implies starting validity**. Assume that $M$ satisfies the validity property, i.e., $M \models \bigwedge_{v \in \mathcal{V}} \bigwedge_{a \in \Pi} \left(\mathsf{choose}_{\mathsf{a}}(\mathsf{v}) \Rightarrow K_a \left(\bigvee_{b \in \Pi} \mathsf{init}_{\mathsf{b}}(\mathsf{v})\right)\right)$. Under our correspondence, this states that if a robot $\mathbf{r}$ is in a region $U$, then $\mathbf{r}$ knows that some robot started in $U$. Consider a gathering region $U^*$ and suppose that $\bigwedge_{\mathbf{r} \in \mathcal{R}} \mathsf{i}_{\mathbf{r}}(\mathsf{U}^*)$, meaning that robot $\mathbf{r}$ started at $U^*$. By agreement, there exists $U^{**} \in \mathsf{gather}(\mathcal{X})$ such that $M \models \bigwedge_{\mathbf{r} \in \mathcal{R}} \Diamond! \Box \mathsf{p}_{\mathbf{r}}(\mathsf{U}^{**})$. Consider now any robot $\mathbf{r}$ and global state $\mathbf{c}$ where $M, \mathbf{c} \models \mathsf{p}_{\mathbf{r}}(\mathsf{U}^{**})$ holds. By validity, we have that $M, \mathbf{c} \models K_{\mathbf{r}} \left(\bigvee_{\mathbf{r}' \in \mathcal{R}} \mathsf{init}_{\mathbf{r}'}(\mathsf{U}^{**})\right)$. By the T axiom, this implies that $M, \mathbf{c} \models \bigvee_{\mathbf{r}' \in \mathcal{R}} \mathsf{init}_{\mathbf{r}'}(\mathsf{U}^{**})$, so there exists some robot $\mathbf{r}'$ such that $M, \mathbf{c} \models \mathsf{init}_{\mathbf{r}'}(\mathsf{U}^{**})$. That is, that $\mathbf{r}'$ started in $U^{**}$.

Since by our initial assumption all robots started in $U^*$, we have $M, \mathbf{c} \models \mathsf{init}_{\mathbf{r}'}(\mathsf{U}^*)$ for this same $\mathbf{r}'$, as initial position propositions are static facts and hold at all subsequent global states. As gathering regions partition the space (Definition 4.32), we have that either $U^* \cap U^{**} = \varnothing$ or $U^* = U^{**}$. Since both $\mathsf{init}_{\mathbf{r}'}(\mathsf{U}^*)$ and $\mathsf{init}_{\mathbf{r}'}([]U^{**}]$ hold, the partition property forces that $U^* = U^{**}$. As such, the robots stabilize in their initial gathering regions. Since the choice of $U^*$ was arbitrary, we obtain

$$M \models \bigwedge_{U \in \mathsf{gather}(\mathcal{X})} \left( \bigwedge_{\mathbf{r} \in \mathcal{R}} \mathsf{i}_{\mathbf{r}}(\mathsf{U}) \Rightarrow \bigwedge_{\mathbf{r} \in \mathcal{R}} \Diamond! \Box \mathsf{p}_{\mathbf{r}}(\mathsf{U}) \right)$$

which is the starting validity property.

■

Theorem 4.36 establishes that spatial coordination is an agreement problem where decision values are regions of space and choices are physical positions[4]. The

[4]This sort of reduction will be explored again in Section 6.4, as synthesized in Remark 6.39.

main distinction between approximate gathering and the more abstract stabilizing agreement is the physical constraint, where gathering regions must be physically realizable (Assumption 3.35) by the robot dynamics and the environment constraints from Chapter 3. These geometrical and dynamical feasibility conditions, while absent in pure distributed computing where agents can often decide arbitrary values, constrain which system frames are valid models of the physical system. The logical framework then reasons about the coordination requirements over these frames that encode what is computationally observable by the robots, separating the ontic modeling and epistemic reasoning steps, while maintaining the latter consistent with the former.

## 4.4 CONCLUSION

This chapter established a temporal-epistemic logical framework for reasoning about knowledge in distributed multi-robot systems. System frames are introduced as a geometric representation of knowledge evolution in the system, extending the runs and systems framework of Fagin et al. [Fag+95] with explicit causality relations and indistinguishability relations that are parametrized by groups of robots. We use this structure to model the fact that constraints in the environment (e.g., regarding visibility, asynchronicity, dynamics, etc.) manifest as uncertainty in the possible robots' epistemic states.

The temporal-epistemic logic $\mathbf{L}_{space}$ with spatial propositions provided a formal language to specify properties of multi-robot systems. As we derive the atomic propositions from the topology of the mission space, we connect the physical behavior of the robots to their computational reasoning. From our modeling choices in Chapter 3, and the assumption of perfect recall, we showed the persistence of spatial knowledge (Theorem 4.13), which establishes the monotonic growth of knowledge, a property that is fundamental for guaranteeing progress in distributed tasks.

We describe and derive sufficient epistemic conditions to different robot tasks. For exploration, we showed that exploration liveness and communication liveness imply together cooperative termination (Theorem 4.21), separating the physical coverage from the distributed coordination problem. For surveillance, we showed that a notion of temporal cut captures the geometrical intuition of exploring a

mission space over time, and that the property of temporal cut liveness alongside the previous communication properties guarantees that a multi-robot system can cooperatively decide whether the space is secure or not (Theorem 4.30). For gathering, we provide a correspondence between the robotic and distributed computing analogous problems by showing that the sufficient epistemic conditions for solving stabilizing agreement are also sufficient for approximate gathering, reducing spatial coordination to a variation of consensus with physically realizable values (Theorem 4.36).

We note that the current framework assumes perfect recall, and treats crashed agents as permanently inactive, limiting the applications to systems susceptible to faults and heterogeneous behavior. An interesting generalization is to incorporate partial equivalence relations, as in Goubault et al. [Gou+23], enabling reasoning about non-standard group knowledge with faults. Furthermore, our communication model abstracts away message complexity, as communication liveness assumes the eventual propagation of information without considering constraints such as delays and message size.

The taxonomy of multi-robot systems remains largely unexplored from the knowledge perspective. Physical robotic systems introduce restrictions absent in distributed computing, such as dynamical constraints and spatial embedding problems that modify how knowledge evolves. The analysis of concrete system models under varying types of failures, communication protocols and dynamical constraints may reveal fundamental limitations from the interaction of physical and computational components.

The system frame and its representation of knowledge through indistinguishability relations provides the epistemic structure for Chapter 5, which employs it as the parametrizing space for the topological constraints imposed by distributed tasks. Chapter 6 focuses on the iterative construction of structures that capture similar epistemic information as system frames, and develops the categorical machinery to relate different representations of distributed knowledge in multi-robot systems.

*"La géométrie n'est pas vraie, elle est avantageuse."*

"Geometry is not true, it is advantageous."

— Henri Poincaré, La science et l'hypothèse [Poi02]

# 5

# CELLULAR SHEAVES OF DISTRIBUTED TASKS

We build now on the knowledge encoding of system frames from Chapter 4, whose global states parametrized spatial propositions for evaluating epistemic and temporal formulas. We explore the locality of this knowledge structure: decisions are computed based on local epistemic states, which must be consistent across all indistinguishable global states. At the same time, the validity of a decision function is evaluated at every reachable global state. This tension between local decision constraints and global task requirements is central to distributed computing, famously reconciled through topological methods modeling local views as simplices with connectivity arguments revealing global impossibilities [HKR14]. This chapter formalizes the insight that distributed coordination is solvable precisely when local constraints can be simultaneously satisfied, a requirement that is naturally expressed in the theory of cellular sheaves [Cur14].

A task sheaf is constructed to identify, within its structure, the specific terminating decision functions that solve a given task, namely, those satisfying determinism, termination and task satisfaction. The first two properties are encoded structurally into a system slice, a finite substructure of the system frame built by iteratively identifying the global states that must be causally consistent

with an arbitrary decision frontier. The latter property is ensured by restricting the space of possible decision functions to contain only valid assignments for each global state in the slice. This construction allows us to decompose the problem into agent task sheaves (Theorem 5.29), where valid decision functions correspond precisely to the sections, objects witnessing the simultaneous satisfaction of all consistency constraints (Theorem 5.33). We then show that the existence of such sections, and hence of solutions, is semidecidable, with a constructive procedure that outputs a terminating decision function when one exists (Theorem 5.44). This result demonstrates the utility of sheaves for reasoning about distributed systems, as tools such as cohomology become available to effectively reduce the task solvability question to linear algebra over the integers.

This chapter corresponds to the contents of [FHR25b] and of its extended version [FHR25a].

**Related work** Topological methods for distributed computing, reviewed in Chapter 2, have been central since the Asynchronous Computability Theorem of Herlihy and Shavit [HS93], which characterized wait-free solvability through simplicial connectivity. This framework was systematically developed in [HKR14], and later extended to general models by Attiya, Castañeda and Nowak [ACN23], to continuous task specifications by Galeana, Rajsbaum and Schmid [RRS22], and to network specific models by Fraigniaud and Paz [FP23]. A point-set topological approach was developed by Nowak, Schmid and Winkler [NSW24], characterizing consensus on topologies of infinite executions, where the distance between executions is measured by how long processes take to distinguish them. An epistemic perspective on task solvability was established by Goubault, Ledent and Rajsbaum [GLR21], connecting simplicial models to Kripke structures, and extended to chromatic semi-simplicial sets by Goubault et al. [Gou+23], as discussed in Chapters 4 and 6. We retain the topological perspective of these works, but develop a local characterization, replacing the existence of continuous maps between complexes with the requirement that decisions at individual global states remain valid when extended to the entire distributed state space.

The theory of cellular sheaves was largely extended by Curry [Cur14] as a combinatorial reformulation of classical sheaf theory on cellular complexes, with direct usage in computational topology. Hansen [Han20] developed a spectral

theory of cellular sheaves, lifting the graph Laplacian to model and analyze diffusion and consensus over networks. This line of work has found applications in opinion dynamics [HG21], distributed optimization [HG19a], and machine learning [Bod+23]. This approach has been recently expanded to multi-agent coordination by Hanks et al. [Han+25] and shown compatible with more relaxed forms of data by Ghrist et al. [Ghr+25]. Robinson [Rob17; Rob20] applied similar concepts to sensor fusion and signal processing, where sections are interpreted are consistent data across heterogeneous nodes in a network. In these works, the base space is a communication graph and stalks contain working agent data. Our construction differs in both base space and data, as we use an epistemic structure derived from a system frame rather than an interaction network, and data are possible decision functions constrained by the task specification. This approach aligns closer to the usage of Abramsky and Brandenburger [AB11] for contextuality in quantum systems, where obstructions to global sections encode fundamental limitations on consistent global assignments.

**Chapter organization** Section 5.1 builds a system slice (Definition 5.13), a finite (Lemma 5.9) and epistemically complete (Theorem 5.10) substructure of system frames. It encodes the consistency constraints (Definition 5.11) of a valid terminating decision function, forming a cell category (Definition 5.17) later used to define a cellular sheaf.

Section 5.2 motivates the solution data of a task as being the space of decision functions (Definition 5.20), parametrized by the constraints of a system slice. Together, they define a task sheaf (Definition 5.23), a structure encoding the precise relationship between valid decisions and global states of a system, which is later shown isomorphic (Theorem 5.29) to the composition of agent-wise sheaves (Definition 5.27).

Section 5.3 defines the sections of a task sheaf (Definition 5.30), the sheaf-theoretic counterpart to terminating decision functions (Lemma 5.31). Those are thoroughly illustrated in Example 5.32, and followed by the main characterization result corresponding task solvability to the existence of sections in a system slice (Theorem 5.33).

Section 5.4 establishes a first application of the bridge between distributed systems and sheaf theory by linearizing the task sheaf (Definition 5.35) and defin-

ing its zeroth cohomology group (Definition 5.39). A direct usage of the standard identification of the zeroth cohomology with the space of sections gives us a characterization of task solvability in terms of this cohomology (Corollary 5.42). Example 5.43 illustrates how this correspondence partially reduces reasoning about tasks as a problem of linear algebra. Finally, the finiteness and correspondence results lead to a semidecidable procedure for constructing a terminating decision function when one exists (Theorem 5.44).

## 5.1 SYSTEM SLICES AND LOCAL CONSISTENCY

We formalize through cellular sheaves how the epistemic structure of system frames parametrizes topological task data, making explicit the passage from local constraints to global properties. The main advantage of this perspective will be a local definition of terminating task solvability, defined in Section 5.2, besides the compositional and computational properties explored thereafter. More precisely, we will build locally to each global state a solution to the problem of determining whether a terminating decision function exists. Recall from Chapter 2 that a terminating decision function is a deterministic function from local states to decision values that satisfies simultaneously termination and the task specification. In the context of system frames, this is instantiated as follows.

▶ **Remark 5.1 (Terminology).** We adopt now the term *agent* to encompass both the robots from Chapters 3 and 4 and purely computational processes. The framework developed in this chapter applies to general distributed systems with epistemic constraints on local and global states, such as with system frames. ◀

▶ **Remark 5.2 (Task solvability in system frames).** We instantiate the notion of task solvability from Definition 2.16 to system frames with explicit termination as follows. For task $\mathbf{T} = (I, O, \Delta)$, we extend $O$ by adding all combinations of *decisions* with the extra value $\perp$, without adding any valid relation to those in $\Delta$. These represent explicitly[1] the trivial choice of postponing the decision, while making such choices invalid for satisfying the task specification. For a system

[1]This will facilitate the definition of task data in Section 5.2.

frame $\mathcal{K}_{\mathbf{S}} = (\mathbf{C}, \sim, \leq)$, a function $d$ mapping local states to output values in $O \cup \{\perp\}$ *solves* $\mathbf{T}$ if it satisfies for every agent $a \in \mathcal{A}$ the following:

- *Determinism*: The decision of $a$ is the same for all identical local states, i.e., if $\mathbf{c} \sim_a \mathbf{c}'$, then $d(\pi_a(\mathbf{c}')) = d(\pi_a(\mathbf{c}))$;
- *Termination*: The decision of $a$ is final once made, remaining the same in all causally related states, i.e., if $\mathbf{c} \leq_a \mathbf{c}'$ and $d(\pi_a(\mathbf{c})) \neq \perp$, then $d(\pi_a(\mathbf{c}')) = d(\pi_a(\mathbf{c}))$;
- *Task satisfaction*: Every execution is eventually consistent with the task specification, i.e., for every sequence $(\mathbf{c}_i)_{i\geq 0}$ from some $\mathbf{c}_0 \in I$ such that $\mathbf{c}_i \leq \mathbf{c}_{i+1}$, there exists some $\mathbf{c}_j$ where $d(\pi_a(\mathbf{c}_j)) \neq \perp$ for all $a \in \mathcal{A}$, the tuple $(d(\pi_a(\mathbf{c}_j)))_{a\in\mathcal{A}} \in O$, and $(\mathbf{c}_0, (d(\pi_a(\mathbf{c}_j)))_{a\in\mathcal{A}}) \in \Delta$.

Note that the decision functions are identical for all agents, and assemble piecewise into the typical decision function $\delta$ of Definition 2.16. ◀

In order to define such a sheaf, recall from Section 2.3 that a sheaf defined on a graph assigns some data to each of its vertices and edges (its cells), with coherent maps restricting data from vertices to edges. For distributed tasks, we will derive a graph-like structure from system frames, where global states serve as 0-cells, parametrizing the terminating decision functions later assigned to them, and 1-cells encode consistency constraints from the epistemic structure. An immediate usage of a system frame is not possible, however, as it presents two obstacles to forming a well-defined base space. First, system frames may contain an unbounded number of states in non-terminating protocols. While cellular sheaves can be defined on infinite spaces, computational methods such as through cohomology in Section 5.4 require a finite structure. Second, system frames have two relations per agent, that of causality ($\leq$) and indistinguishability ($\sim$), which need to be converted to a single symmetric relation for defining the restriction maps. Both obstacles are solved through the construction of system slices, a finite substructure of system frames with a single causal consistency relation ($\simeq$) that preserves task solvability. We describe now the procedure of extracting a system slice from a system frame.

A finite substructure of the system frame is obtained from a subset of its global states creating a decision frontier, alongside its related states. An execution cut captures this frontier.

▶ **Definition 5.3 (Execution cut).** Let $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$ be a system frame. An *execution cut* is a set $A \subseteq \mathbf{C}$ such that every execution, i.e., maximal sequence of causally related global states, intersects $A$. ◀

An execution cut describes a frontier through which every execution must pass, and it is called *terminal* when all agents have decided values that satisfy the task specification. Recall that a solving protocol reaches a terminal global state in every execution within a finite number of steps (see Definitions 4.15, 4.26 and 4.32), constituting a terminal execution cut by definition.

▶ **Lemma 5.4 (Existence of terminal execution cuts).** If a task $\mathbf{T}$ is solvable for system $\mathbf{S}$, then there exists a terminal execution cut in the corresponding system frame $\mathcal{K}_{\mathbf{S}}$. ◀

*Proof.* Let $\mathbf{T} = (I, O, \Delta)$ be a task, and $d$ be a terminating decision function. By the definition of solution, every execution sequence of global states $\sigma = (\mathbf{c}_i)_{i\geq 0}$, from some $\mathbf{c}_0 \in \mathbf{C}$ such that $\mathbf{c}_i \leq \mathbf{c}_{i+1}$ has some state $\mathbf{c}_\sigma = (e_1, \ldots, e_n)$ where $\{d(e_i)\}_{1\leq i\leq n}$ satisfies $\Delta$. We construct an execution cut $A = \{\mathbf{c}_\sigma \mid \sigma \text{ is a possible execution}\}$, formed by those states that satisfy the task described above, which exist by assumption of terminating solvability. This set intersects every execution by construction and satisfies $\Delta$ at every $\mathbf{c} \in A$, making the cut terminal. ■

Observe that the choice of an execution cut for analysis is not unique. When decisions are final, as with terminating decision functions that preserve the same output after some decision point, any execution cut that is causally consequent of a terminal execution cut is also terminal.

▶ **Lemma 5.5 (Persistence of terminal execution cuts).** Let $\mathcal{K}_{\mathbf{S}}$ be system frame and $d$ be a terminating decision function. If $A$ is a terminal execution cut in $\mathcal{K}_{\mathbf{S}}$, then any execution cut $A'$ that is causally after $A$, i.e., for every $\mathbf{c}' \in A'$ there exists $\mathbf{c} \in A$ such that $\mathbf{c} \leq \mathbf{c}'$, $A'$ is also terminal. ◀

*Proof.* Consider the system frame $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}}$. Let $A$ be a terminal execution cut of $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}}$, and let $A'$ be an execution cut such that every global state $\mathbf{c}' \in A'$ has a causal

antecedent $\mathbf{c} \in A$ satisfying $\mathbf{c} \leq \mathbf{c}'$. Since $A$ is terminal, every $\mathbf{c} = \{e_a\}_{a\in\mathcal{A}} \in A$ is assigned final decisions $d(e_a)$. That is, for every agent $a \in \mathcal{A}$ and every global state $\mathbf{c}_k = \{e'_a\}_{a\in\mathcal{A}}$ such that $\mathbf{c} \leq_a \mathbf{c}_k$, all decisions satisfy $d(e'_a) = d(e_a)$. As the decision of every $a \in \mathcal{A}$ is simultaneously final in $\mathbf{c}$, we have that any state $\mathbf{c} \leq \mathbf{c}_k$ satisfies $d(e'_a) = d(e_a)$, regardless by which agent the transition is causally related. As we assumed that the execution cut $A'$ is entirely composed of such $\mathbf{c}_k$ causally related to $\mathbf{c} \in A$, they are assigned the same final decisions as in $\mathbf{c}$. In other words, every global state $\mathbf{c}' \in A'$ satisfies the task, making $A'$ a terminal execution cut. ■

▶ **Remark 5.6 (Choice of execution cut).** A consequence of Lemma 5.5 is that, for a system with expected final decisions, we do not need to identify an exact earliest execution cut, as any cut following a terminal one will suffice. This creates an evident trade-off, where earlier cuts contain fewer global states, making it less costly to verify, while later cuts are more likely to be terminal, when it occurs. The choice is based on the available resources, as a sufficiently late execution cut will be terminal by Lemma 5.4.

When not considering terminating decision functions, or system guarantees such as perfect recall (Assumption 3.19), later cuts might not be terminal even if earlier ones are, as the agents may recompute their outputs with the different information available. The problem of identifying an optimal cut, i.e., the earliest point with the fewest global states, is a separate one from that of task solvability, and we do not address here. ◀

Note now that an execution cut contains too little information about the system's execution for a candidate decision function to be meaningfully verified to respect determinism and termination. Recall, first, that any assignment of a decision value to a local state must be identical across all global states where it appears. For a global state within an execution cut $\mathbf{c} \in A$, if another global state $\mathbf{c}' \notin A$ is such that $\mathbf{c} \sim_a \mathbf{c}'$, then agent $a$ has the same local state in both $\mathbf{c}$ and $\mathbf{c}'$ and must have the same assignment. Moreover, decisions must be final, which requires them to remain identical throughout causally related global states. Those may go beyond the indistinguishable states, as any identifiable notion of progress (as with perfect recall) will create a distinct local epistemic state. Consequently, we must check both indistinguishable global states to the ones in a cut, and the global states accessible from those through causal sequences, to verify all properties of a

terminating decision function. The causal closure formalizes this extension of the execution cut.

▶ **Definition 5.7 (Causal closure).** Let $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$ be a system frame and $A \subseteq \mathbf{C}$ an execution cut. The *causal closure* of $A$ for agent $a \in \mathcal{A}$, denoted $\mathrm{ccl}_a(A)$, is the smallest subset of $\mathbf{C}$ such that:

- $A \subseteq \mathrm{ccl}_a(A)$,
- if $\mathbf{c} \in \mathrm{ccl}_a(A)$ and $\mathbf{c} \sim_a \mathbf{c}' \leq_a \mathbf{c}'' \leq \mathbf{c}_A$, for some $\mathbf{c}', \mathbf{c}'' \in \mathbf{C}$ and $\mathbf{c}_A \in A$, then $\mathbf{c}', \mathbf{c}'' \in \mathrm{ccl}_a(A)$.

The causal closure of $A$ over all agents in $\mathcal{A}$, denoted $\mathrm{ccl}(A)$, is defined as

$$\mathrm{ccl}(A) = \bigcup_{a \in \mathcal{A}} \mathrm{ccl}_a(A).$$

◀

▶ **Remark 5.8 (Computing causal closure).** For a system frame $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$, the causal closure set $\mathrm{ccl}(A)$ for the execution cut $A \subseteq \mathbf{C}$ can be computed via a fixed-point iteration. Let the initial value be the cut, i.e., $A_0 = A$, then apply the following

$$A_{n+1} = A_n \cup \{\mathbf{c}_e, \mathbf{c}_c \in \mathbf{C} \mid \exists a \in \mathcal{A}, \exists \mathbf{c} \in A_n : \mathbf{c} \sim_a \mathbf{c}_e \leq_a \mathbf{c}_c\}$$

and iterate until $A_n = A_{n+1}$, at which moment $\mathrm{ccl}(A) = A_n$. That is, for every global state in the execution cut, first find all indistinguishable states. Those must have identical decisions for they have identical local states. Then, find all causally related states. Those must have identical decisions for them to be final. This process is then repeated for the enlarged set of global states as new constraints must be added for the same reasons. This procedure is done until no more states are added, stopping at the execution cut itself, which by definition intersects any execution that may be reached during the procedure. ◀

▶ **Lemma 5.9 (Finiteness of causal closure).** Let $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$ be a system frame that is locally finite, i.e., for every agent $a \in \mathcal{A}$ each global state $\mathbf{c} \in \mathbf{C}$ is related by $\sim_a$ and $\leq_a$ to finitely many other global states. Then, for any execution cut $A \subseteq \mathbf{C}$, the causal closure $\mathrm{ccl}(A)$ is finite. ◀

*Proof.* The causal closure iteratively adds global states backwards from $A$ via indistinguishability and causality relations. Since the cut happens at a finite depth in each execution sequence, going backward explores finitely many prior global states. Each iteration step adds finitely many global states by local finiteness and finiteness of $\mathcal{A}$. As such, the set $\mathrm{ccl}(A)$ is finite. ■

The causal closure provides a local space in which all requirements of a terminating decision function can be verified. Given a choice of execution cut limiting the scope of the task solvability analysis, it suffices to check a candidate decision function on the global states of the causal closure, rather than on the entire system frame until the cut.

▶ **Theorem 5.10 (Constraint locality).** Let $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$ be a system frame, let $\mathbf{T} = (I, O, \Delta)$ be a task, and let $A \subseteq \mathbf{C}$ be a terminal execution cut with causal closure $\mathrm{ccl}(A)$. If $d$ is a function from local states appearing in $\mathrm{ccl}(A)$ to $O \cup \{\bot\}$ such that:

1. $d$ satisfies determinism and termination on the subframe induced by $\mathrm{ccl}(A)$;
2. For every $\mathbf{c} \in A$, the pair $(\mathbf{c}_0, (d(\pi_a(\mathbf{c})))_{a\in\mathcal{A}}) \in \Delta$, where $\mathbf{c}_0 \in I$ is the initial state of the execution sequence leading to $\mathbf{c}$.

Then, $d$ extends to a terminating decision function solving $\mathbf{T}$ in $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}}$.

*Proof.* We extend $d$ to all local states in the system frame by assigning $\bot$ to any state not appearing in $\mathrm{ccl}(A)$, and denote this new function $\overline{d}$. We proceed by verifying that $\overline{d}$ satisfies determinism, termination and task satisfaction, as per Remark 5.2.

- Determinism: Consider any pair of global states $\mathbf{c}, \mathbf{c}' \in \mathbf{C}$ such that $\mathbf{c} \sim_a \mathbf{c}'$ for some agent $a \in \mathcal{A}$, i.e., $\pi_a(\mathbf{c}) = \pi_a(\mathbf{c}')$. If both $\mathbf{c}, \mathbf{c}' \in \mathrm{ccl}(A)$, then, by assumption of $d$ satisfying determinism within the causal closure, $\overline{d}(\pi_a(\mathbf{c})) = d(\pi_a(\mathbf{c}'))$ assigns an identical value. If $\mathbf{c}, \mathbf{c}' \notin \mathrm{ccl}(A)$, then $\overline{d}$ assigns $\bot$ to both, satisfying determinism. By the construction of the causal closure (Definition 5.7), if $\mathbf{c} \in \mathrm{ccl}(A)$ and $\mathbf{c} \sim_a \mathbf{c}'$, then $\mathbf{c}' \in \mathrm{ccl}(A)$, and vice-versa. As such, it cannot be the case that only one of the two global states lies in $\mathrm{ccl}(A)$.

- Termination: Consider any pair of global states $\mathbf{c}, \mathbf{c}' \in \mathbf{C}$ such that $\overline{d}(\pi_a(\mathbf{c})) \neq \bot$ and $\mathbf{c} \leq_a \mathbf{c}'$ for some agent $a \in \mathcal{A}$. By the definition of the extension $\overline{d}$, $\mathbf{c} \in \mathrm{ccl}(A)$, as otherwise it would have been assigned $\bot$. By the definition of causal closure, if $\mathbf{c} \in \mathrm{ccl}(A)$ and $\mathbf{c} \leq_a \mathbf{c}'$, then, by reflexivity $\mathbf{c} \sim_a \mathbf{c}$, we have that $\mathbf{c}' \in \mathrm{ccl}(A)$. As both $\mathbf{c}, \mathbf{c}' \in \mathrm{ccl}(A)$, by the assumption of termination holding for $d$ within the causal closure, we have that $\overline{d}(\pi_a(\mathbf{c}')) = \overline{d}(\pi_a(\mathbf{c}))$, satisfying termination.
- Task satisfaction: Consider any execution sequence $\sigma = (\mathbf{c}_i)_{i \geq 0}$ starting from some $\mathbf{c}_0 \in I$. By the definition of execution cut, this sequence intersects $A$ at some $\mathbf{c}_j \in A$. By our assumption that every global state at the cut satisfies the task specification $\Delta$, we have that $(\mathbf{c}_0, (d(\pi_a(\mathbf{c}_j)))_{a \in \mathcal{A}}) \in \Delta$, therefore as also have that $(\mathbf{c}_0, (\overline{d}(\pi_a(\mathbf{c}_j)))_{a \in \mathcal{A}}) \in \Delta$, as the values are the same within $\mathrm{ccl}(A)$. By Lemma 5.5, since $A$ is terminal by assumption, decisions on $\mathbf{c}_j$ are final and any global state $\mathbf{c}_k \in \sigma$ with $k \geq j$ maintains these decisions. As the execution sequence was arbitrary, we have that every execution from an initial state verifies the task satisfaction.

■

Theorem 5.10 reduces verifying task solvability to checking for the compatibility of candidate terminating decision functions with determinism, termination and task satisfaction within the subframe induced by $\mathrm{ccl}(A)$, rather than on the arbitrarily large system frame. The problem of verification is then reduced in size following a choice of execution cut, as noted in Remark 5.6.

We convert now the subframe induced by the causal closure of some execution cut to a set of uniform constraints to be used as the base space of our sheaf in Section 5.2. Note that both the termination and determinism properties translate to equality constraints over pairs of global states under specific consistency requirements, derived from the expected behavior of a terminating decision function. As such, we merge these into a single equivalence relation per agent, forming equivalence classes of global states where the agent must assign identical decisions.

▶ **Definition 5.11 (Causal consistency relation).** Let $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$ be a system frame and $A \subseteq \mathbf{C}$ an execution cut. For each agent $a \in \mathcal{A}$, the *causal consistency* relation

$\simeq_a \subseteq \mathbf{C} \times \mathbf{C}$ is the symmetric-transitive closure of

$$(\sim_a \cap(\mathrm{ccl}(A) \times \mathrm{ccl}(A))) \cup (\leq_a \cap(\mathrm{ccl}_a(A) \times (\mathrm{ccl}_a(A))))$$

◀

The causal consistency relation is the smallest equivalence relation on $\mathrm{ccl}(A)$ containing the agent-wise constraints. Note that determinism requires any pair of indistinguishable global states to have the same decision, while termination only applies to those reachable by the same agent from a candidate final decision in the execution cut.

▶ **Lemma 5.12 (Causal consistency is an equivalence relation).** Let $A$ be an execution cut on system frame $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}}$ with causal closure $\mathrm{ccl}(A)$. For each agent $a \in \mathcal{A}$, the causal consistency relation $\simeq_a$ is an equivalence relation on $\mathrm{ccl}(A)$. ◀

*Proof.* Consider agent $a \in \mathcal{A}$. By construction, $\simeq_a$ is the smallest set containing the transitive and symmetric relations from $\sim_a$ on $\mathrm{ccl}(A)$ and $\leq_a$ on $\mathrm{ccl}_a(A)$. As such, $\simeq_a$ is transitive and symmetric. For reflexivity, note that $\sim_a$ is reflexive by Definition 4.1, i.e., for any $\mathbf{c} \in \mathrm{ccl}(A)$, we have $\mathbf{c} \sim_a \mathbf{c}$. As $\sim_a \cap (\mathrm{ccl}(A) \times \mathrm{ccl}(A)) \subseteq \simeq_a$ by construction (Definition 5.11), the causal consistency relation is also reflexive. ■

We can now transform the potentially unbounded epistemic structure of system frames into a finite and uniform constraint space. The closure operation extracts finitely many global states relevant to verifying a terminating decision function, while the new causal consistency relation encode the equality constraints imposed to each agent. The resulting structure is what we call a system slice.

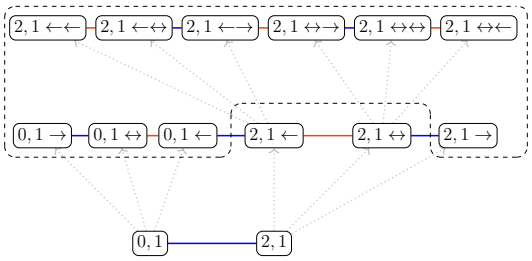
▶ **Definition 5.13 (System slice).** Let $\mathcal{K}_{\mathbf{S}_{\mathcal{A},\mathcal{E}}} = (\mathbf{C}, \sim, \leq)$ be a system frame and $A \subseteq \mathbf{C}$ be an execution cut. The *system slice* over $A$ is the pair $\mathcal{S}_A = (\mathrm{ccl}(A), \{\simeq_a\}_{a\in\mathcal{A}})$, where $\mathrm{ccl}(A)$ is the causal closure of cut $A$, and $\simeq_a$ is the causal consistency relation for each agent $a \in \mathcal{A}$. ◀

The system slice structure exposes how the two separate notions of indistinguishability and causality impose a unified set of consistency constraints. This uniformity shows the local nature of distributed coordination, both across agents,

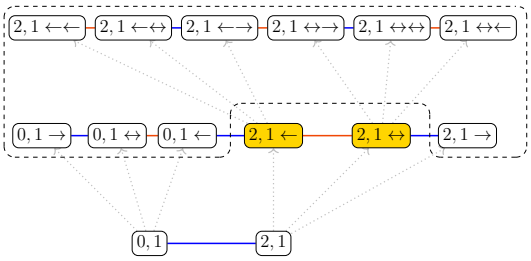
as each $\simeq_a$ forms independent equivalence classes, and across the state-space, as consistency can be checked by looking pairs of global states related by $\simeq_a$. As such, verifying a candidate terminating decision function reduces to checking that it assigns identical values within each equivalence class, without the need to consider the entire input and output spaces of the task. Note that $\mathrm{ccl}(A)$ is finite by Lemma 5.9, making the system slice a finite structure. The transformation from frames to slices makes explicit that global solvability is entirely determined by local constraints, the idea that we explore for the remaining of the chapter.

▶ **Example 5.14 (Causal closure and system slice).** We illustrate now the iterative construction of a system slice $\mathcal{S}_A$ from a system frame $\mathcal{K}_S$ and a choice of execution cut $A$, as described in Remark 5.8. The construction depicted in Figure 5.1, where indistinguishable global states are connected by solid blue and red lines for each agent, and the causal relations are the dotted gray arrows. The dashed blue lines are the newly added causal consistency relations, alongside the previous indistinguishability ones. From the choice of an execution cut $A$ (Figure 5.1a), which must intersect every possible execution sequence of $\mathcal{K}_S$, the set of global states in the causal closure is first expanded by indistinguishability relations (Figure 5.1b) and then by causality relations (Figure 5.1c).
Recall from Remark 5.6 that assessing whether a terminating decision function solves a task requires working from a terminal execution cut. In this scenario, the global states accessible from the cut via an indistinguishability relation are then partially terminated, meaning that at least one agent has identical local state. For determinism, its decision must be identical. Similarly, the states subsequently accessible via causality relations, by the same agent, must have identical decisions in order to satisfy termination. ◀

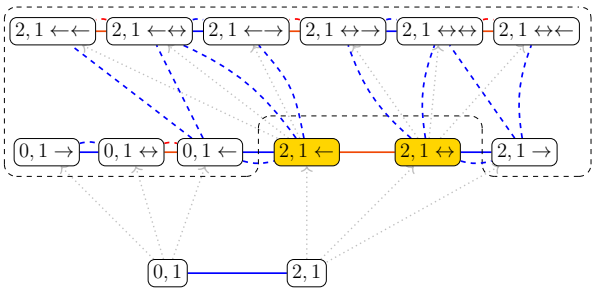
We can now define the base space needed to encode task solvability constraints in a cellular sheaf. That is, we derive from the system slice a cell category that will serve as base space for a sheaf of decision functions in Section 5.2. Recall from Section 2.3.4 that Curry [Cur14] defines cellular sheaves as functors from the face incidence poset of a regular cellular complex (Definition 2.37), viewed as a category, to a data category. We apply this construction to system slices by viewing them as multigraphs (1-dimensional regular cellular complexes) and deriving the associated poset.




(a) Selection of a terminal execution cut $A$.




(b) Expansion via indistinguishability $\sim$.




(c) Final expansion via causality $\leq$ resulting in the slice $\mathcal{S}$.

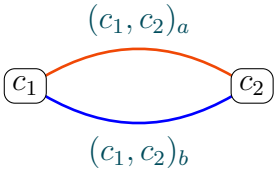
Figure 5.1
Fixed-point iteration transforming a system frame subset into a valid system slice.

▶ **Remark 5.15 (System slices as multigraphs and their structure).** A system slice $\mathcal{S} = (\mathrm{ccl}(A), \{\simeq_a\}_{a\in\mathcal{A}})$ viewed as an undirected multigraph has global states as vertices and causal consistency relations as edges. This induces a *face incidence poset* $P_{\mathcal{S}}$ analogous to Example 2.38. By writing each pair of related states $(\mathbf{c}, \mathbf{c}') \in \simeq_a$ as $(\mathbf{c}, \mathbf{c}')_a$, the poset $P_{\mathcal{S}} = (\mathrm{ccl}(A) \cup \{\simeq_a\}_{a\in\mathcal{A}}, \trianglelefteq)$ has one element for each global state $\mathbf{c} \in \mathrm{ccl}(A)$ and each edge $(\mathbf{c}, \mathbf{c}')_a \in \{\simeq_a\}_{a\in\mathcal{A}}$. The partial ordering $\trianglelefteq$ relates global state $\mathbf{c}$ to the edge $(\mathbf{c}, \mathbf{c})_a$, written $\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a$, whenever $\mathbf{c}$ is a face of $(\mathbf{c}, \mathbf{c}')_a$.

The system slice $\mathcal{S}$ together with its face incidence poset $P_{\mathcal{S}}$ forms a regular cellular complex, as per Definition 2.37. As it is regular, its poset $P_{\mathcal{S}}$ entirely represents its combinatorial structure and lets us forget the original space. Note that if $(\mathbf{c}, \mathbf{c}') \in \simeq_a$ and $(\mathbf{c}, \mathbf{c}') \in \simeq_{a'}$ for distinct agent $a \neq a'$, then $(\mathbf{c}, \mathbf{c}')_a \neq (\mathbf{c}, \mathbf{c}')_{a'}$ are distinct elements in $P_{\mathcal{S}}$, preserving the independence of consistency constraints

for each agent. ◀

▶ **Example 5.16 (Cellular complex from a multigraph).** Consider a multigraph with vertices $V = \{c_1, c_2\}$ and edges $E = \{(c_1, c_2)_a, (c_1, c_2)_b\}$ between the same pair of vertices. The face incidence poset is then $P = (\{c_1, c_2, (c_1, c_2)_a, (c_1, c_2)_b\}, \trianglelefteq)$, where

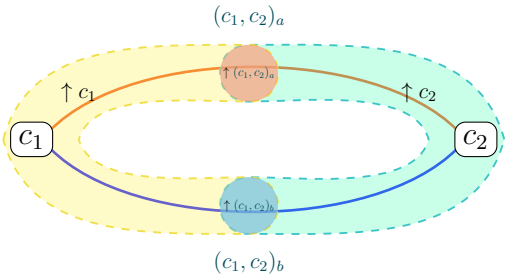



$\trianglelefteq$ contains the following relations:

$$\trianglelefteq := \{(c_1, (c_1, c_2)_a), (c_1, (c_1, c_2)_b), (c_2, (c_1, c_2)_a), (c_2, (c_1, c_2)_b)\}$$

The underlying topological space of the regular cellular complex[2] is recovered by taking the upset of each element of $P$. The cells below make evident how the face relation $\trianglelefteq$ encodes the (contravariant) inclusion of cells within each other.

$$\uparrow c_1 = \{c_1, (c_1, c_2)_a, (c_1, c_2)_b\}$$
$$\uparrow c_2 = \{c_2, (c_1, c_2)_a, (c_1, c_2)_b\}$$
$$\uparrow (c_1, c_2)_a = \{(c_1, c_2)_a\}$$
$$\uparrow (c_1, c_2)_b = \{(c_1, c_2)_b\}$$




In the case of a system slice, the corresponding topological space depicts, for each global state, an open set containing its neighborhood. That is, the state itself and the causal consistency relations attached to it, without the other end points. ◀

This face incidence poset can be seen as a category, where the partial order $\trianglelefteq$ defines its morphisms. This is the construction of the cell category we will use for the remainder of the chapter.

▶ **Definition 5.17 (Cell category from a system slice).** Let $\mathcal{S} = (\mathrm{ccl}(A), \{\simeq_a\}_{a \in \mathcal{A}})$ be a system slice from an execution cut $A$, and let $P_{\mathcal{S}}$ be its face incidence poset. The *cell category* $\mathbf{Cell}(\mathcal{S})$ is defined by the following data:

[2] This is in fact the Alexandrov topology. More details on its properties and construction can be found in [Cur14, Chapter 4].

- Objects: the elements of $P_{\mathcal{S}}$, i.e., the global states $\mathbf{c} \in \mathrm{ccl}(A)$ (0-cells) and edges $(\mathbf{c}, \mathbf{c}')_a$ where $(\mathbf{c}, \mathbf{c}') \in \simeq_a$ (1-cells);
- Morphisms: a unique map $\delta : \mathbf{c} \to (\mathbf{c}, \mathbf{c}')_a$ for each relation $(\mathbf{c}, (\mathbf{c}, \mathbf{c}')_a) \in \trianglelefteq$ in $P_{\mathcal{S}}$.

Composition and identity are given by the transitivity and reflexivity of the partial order $\trianglelefteq$ in $P_{\mathcal{S}}$. ◀

The cell category extends the construction of Example 2.38 to multigraphs with agent-labeled edges, where the labels distinguish the independent sets of constraints per agent. These will also translate in Section 5.2 to an agent-wise decomposition in the sheaf encoding of the problem. Note that the underlying poset structure ensure that the cellular sheaves defined on a category obtained by Definition 5.17 satisfy the sheaf axioms, as [Cur14, Theorem 4.2.10] establishes the equivalence between sheaves on a poset and functors from that poset viewed as a category. This cell category captures consistency constraints enforced by the epistemic structure of the original system frame, making it the appropriate setting for characterizing task solvability in the following.

## 5.2 TASK SHEAVES

Recall that our objective is to give a local definition of terminating task solvability. So far, we defined a cell category induced by a system slice as encoding the set of constraints any solving terminating decision function must satisfy. In this section, we show that the possible decision functions define a collection of maps from the set of agents to the set of permissible decision values, and that those maps assemble into a cellular task sheaf, a functor attaching decision data to each cell. The restriction maps in this sheaf enforce the agent-wise consistency requirements, making global solvability a consequence of local compatibility.

The topological task specification (Definition 2.24), when generalized from chromatic simplicial complexes to chromatic semi-simplicial sets[3], abbreviated

[3]The usage of csets as a distributed state space enables a fully algebraic depiction of distributed tasks in Chapter 6, while here we explore its topological properties. This generalization of the task specification to csets is thoroughly developed in Example 6.22, alongside its associated task solvability condition.

as csets (Definition 2.35), provides the most adapted setting for our task data. Csets are instances of cellular complexes and have been shown in Goubault et al. [Gou+23] to have similar interpretation as simplicial complexes for distributed systems.

▶ **Definition 5.18 (Task specification with csets).** Let $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \Delta)$ be a task specification (Definition 2.24) with $\mathcal{I}, \mathcal{O} \colon \Gamma^{\mathrm{op}} \to \mathbf{Set}$ as csets (Definition 2.35). The *task specification* sub-cset is $\mathcal{T} \subseteq \mathcal{I} \times \mathcal{O}$, forming a span relation $\mathcal{I} \xleftarrow{\pi_{\mathcal{I}}} \mathcal{T} \xrightarrow{\pi_{\mathcal{O}}} \mathcal{O}$ projecting to the inputs and outputs of the task. For each $U \in \Gamma$ and input values $i \in \mathcal{I}(U)$, the valid outputs are those $o \in \mathcal{O}(U)$ such that $(i, o) \in \mathcal{T}(U)$. ◀

Recall that for each $U \in \Gamma$, the set $\mathcal{I}(U)$ contains the $U$-simplices representing configurations of the agents in $U$, and face maps $\partial_{U,V} \colon \mathcal{I}(V) \to \mathcal{I}(U)$ extract subconfigurations by forgetting agents in $V \setminus U$.

By assuming that the agents preserve access to its initial values, such as with perfect recall[4], an input projection function makes explicit the relationship between global states in a system frame and admissible outputs.

▶ **Definition 5.19 (Input projection).** Let $\mathbf{c} = (s_1, \ldots, s_n)$ be a global state with agents $U_{\mathbf{c}} \in \Gamma$ and $\mathbf{C}$ a set of global states. Let each local state satisfy perfect recall (Assumption 3.19), that is, $s_i = (s_i^0, s_i^1, \ldots, s_i^k)$ for agent $a_i$ at step $k$, where $s_i^0$ is its initial state. The *input projection* is a function $\mathrm{i} \colon \mathbf{C} \to \coprod_{U \in \Gamma} \mathcal{I}(U)$, given by $\mathrm{i}(\mathbf{c}) = (s_1^0, \ldots, s_n^0) \in \mathcal{I}(U_{\mathbf{c}})$. ◀

The projection function in Definition 5.19 recovers the input values of a global state, interpreted as a $U$-simplex in the input cset. This allows us to select, for each global state, only those output values that the task specification permits for the given inputs (Definition 5.18). The selection is done by taking the fiber of the task specification over the input simplex, which extracts the sub-cset of $\mathcal{T}$ with pairs whose input component matches $\mathrm{i}(\mathbf{c})$.

We can now define the space of decision functions that will serve as data for our sheaf. The decision functions at a global state, when expressed through csets, are maps from sets of agents to sets of valid outputs, parametrized by the input

[4] Note that any input preserving mechanism is sufficient for such a construction, but perfect recall is the most immediate as already established in Chapter 3.

values at that state. The output component of the fiber of the task specification will provide the codomain for such functions.

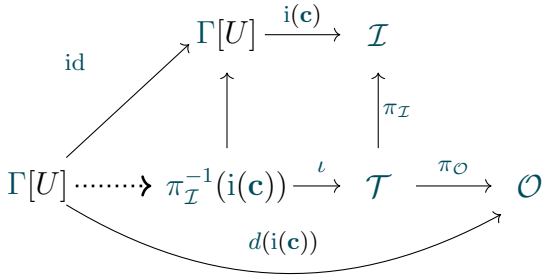
▶ **Definition 5.20 (Space of decision functions).** Let $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \mathcal{T})$ be a task and let $\mathbf{c}$ be a global state with input $\mathrm{i}(\mathbf{c}) \in \mathcal{I}(U)$ for a set of agents $U \in \Gamma$. The *fiber* of $\pi_{\mathcal{I}} \colon \mathcal{T} \to \mathcal{I}$ over $\mathrm{i}(\mathbf{c})$ is the cset $\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c})) \subseteq \mathcal{T}$ defined by the following pullback in $\mathbf{Cset}$.

$$\begin{array}{ccc} \Gamma[U] & \xrightarrow{\mathrm{i}(\mathbf{c})} & \mathcal{I} \\ \uparrow & \lrcorner & \uparrow{\scriptstyle \pi_{\mathcal{I}}} \\ \pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c})) & \xrightarrow{\iota} & \mathcal{T} \end{array}$$

where $\Gamma[U]$ is the representable cset. Explicitly, for any $V \in \Gamma$, the $V$-simplices are

$$\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c}))(V) = \{\sigma \in \mathcal{T}(V) \mid \pi_{\mathcal{I}}(\sigma) = \partial_{V,U}(\mathrm{i}(\mathbf{c}))\}$$

A *decision function* at $\mathbf{c}$ is a morphism of csets $d \colon \Gamma[U] \to \mathcal{O}$ such that the following diagram commutes.




The *space of decision functions* at $\mathbf{c}$ is the external hom

$$\mathcal{D}(\mathbf{c}) = \hom_{\mathbf{Cset}}(\Gamma[U_{\mathbf{c}}], \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c}))))$$

◀

▶ **Remark 5.21 (Decision functions via csets).** Note that space of decision functions $\mathcal{D}(\mathbf{c})$ generalizes the definition of a decision function beyond the maps from local states to output values of Remark 5.2. Here, a decision function assigns output values to all agents simultaneously at a global state $\mathbf{c}$, encoded as a morphisms $d \colon \Gamma[U] \to \mathcal{O}$

in the category of csets. The properties of determinism and termination are no longer internal to the function, but instead externalized to the structural constraints imposed by the system slice. This shift further exposes the combinatorial structure of distributed coordination, transforming functional requirements into topological compatibility conditions, verified through the simplicial structure. ◀

Through Definition 5.20, we enforce the task satisfaction property of Remark 5.2, complementing the termination and determinism encoded in the system slice. While the space of decision functions construction may appear elaborate, the functorial machinery is essential for enforcing task constraints through the simplicial structure of csets. More precisely, the fiber captures the precise sub-cset of $\mathcal{T}$ selecting only the pairs of input and output values allowed at the global state, an information that would be ignored in more general definitions of decision functions. The decision functions are then morphisms to the output component of this fiber.

While conceptually immediate to the objective, the external hom representation can be simplified through the Yoneda lemma and the externalization of properties into the simplicial structure discussed in Remark 5.21. That is, the Yoneda lemma establishes that morphisms from $\Gamma[U]$ to any cset correspond bijectively to elements of that cset evaluated at $U$. In our setting, this means replacing tracking decision functions (morphisms to the fiber) with tracking directly decision values ($U$-simplices within the fiber).

▶ **Lemma 5.22 (Decision function representability).** Let $\mathbf{c}$ be a global state with agents $U \in \Gamma$, and let $i = \mathrm{i}(\mathbf{c})$. There is a natural isomorphism

$$\mathcal{D}(\mathbf{c}) \cong \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(i))(U)$$

between the space of decision functions and the set of $U$-simplices in the fiber. ◀

*Proof.* By the Yoneda lemma, there is a natural isomorphism $\hom_{\mathbf{Cset}}(\Gamma[U], X) \cong X(U)$ for any cset $X$, given by evaluating the simplices at $U$. Applying it to $X = \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(i))$, we obtain $\mathcal{D}(\mathbf{c}) = \hom_{\mathbf{Cset}}(\Gamma[U], \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(i))) \cong \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(i))(U)$.

By Definition 5.20, $\pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(i))(U) = \pi_{\mathcal{O}}(\{\sigma \in \mathcal{T}(U) \mid \pi_{\mathcal{I}}(\sigma) = i\})$. This set is formed by the pairs of $\mathcal{T}(U)$, and so the projection retrieves $\{o \in \mathcal{O}(U) \mid (i, o) \in \mathcal{T}(U)\}$, which are the valid output values for input $i$. ■

The isomorphism in Lemma 5.22 uses the Yoneda lemma to replace tracking decision functions as morphisms with tracking decision values as simplices. A $U$-simplex $o \in \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c})))(U)$ represents a tuple of $(o_a)_{a\in U}$ of output values for all agents in $U$, satisfying the task specification for the input $\mathrm{i}(\mathbf{c})$. This representation makes explicit that decisions at a global state are tuples of agent outputs, not individual agent functions.

The attribution of decisions to each global state in a system slice is captured by a functor from the cell category to the category of csets, defining a cellular sheaf. This sheaf formalizes the relationship between constraints imposed by the distributed system definition, encoded in the cell structure, and the constraints imposed by the task, encoded within the construction of the space of decision functions.

▶ **Definition 5.23 (Task sheaf).** Let $\mathcal{S} = (\mathrm{ccl}(A), \{\simeq_a\}_{a\in\mathcal{A}})$ be a system slice with cell category $\mathbf{Cell}(\mathcal{S})$, and let $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \mathcal{T})$ be a task. The *task sheaf* is the functor

$$\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}.$$

defined as follows. Recall that the objects of $\mathbf{Cell}(\mathcal{S})$ are graded into $0$-cells (from global states $\mathbf{c} \in \mathrm{ccl}(A)$) and $1$-cells (from causal consistency relations $(\mathbf{c}, \mathbf{c}')_a \in \{\simeq_a\}_{a\in\mathcal{A}}$).

- **On $0$-cells.** For each $0$-cell $\mathbf{c} \in \mathbf{Cell}(\mathcal{S})$, the *stalk* is the cset of valid decisions for input $\mathrm{i}(\mathbf{c})$, i.e.,
$$\mathcal{F}(\mathbf{c}) = \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c})))$$
- **On $1$-cells.** For each $1$-cell $(\mathbf{c}, \mathbf{c}')_a \in \mathbf{Cell}(\mathcal{S})$, the *stalk* is the cset of valid decisions for agent $a$, i.e.,
$$\mathcal{F}((\mathbf{c}, \mathbf{c}')_a) = \partial_{\mathcal{A},\{a\}}(\mathcal{O})$$
- **On face maps.** For each face map $\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a$ in $\mathbf{Cell}(\mathcal{S})$, we assign the *restriction map* from $U$ agents to agent $a$, i.e.,
$$\mathcal{F}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a) = \partial_{U,\{a\}} \colon \mathcal{F}(\mathbf{c}) \to \mathcal{F}((\mathbf{c}, \mathbf{c}')_a)$$

◀

Definition 5.23 joins the individual concerns of task validity and local consistency. The stalks at 0-cells encode the task specification, as they contain only the decision values that are allowed from the specific set of inputs leading to the global state. The stalks at 1-cells acts as the maximal decision space for an agent that is free to decide alone, only regarding its own options. The restriction maps will allow us to enforce in Section 5.3 equality constraints of the causal consistency, where decisions must be identical above both endpoints of a 1-cell. The task sheaf structure captures the problem of agents needing to locally agree on a decision value despite having different information about the global state.

▶ **Remark 5.24 (Stalk size).** Note that stalk $\mathcal{F}((\mathbf{c},\mathbf{c}')_a)$, at the 1-cell, is defined as the full set of possible output values for agent, which will be in general much larger than subset of valid values on the adjacent global states. A more restrictive definition would be to consider only the union of $a$-simplices in the stalks over each global state. The main advantage of our chosen approach is to have the stalks uniformly defined, while still respecting the task constraints. ◀

▶ **Lemma 5.25 (The task sheaf is well-defined).** The task sheaf $\mathcal{F}\colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ of Definition 5.23 is a well-defined functor. ◀

*Proof.* We proceed by showing that $\mathcal{F}$ consistently assigns objects and morphisms in $\mathbf{Cset}$ respecting functoriality.

**Objects.** For a 0-cell $\mathbf{c}$, the stalk $\mathcal{F}(\mathbf{c}) = \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c})))$ is a cset by Definition 5.20. For a 1-cell $(\mathbf{c},\mathbf{c}')_a$, the stalk $\mathcal{F}((\mathbf{c},\mathbf{c}')_a) = \partial_{\mathcal{A},\{a\}}(\mathcal{O})$ is a cset by direct application of face maps to $\mathcal{O}$.

**Morphisms.** Let $\mathbf{c} \trianglelefteq (\mathbf{c},\mathbf{c}')_a$ be a face map in $\mathbf{Cell}(\mathcal{S})$, with the task sheaf assignment of the restriction map $\partial_{U,\{a\}}\colon \mathcal{F}(\mathbf{c}) \to \mathcal{F}((\mathbf{c},\mathbf{c}')_a)$. Since $\mathbf{c} \trianglelefteq (\mathbf{c},\mathbf{c}')_a$, we have that $a \in U$. This defines a natural transformation for any $W \subseteq V$ in $\Gamma$, and so we have $\partial_{W,\{a\}} \circ \partial_{V,W} = \partial_{V,\{a\}}$ by composition of face maps.

**Identities.** For any object $\alpha \in \mathbf{Cell}(\mathcal{S})$, the identity morphism $\mathrm{id}\alpha\colon \alpha \to \alpha$ corresponds to the reflexivity of $\trianglelefteq$ in the relation $\alpha \trianglelefteq \alpha$. The task sheaf $\mathcal{F}(\mathrm{id}\alpha) = \partial_{W,W}$ for the appropriate $W$ within $\alpha$, which is the identity natural transformation, as $\partial_{W,W}$ is the identity function on each set of simplices.

**Composition**. The cell category $\mathbf{Cell}(\mathcal{S})$ is a poset of height $1$, i.e., its only non-identity morphisms are from $0$-cells to $1$-cells. As such, there are no composable pairs of distinct non-identity morphisms. Composition with identities is preserved by the identity preservation above.

■

▶ **Example 5.26 (Task sheaf encoding binary consensus).** We illustrate the task sheaf construction for the binary consensus task after one round of a lossy link synchronous message adversary. Consider two agents $\mathcal{A} = \{a, b\}$ with binary input values $\{0, 1\}$. Binary consensus requires both agents to decide on a common value appearing among the inputs. The task is defined by the input cset $\mathcal{I}$ containing the $1$-simplices $\{(0,0),(0,1),(1,0),(1,1)\}$, the output cset $\mathcal{O}$ with $1$-simplices $\{(0,0),(1,1)\}$, and $\mathcal{T} \subseteq \mathcal{I} \times \mathcal{O}$ relating the valid pairs.

The *lossy link* message adversary allows each agent to send its value to the other at each round, where at most one may not be delivered. We denote global states as $(v_a, v_b\mathrm{com})$, where $v_a, v_b \in \{0,1\}$ are the initial values and $\mathrm{com} \in \{\leftarrow, \leftrightarrow, \rightarrow\}$ indicates the delivery pattern. Recall that indistinguishability relations associate pairs of global states with identical local states. For instance, agent $a$ cannot distinguish between $(0, 1 \leftarrow)$ and $(0, 1 \leftrightarrow)$, as both provide the local state $s_a = (0, (1_b))$.

Consider now system slice depicted in Figure 5.2, covering the possible global states after one step of executions from the input values. The task sheaf $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ assigns data as follows.

- Stalks at $0$-cells are the valid output simplices for the inputs at the corresponding global state. At $(0_a, 1_b, \leftrightarrow)$ both values appear, hence $\mathcal{F}((0_a, 1_b \leftrightarrow))$ is the cset with values $(0_a, 0_b)$ and $(1_a, 1_b)$. At $(1_a, 1_b, \leftrightarrow)$, only $1$ is allowed, i.e., the cset of $(1_a, 1_b)$.
- Stalks at $1$-cells are all decision values for non-distinguishing agent, i.e., $\{0, 1\}$.
- Restriction maps are the face maps $\partial_{\{a,b\},\{a\}}$ to the relevant agent's component, defined on the stalks.

◀

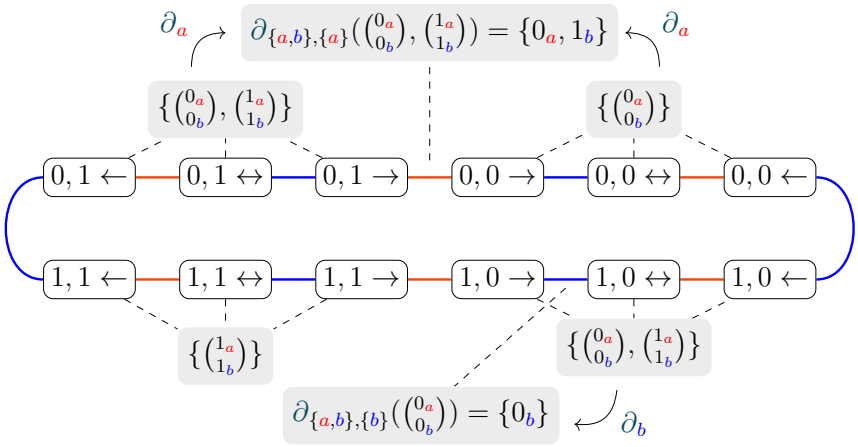



Figure 5.2
The task sheaf for binary consensus after one round from input values $(0_a, 0_b)$ and $(0_a, 1_b)$. Causal consistency relations are derived only from indistinguishability relations, and are identified by red lines for agent $a$ and blue for $b$.

We can see now in Example 5.26 how the constraints encoded by the task sheaf arise from the communication uncertainty. When an agent receives a message, it cannot determine whether its own message was also delivered. Conversely, when no message arrives, the adversary specification implies that the outgoing message was delivered. This asymmetry creates the indistinguishability relations that are carried over to the system slice, and forces identical decisions in both scenarios, which is enforced by the restriction maps derived from those relations.

We develop now the modularity of the system slice in view of the possible agent-wise definition of the constraints. The independence of the causal consistency relations imply that the task sheaf constructed in Definition 5.23, may in fact be decomposed into agent task sheaves, each a functor isolating the perspective of a single agent.

▶ **Definition 5.27 (Agent task sheaf).** Let $\mathcal{S} = (\mathrm{ccl}(A)), \{\simeq_a\}_{a\in\mathcal{A}})$ be a system slice and $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \mathcal{T})$ a task. For each agent $a \in \mathcal{A}$, let $\mathbf{Cell}_a(\mathcal{S})$ be the subcategory of $\mathbf{Cell}(\mathcal{S})$ containing all 0-cells and only the 1-cells labeled by $a$. The *agent task sheaf* is the functor $\mathcal{F}_a \colon \mathbf{Cell}_a(\mathcal{S}) \to \mathbf{Cset}$ defined by the same construction of Definition 5.23 restricted $\mathbf{Cell}_a(\mathcal{S})$, i.e.,

- For each 0-cell $\mathbf{c}$ in $\mathbf{Cell}_a(\mathcal{S})$, $\mathcal{F}_a(\mathbf{c}) = \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c})))$, i.e., the cset of valid

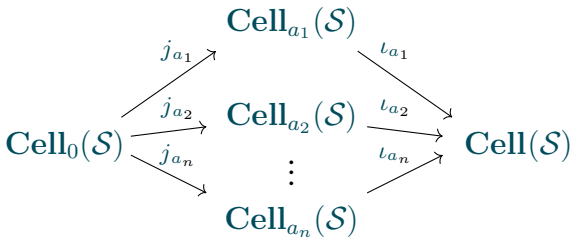
decisions at $\mathbf{c}$;

- For each 1-cell $(\mathbf{c}, \mathbf{c}')_a$ in $\mathbf{Cell}_a(\mathcal{S})$, $\mathcal{F}_a((\mathbf{c}, \mathbf{c}')_a) = \partial_{\mathcal{A},\{a\}}(\mathcal{O})$, i.e., the cset of local decisions of agent $a$;
- For each face map $\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a$ in $\mathbf{Cell}_a(\mathcal{S})$, $\mathcal{F}_a(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a) = \partial_{U_{\mathbf{c}},\{a\}} : \mathcal{F}_a(\mathbf{c}) \to \mathcal{F}_a((\mathbf{c}, \mathbf{c}')_a)$, i.e., the projection to agent $a$'s faces.

◀

Each agent sheaf encodes only the constraints that a single agent must satisfy for causal consistency. Because of their independence across each agent, but still over the same set of global states, we characterize the cell category $\mathbf{Cell}(\mathcal{S})$ as the gluing the agent slices $\mathbf{Cell}_a(\mathcal{S})$ along their common 0-cells.

▶ **Lemma 5.28 (Colimit of agent base spaces).** Let $\mathcal{S} = (\mathrm{ccl}(A), \{\simeq_a\}_{a\in\mathcal{A}})$ be a system slice. The cell category $\mathbf{Cell}(\mathcal{S})$ alongside the morphisms $\{\iota_a\}_{a\in\mathcal{A}}$ is the colimit in $\mathbf{Cat}$ of the following diagram.




(5.1)

where $\mathbf{Cell}_0(\mathcal{S})$ is the discrete subcategory of 0-cells and only identity morphisms and the arrows $\{j_a\}_{a\in\mathcal{A}}$ are inclusion functors. ◀

*Proof.* By Definition 5.17, the objects of $\mathbf{Cell}(\mathcal{S})$ are 0-cells (global states in $\mathrm{ccl}(A)$) and 1-cells (pairs $(\mathbf{c}, \mathbf{c}')_a$ for $(\mathbf{c}, \mathbf{c}') \in \simeq_a$). Each 1-cell is labeled by exactly one agent $a \in \mathcal{A}$, as the causal consistency relations are disjoint. This means that the 1-cells are partitioned across agents. By Definition 5.27, $\mathbf{Cell}_a(\mathcal{S})$ contains all 0-cells and precisely the 1-cells labeled by $a$. Consider $\mathbf{Cell}_a(\mathcal{S})$ and $\mathbf{Cell}_{a'}(\mathcal{S})$ for any two distinct agents $a, a' \in \mathcal{A}$, and let $\mathbf{Cell}_0(\mathcal{S})$ be the discrete subcategory formed only by the 0-cells from all global states in $\mathrm{ccl}(A)$ with only identity morphisms. As the 1-cells are disjoint while all three contain all 0-cells, we have that $\mathbf{Cell}_a(\mathcal{S}) \cap \mathbf{Cell}_{a'}(\mathcal{S}) = \mathbf{Cell}_0(\mathcal{S})$, for any $a \neq a'$.

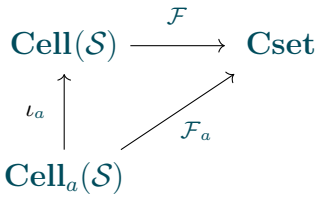
For each $a \in \mathcal{A}$, consider the inclusion functor $\iota_a \colon \mathbf{Cell}_a(\mathcal{S}) \to \mathbf{Cell}(\mathcal{S})$ defined by $\iota_a(\sigma) = \sigma$ on objects and morphisms. Consider $j_a \colon \mathbf{Cell}_0(\mathcal{S}) \to \mathbf{Cell}_a(\mathcal{S})$ to be the inclusion of 0-cells. Note that object $\mathbf{Cell}(\mathcal{S})$ and the inclusions $\{\iota_a\}_{a\in\mathcal{A}}$ form a cocone over the diagram in $\mathbf{Cat}$ of the objects $\mathbf{Cell}_0(\mathcal{S})$ and $\{\mathbf{Cell}_a(\mathcal{S})\}_{a\in\mathcal{A}}$ with morphisms $\{j_a\}_{a\in\mathcal{A}}$. That is, for any two agents $a, a'$ and $\mathbf{c} \in \mathbf{Cell}_0(\mathcal{S})$, we have that $\iota_a(j_a(\mathbf{c})) = \mathbf{c} = \iota_{a'}(j_{a'}(\mathbf{c}))$.

We verify now that this cocone is universal, i.e., that any compatible family of functors factors uniquely through $\mathbf{Cell}(\mathcal{S})$. Consider any category $\mathbf{C} \in \mathbf{Cat}$ and compatible family of functors $\{\mathcal{G}_a \colon \mathbf{Cell}_a(\mathcal{S}) \to \mathbf{C}\}_{a\in\mathcal{A}}$, meaning that $\mathcal{G}_a \circ j_a = \mathcal{G}_{a'} \circ j_{a'}$ for all $a, a'$. We construct a functor $\mathcal{G} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{C}$ as follows. For any 0-cell $\mathbf{c} \in \mathbf{Cell}_0(\mathcal{S})$, let $\mathcal{G}(\mathbf{c}) = \mathcal{G}_a(\mathbf{c})$ for any $a \in \mathcal{A}$. This is well-defined by compatibility, as $\mathcal{G}_a(\mathbf{c}) = \mathcal{G}_{a'}(\mathbf{c})$ for all $a, a' \in \mathcal{A}$. For any 1-cell $(\mathbf{c}, \mathbf{c}')_a$ labeled by $a$, define $\mathcal{G}((\mathbf{c}, \mathbf{c}')_a) = \mathcal{G}_a((\mathbf{c}, \mathbf{c}')_a)$. This is also well-defined as $(\mathbf{c}, \mathbf{c}')_a \in \mathbf{Cell}_a(\mathcal{S})$ only as 1-cells are disjoint across agents. For morphisms, for every cell $\sigma$ define $\mathcal{G}(\mathrm{id}\sigma) = \mathrm{id}_{\mathcal{G}(\sigma)}$ for identities, and $\mathcal{G}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a) = \mathcal{G}_a(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)$ for face maps. By construction, $\mathcal{G} \circ \iota_a = \mathcal{G}_a$ for all $a \in \mathcal{A}$.

In order to show that $\mathcal{G} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{C}$ is unique, suppose that there is another $\mathcal{H} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{C}$ also satisfying $\mathcal{H} \circ \iota_a = \mathcal{G}_a$ for all $a \in \mathcal{A}$. For any 0-cell $\mathbf{c}$, we have that $\mathcal{H}(\mathbf{c}) = \mathcal{H}(\iota_a(\mathbf{c})) = \mathcal{G}_a(\mathbf{c}) = \mathcal{G}(\mathbf{c})$. For any 1-cell $(\mathbf{c}, \mathbf{c}')_a$, we have that $\mathcal{H}((\mathbf{c}, \mathbf{c}')_a) = \mathcal{H}(\iota_a((\mathbf{c}, \mathbf{c}')_a)) = \mathcal{G}_a((\mathbf{c}, \mathbf{c}')_a) = \mathcal{G}((\mathbf{c}, \mathbf{c}')_a)$. The same can be verified for the morphisms. As such, $\mathcal{H} = \mathcal{G}$. This means that the cocone $\mathbf{Cell}(\mathcal{S})$ with morphisms $\{\iota_a\}_{a\in\mathcal{A}}$ over the diagram of $\{j_a \colon \mathbf{Cell}_0(\mathcal{S}) \to \mathbf{Cell}_a(\mathcal{S})\}_{a\in\mathcal{A}}$ satisfies the universal property of the colimit. ■

As each 1-cell of $\mathbf{Cell}(\mathcal{S})$ belongs to exactly one agent, the colimit glues these disjoint sets of edges while identifying the common vertices. The global task sheaf inherits this decomposition through the gluing condition for sheaves on a colimit as we see now.

▶ **Theorem 5.29 (Characterization of the task sheaf by agent sheaves).** Let $\{\mathcal{F}_a \colon \mathbf{Cell}_a(\mathcal{S}) \to \mathbf{Cset}\}_{a\in\mathcal{A}}$ be the agent task sheaves for a system slice $\mathcal{S}$ and task $\mathbf{T}$. The task sheaf $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ is the unique functor making the following diagram commute for all agents $a \in \mathcal{A}$.




That is, $\mathcal{F} \circ \iota_a = \mathcal{F}_a$ for all $a \in \mathcal{A}$, where $\iota_a$ is the inclusion from Lemma 5.28.

*Proof.* By Lemma 5.28, $\mathbf{Cell}(\mathcal{S})$ alongside the inclusions $\{\iota_a\}_{a\in\mathcal{A}}$ is the colimit of the diagram formed by the embeddings $\{j_a \colon \mathbf{Cell}_0(\mathcal{S}) \to \mathbf{Cell}_a(\mathcal{S})\}_{a\in\mathcal{A}}$. By the universal property of the colimit, any compatible family of functors from categories $\mathbf{Cell}_a(\mathcal{S})$ to some category $\mathbf{C}$ factor uniquely through $\mathbf{Cell}(\mathcal{S})$. We apply this with $\mathbf{C} = \mathbf{Cset}$ to derive that the task sheaf is such unique factorization. For that, we verify that the agent task sheaves $\{\mathcal{F}_a \colon \mathbf{Cell}_a(\mathcal{S}) \to \mathbf{Cset}\}_{a\in\mathcal{A}}$ form a compatible family over Diagram 5.1. That is, $\mathcal{F}_a \circ j_a = \mathcal{F}_{a'} \circ j_{a'}$ for all $a, a' \in \mathcal{A}$. Note that, for any global state $\mathbf{c} \in \mathbf{Cell}_0(\mathcal{S})$ and any agents $a, a' \in \mathcal{A}$, by Definition 5.27, we have that $\mathcal{F}_a(j_a(\mathbf{c})) = \mathcal{F}_a(\mathbf{c}) = \pi_{\mathcal{O}}(\pi_{\mathcal{I}}^{-1}(\mathrm{i}(\mathbf{c}))) = \mathcal{F}_{a'}(\mathbf{c}) = \mathcal{F}_{a'}(j_{a'}(\mathbf{c}))$. As such, the agent task sheaves do form a compatible family. Thus, there exists a unique functor $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ such that $\mathcal{F} \circ \iota_a = \mathcal{F}_a$ for all $a \in \mathcal{A}$. Note now that by Definition 5.27, each agent task sheaf is defined by restricting the construction of Definition 5.23 to $\mathbf{Cell}_a(\mathcal{S})$. The colimit recovers $\mathcal{F}$ by setting $\mathcal{F}(\sigma) = \mathcal{F}_a(\sigma)$ for any cell $\sigma \in \mathbf{Cell}_a(\mathcal{S})$. As such, the unique factorization $\mathcal{F}$ matches the original task sheaf. ■

The characterization of Theorem 5.29 shows that the task sheaf arises as the unique gluing of agent task sheaves. This form of compositionality is direct consequence of the independence of the causal consistency relations across agents, and enables a modularity in reasoning about the agent-wise tasks. Each agent task sheaf may be expressed and modified independently with the guarantee that they can be uniquely composed into a single task solvability problem.

From the understanding of the task sheaf as encoding the constraints imposed by the distributed system (in the system slice) alongside the possible solutions to a task (in the space of decision functions), we proceed now to characterize when these constraints admit a solution.

## 5.3 TASK SOLVABILITY AND SHEAF SECTIONS

The task sheaf constructed in Section 5.2 formalizes the relationship between the validity constraints from task specification and the consistency constraints from the system slice. We turn now to a central application of this framework: assessing whether the distributed system and task encoded in such a sheaf admit a solution. For that purpose, we will make use of the notion of a section (Definition 2.41), the standard definition of consistency in sheaf theory.

While sections have been vastly used to encode consistent data in distributed networks (as reviewed in Section 2.3.4), we turn to an application closer to the original motivation within sheaf theory, namely the study of locally defined functions over a parameter space that must agree on overlapping regions. The key insight is that the sections of a task sheaf correspond precisely to the terminating decision functions solving the task. This correspondence makes explicit the local nature of task solvability, as a global solution exists if and only if constraints local to each global state and agent can be simultaneously satisfied. We recall now what sections are for cellular sheaves in our context.

▶ **Definition 5.30 (Sections in a task sheaf).** Let $\mathcal{F}\colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ be a task sheaf over the cell category of a system slice $\mathcal{S}$. A *section* of $\mathcal{F}$ is a family $s = (s_\alpha)_{\alpha \in \mathbf{Cell}(\mathcal{S})}$ with $s_\alpha \in \mathcal{F}(\alpha)$ for each cell $\alpha$, such that for every 0-cell $\mathbf{c}$ and incident 1-cell $(\mathbf{c}, \mathbf{c}')_a$, the restriction maps satisfy

$$\mathcal{F}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)(s_{\mathbf{c}}) = s_{(\mathbf{c},\mathbf{c}')_a}.$$

The set of all sections of $\mathcal{F}$ over $\mathcal{S}$ is denoted $\Gamma(\mathcal{S}, \mathcal{F})$. ◀

The compatibility condition of Definition 5.30 requires that the global states at both ends of a causal consistency project to the same value within the edge. Explicitly, for each edge $(\mathbf{c}, \mathbf{c}') \in \simeq_a$, we get the constraint

$$\mathcal{F}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)(s_{\mathbf{c}}) = s_{(\mathbf{c},\mathbf{c}')_a} = \mathcal{F}(\mathbf{c}' \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)(s_{\mathbf{c}'}).$$

Since the restriction maps are projections to the $a$-th component (Definition 5.23), this enforces that agent $a$ decides identically in both global states $\mathbf{c}$ and $\mathbf{c}'$. The section condition captures then the determinism and termination requirements that causal consistency encodes (Definition 5.11). When combined with the stalks

that contain only task consistent outputs by construction, a section satisfies all criteria for a solving terminating decision function. We formalize now this correspondence.

▶ **Lemma 5.31 (Correspondence of sections and decisions).** Let $\mathcal{S}$ be a system slice over an execution cut $A$, $\mathbf{T}$ a task, and $\mathcal{F}\colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ the corresponding task sheaf. A family $(s_\alpha)_{\alpha\in\mathbf{Cell}(\mathcal{S})}$ is a section of $\mathcal{F}$ if and only if the assignment $d(\pi_a(\mathbf{c})) = \partial_{U_\mathbf{c},\{a\}}(s_\mathbf{c})$ defines a terminating decision function solving $\mathbf{T}$ on the causal closure of $A$. ◀

*Proof.* We proceed with each direction separately.

⇒ Assume that $(s_\alpha)_{\alpha\in\mathbf{Cell}(\mathcal{S})}$ is a section and consider the agent-wise candidate decision function $d(\pi_a(\mathbf{c})) = \partial_{U_\mathbf{c},\{a\}}(s_\mathbf{c})$ for each agent $a$ and global state $\mathbf{c}$ in $\mathrm{ccl}(A)$. We verify now that $d$ satisfies termination, determinism and task satisfaction from Remark 5.2. By Definition 5.23, we have that the stalk $\mathcal{F}(\mathbf{c}) = \pi_\mathcal{O}(\pi_\mathcal{I}^{-1}(\mathrm{i}(\mathbf{c})))$ contains only valid output simplices for its inputs. Since $s_\mathbf{c} \in \mathcal{F}(\mathbf{c})$, the assignment of $d$ satisfies task satisfaction.

For determinism and termination, suppose that $\mathbf{c} \sim_a \mathbf{c}'$ or $\mathbf{c} \leq_a \mathbf{c}'$ within $\mathrm{ccl}(A)$. By Definition 5.11, both relations are contained in $\simeq_a$, as such $(\mathbf{c},\mathbf{c}') \in \simeq_a$, and so by Definition 5.17, there exists a 1-cell $(\mathbf{c},\mathbf{c}')_a \in \mathbf{Cell}(\mathcal{S})$. Note now that the section condition of Definition 5.30 requires that $\mathcal{F}(\mathbf{c} \trianglelefteq (\mathbf{c},\mathbf{c}')_a)(s_\mathbf{c}) = s_{(\mathbf{c},\mathbf{c}')_a}$. This means that, for agents $U_\mathbf{c}$ in $\mathbf{c}$, $\partial_{U_\mathbf{c},\{a\}}(s_\mathbf{c}) = s_{(\mathbf{c},\mathbf{c}')_a}$. The same holds for $\mathbf{c}'$, meaning that $\partial_{U_{\mathbf{c}'},\{a\}}(s_{\mathbf{c}'}) = s_{(\mathbf{c},\mathbf{c}')_a}$. As such, $d(\pi_a(\mathbf{c})) = d(\pi_a(\mathbf{c}'))$, completing the conditions for a terminating decision function.

⇐ Assume that $d$ is a terminating decision function on $\mathrm{ccl}(A)$ and consider the family $s_\mathbf{c} = (d(\pi_a(\mathbf{c})))_{a\in U_\mathbf{c}}$ and $s_{(\mathbf{c},\mathbf{c}')_a} = d(\pi_a(\mathbf{c}))$ for each $(\mathbf{c},\mathbf{c}') \in \simeq_a$. By Definition 5.23, a task sheaf assigns only valid outputs to each stalk, and so by task satisfaction, each $s_\mathbf{c} \in \mathcal{F}(\mathbf{c})$. By Definition 5.17, for each pair $(\mathbf{c},\mathbf{c}') \in \simeq_a$, there exists one 1-cell $(\mathbf{c},\mathbf{c}')_a \in \mathbf{Cell}(\mathcal{S})$, meaning that the values of $s_{(\mathbf{c},\mathbf{c}')_a}$ are defined for all 1-cells. By Definition 5.11, for $(\mathbf{c},\mathbf{c}') \in \simeq_a$, it holds either that $\mathbf{c} \sim_a \mathbf{c}'$ or $\mathbf{c} \leq_a \mathbf{c}'$. As $d$ satisfies determinism and termination, we have that $d(\pi_a(\mathbf{c})) = d(\pi_a(\mathbf{c}'))$. Since the definition of $s_{(\mathbf{c},\mathbf{c}')_a}$ is independent of which endpoint is used for its definition, it is well-defined.

In order to verify the section condition of Definition 5.30, consider any face relation $\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a$. Recall that $s_{\mathbf{c}}$ and $s_{(\mathbf{c},\mathbf{c}')_a}$ are defined by $d$ as $s_{(\mathbf{c},\mathbf{c}')_a} = d(\pi_a(\mathbf{c}))$, and $s_{\mathbf{c}} = (d(\pi_{a'}(\mathbf{c})))_{a' \in U_{\mathbf{c}}}$, and note that by Definition 5.23 the restriction map is $\mathcal{F}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a) = \partial_{U_{\mathbf{c}},\{a\}}$, which projects to the $a$-th component. As such, we have that $\mathcal{F}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)(s_{\mathbf{c}}) = \partial_{U_{\mathbf{c}},\{a\}}((d(\pi_{a'}(\mathbf{c})))_{a' \in U_{\mathbf{c}}}) = d(\pi_a(\mathbf{c})) = s_{(\mathbf{c},\mathbf{c}')_a}$. That is, $s_\alpha$ is a section.

■

We illustrate this correspondence with the $\varepsilon$-agreement task in Example 5.32, where relaxed constraints with respect to binary consensus enable the existence of sections.

▶ **Example 5.32 (Approximate agreement).** We demonstrate through the $\varepsilon$-agreement task how relaxing the agreement constraints affect the existence of sections. Recall from Section 2.2.1 that $\varepsilon$-agreement requires two agents with inputs $(v_a, v_b)$ to decide values $(d_a, d_b)$ that are within the convex hull of the inputs (VALIDITY) and within $\varepsilon$ distance of each other (AGREEMENT), i.e., $|d_a - d_b| \leqslant \varepsilon$. Note that, while smaller $\varepsilon$ provides more valid combinations of output values, each interval is more tightly constrained, reducing the flexibility when constructing sections.

Consider the lossy link message adversary with two agents $\mathcal{A} = \{a, b\}$ and binary inputs $\{v_a, v_b \in \{0, 1\}\}$. We take the system slice after one round where at most one message may be lost, as in Example 5.26. The global states are of the form $(v_a, v_b, \mathrm{com})$, where $\mathrm{com} \in \{\leftarrow, \leftrightarrow, \rightarrow\}$, displaying the input values and the pattern of message delivered. The corresponding task sheaf with $\varepsilon = 0.25$ is constructed according to Definition 5.23 and displayed in Figure 5.3. The stalks above each global state (0-cells), depicted as the values connected by a dashed lined, are constrained by VALIDITY, which are singletons whenever the inputs are identical. Note that such states $(1_a, 1_b, \leftarrow)$ and $(0_a, 0_b, \rightarrow)$ form a boundary with decisions $(1_a, 1_b)$ and $(0_a, 0_b)$, respectively, where the intermediate global states must bridge this gap.

The values shown in yellow trace a candidate section, starting from the forced decision of $(1_a, 1_a)$ at state $(1_a, 1_b, \leftarrow)$. By following the alternating agent edges (1-cells), the value of the corresponding agent is preserved at each step, while the other is allowed to move by at most $0.25$. Each choice is required as the

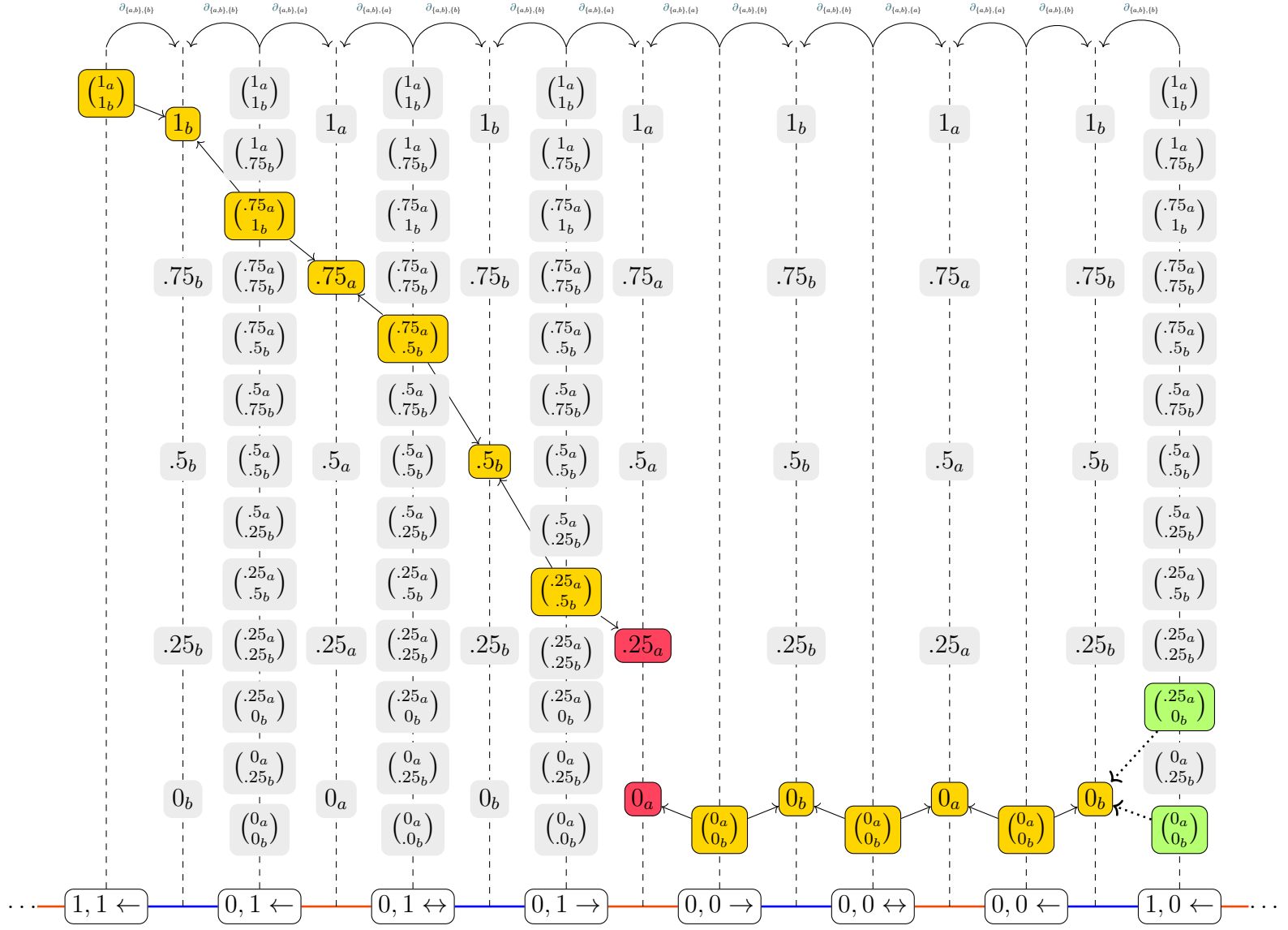



Figure 5.3
Task sheaf for $\varepsilon$-agreement at $\varepsilon = 0.25$. The stalks are shown as gray boxes above each cell, connected by the dashed line. Yellow values show a candidate section, where the mismatch is marked red. Green values are two valid assignments for the given constraints, while yellow ones are forced as the only options.

best attempt at global consistency, which fails at the edge $((0_a, 1_a, \rightarrow), (0_a, 0_b, \rightarrow))_a$, where agent $a$ at the left cell could choose between values $\{0.25, 0.5, 0.75\}$ in order to respect agent $b$'s fixed value of $0.5$, but at the same time the right cell required a choice of value $0$. The red highlighted values mark this incompatibility. In contrast,

at global state $(1_a, 0_b, \leftarrow)$, the only constraint comes from the fixed decision of $0$ for agent $b$ from the left cell, making two possible assignments equally valid, marked green. Had the slice continued further, there may have been an equal or smaller number of options, including possibly none.

Alternatively, Figure 5.4 shows the task sheaf for $\varepsilon = 0.5$. Although the stalks contain fewer elements, each step cover a larger interval of possible decisions. This sheaf admits a valid section, indicated by the yellow chain with the following values.

$$
\begin{aligned}
s_{(1_a,1_b,\leftarrow)} &= (1_a, 1_b), & s_{((1_a,1_b,\leftarrow),(0_a,1_b,\leftarrow))_b} &= 1_b,\\
s_{(0_a,1_b,\leftarrow)} &= (0.5_a, 1_b), & s_{((0_a,1_b,\leftarrow),(0_a,1_b,\leftrightarrow))_a} &= 0.5_a,\\
s_{(0_a,1_b,\leftrightarrow)} &= (0.5_a, 0_b), & s_{((0_a,1_b,\leftrightarrow),(0_a,1_b,\rightarrow))_b} &= 0.5_b,\\
s_{(0_a,1_b,\rightarrow)} &= (0_a, 0_b), & s_{((0_a,1_b,\rightarrow),(0_a,0_b,\rightarrow))_b} &= 0_b,\\
s_{(0_a,0_b,\rightarrow)} &= (0_a, 0_b).
\end{aligned}
$$

By Lemma 5.31, this section corresponds to a terminating decision function from local states $(v, m)$, where $m$ is the message received or $\perp$ if no message has arrived, to output values in $\mathcal{O}$. The decision of each agent can be extracted by associating each local state to the corresponding section, which will be unique, as it respects causal consistency. The following map describes this instance.

$$
d(v, m) = \begin{cases} v & \text{if } m = \perp, \\ \dfrac{v + m}{2} & \text{otherwise.} \end{cases}
$$

Note that the impossibility for $\varepsilon = 0.25$ is a consequence of the choice of system slice, rather than inherent to the task. Additional rounds of communication would grow the system slice to have sufficient intermediary global states between the forced boundary decisions. For the lossy link adversary, each round doubles the number of steps available in the chain of Figure 5.3, allowing agreement with smaller values of $\varepsilon$ to be solvable. ◀

Example 5.32 demonstrates how the size of the stalks translates to a flexibility in the choice of values, possibly enabling the existence of sections. In the case of the approximate agreement task this is regulated by the $\varepsilon$ parameter, where binary consensus can be understood as one end of the spectrum where $\varepsilon = 0$.

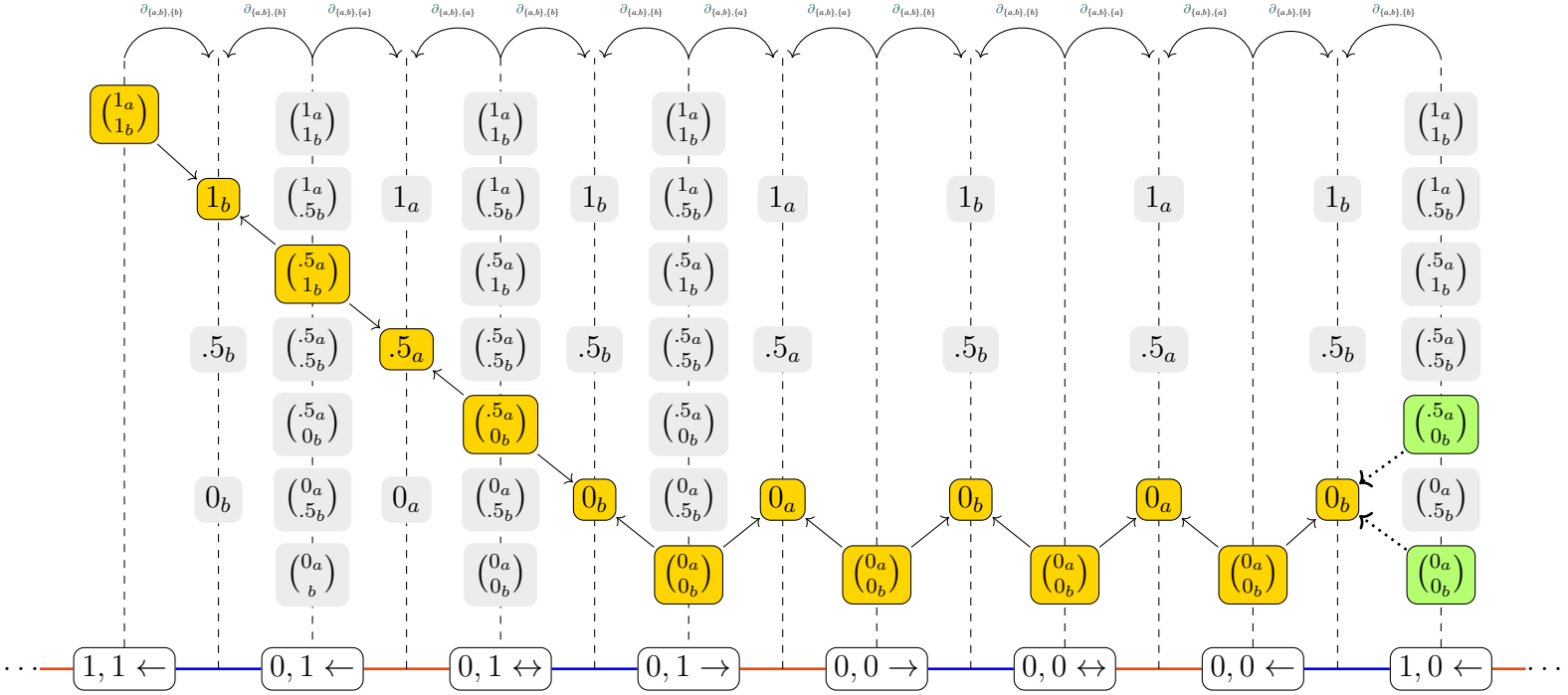



Figure 5.4
Task sheaf for $\varepsilon$-agreement at $\varepsilon = 0.5$. Yellow values show the only valid section, while green values at are both valid choices.

Recall from Section 5.1 that system slices extract finite substructures with a meaningful interpretation of task solvability. We make use of this observation to establish that solvability in the original system frame reduces to the existence of sections in the task sheaf associated to any terminal system slice.

▶ **Theorem 5.33 (Characterization of task solvability).** Let $\mathbf{T}$ be a task and $\mathbf{S}$ be a distributed system with system frame $\mathcal{K}_{\mathbf{S}}$. There exists a terminating decision function solving $\mathbf{T}$ in $\mathcal{K}_{\mathbf{S}}$ if and only if there exists an execution cut $A$ such that the task sheaf $\mathcal{F} : \mathbf{Cell}(\mathcal{S}_A) \to \mathbf{Cset}$ over the system slice $\mathcal{S}_A$ admits a section.

*Proof.* We proceed with each direction separately.

⇒ Assume that there exists a terminating decision function $d$ solving $\mathbf{T}$ in $\mathcal{K}_{\mathbf{S}}$. By Lemma 5.4, there exists a terminal execution cut $A$ such that for every agent $a \in \mathcal{A}$ and every global state $\mathbf{c} \in A$, it holds that $d(\pi_a(\mathbf{c})) \neq \bot$. Note that the restriction of $d$ to $\mathrm{ccl}(A)$ satisfies determinism, termination and

task satisfaction within the causal closure, as there properties are preserved when restricting to a subset of global states. By Lemma 5.31, the family $(s_\alpha)_{\alpha \in \mathbf{Cell}(\mathcal{S}_A)}$, defined by $s_\alpha = (d(\pi_a(\mathbf{c})))_{a \in U_\mathbf{c}}$ for 0-cells, and $s_{(\mathbf{c},\mathbf{c}')_a} = d(\pi_a(\mathbf{c}))$ for 1-cells, is a section of $\mathcal{F}$. Therefore, there exists at least one section, i.e., $\Gamma(\mathcal{S}_A, \mathcal{F}) \neq \varnothing$.

$\Leftarrow$ Assume that there exists an execution cut $A$ such that the corresponding task sheaf $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}_A) \to \mathbf{Cset}$ admits a section $(s_\alpha)_{\alpha \in \mathbf{Cell}(\mathcal{S}_A)}$. By Lemma 5.31, the assignment $d(\pi_a(\mathbf{c})) = \partial_{U_\mathbf{c},\{a\}}(s_\mathbf{c})$ defines a terminating decision function on the causal closure $\mathrm{ccl}(A)$, that is, one that satisfies determinism, termination and task satisfaction. By Theorem 5.10, such properties verified on $\mathrm{ccl}(A)$ extend to a terminating decision function solving $\mathbf{T}$ on the entire system frame $\mathcal{K}_\mathbf{S}$.

■

Theorem 5.33 connects the operational perspective of distributed computing with the topological perspective of sheaf theory, enabled principally by the locality of system slices shown in Theorem 5.10 and the correspondence of sections and solutions in Lemma 5.31. This characterization shifts the problem from searching for a protocol to verifying the existence of sections, where for a given system model and task, one must identify a terminal execution cut and compute whether the induced task sheaf admits a section. When sections exist, they encode terminating decision functions directly. The computational method needed for finding such solutions through cohomology is the subject of the following section.

## 5.4 COMPUTING SOLUTIONS

While the characterization of Theorem 5.33 reduces task solvability to the existence of sections in the corresponding task sheaf, it does not provide a method for finding them. We close this gap by applying the standard cohomology theory for cellular sheaves [Cur14, Chapter 7] to task sheaves, which allows us to transform the conditions for a section to exist into a linear algebra problem. The approach consists of linearizing the stalks of the sheaf as the free abelian group that they generate, where the restriction map constraints assemble into coboundary operators, whose kernel captures all assignments that agree under restriction maps.

Most notably for computational applications, such kernel can be found through standard linear algebra, which translates to a finite search.

The cohomology machinery for cellular sheaves is developed using vector spaces (sheaves valued in $\mathbf{Vect}_{\mathbb{K}}$ for a field $\mathbb{K}$), while classical constructions from algebraic topology use integer coefficients [Hat01]. We follow the latter, as in the context of task sheaves, it suffices to extract from each stalk the specific set of decisions associated to the agents of each cell. That is, at global states $\mathbf{c}$, we look at the $U_{\mathbf{c}}$-simplices, while at edges $(\mathbf{c}, \mathbf{c}')_a$, we need only the local decisions at $\{a\}$-simplices. This information assembles into the free abelian group generated by the sets of decision values.

▶ **Definition 5.34 (Free abelian group functor [Lan02]).** The *free abelian group functor* $\mathbb{Z}[-] \colon \mathbf{Set} \to \mathbf{Ab}$ sends each set $X \in \mathbf{Set}$ to its *free abelian group* $\mathbb{Z}[X] = (|\mathbb{Z}[X]|, +, 0)$, where the carrier set is

$$|\mathbb{Z}[X]| = \left\{ \sum_{x \in X} n_x \cdot x \mid n_x \in \mathbb{Z}, \text{finitely many } n_x \neq 0 \right\}$$

whose zero is the empty sum and a coefficient-wise addition. Each function $f \colon X \to Y$ is sent to the group homomorphism $\mathbb{Z}[f] \colon \mathbb{Z}[X] \to \mathbb{Z}[Y]$ given by

$$\mathbb{Z}[f](\sum n_x \cdot x) = \sum n_x (f(x))$$

Equivalently, $\mathbb{Z}[-]$ is the left adjoint of the functor $U \colon \mathbf{Ab} \to \mathbf{Set}$ forgetting the group structure. ◀

We can now define a linearized task sheaf.

▶ **Definition 5.35 (Linearized task sheaf).** Let $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ be a task sheaf over system slice $\mathcal{S}$. The *linearized task sheaf* is the functor $\mathcal{L} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Ab}$, where $\mathbf{Ab}$ is the category of abelian groups, defined by applying the free abelian group functor to each stalk after evaluating at the relevant set of agents:

- At 0-cell $\mathbf{c}$ with agents $U_{\mathbf{c}}$: $\mathcal{L}(\mathbf{c}) = \mathbb{Z}[\mathcal{F}(\mathbf{c})(U_{\mathbf{c}})]$;
- At 1-cell $(\mathbf{c}, \mathbf{c}')_a$: $\mathcal{L}((\mathbf{c}, \mathbf{c}')_a) = \mathbb{Z}[\mathcal{F}((\mathbf{c}, \mathbf{c}')_a)(\{a\})]$;
- For restriction maps $\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a$: $\mathcal{L}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a) = \mathbb{Z}[\partial_{U_{\mathbf{c}}, \{a\}}]$.

◀

The free abelian group functor comes with an embedding $\iota_X : X \hookrightarrow \mathbb{Z}[X]$ that sends each element $x \in X$ to the corresponding generator $1 \cdot x$ within $\mathbb{Z}[X]$. It lifts to relate the task sheaf to the linearized task sheaf as follows.

▶ **Definition 5.36 (Generator embedding).** Let $\mathcal{F} : \mathbf{Cell}(\mathcal{S}) \to \mathbf{Cset}$ be a task sheaf and $\mathcal{L} : \mathbf{Cell}(\mathcal{S}) \to \mathbf{Ab}$ its linearization. The *generator embedding* is a family of functions $\{\iota_\alpha : \mathcal{F}(\alpha)(U_\alpha) \to \mathcal{L}(\alpha)\}_{\alpha \in \mathbf{Cell}(\mathcal{S})}$, where $U_\alpha$ denotes the set of agents at cell $\alpha$. Each map $\iota_\alpha$ sends a decision simplex $\sigma \in \mathcal{F}(\alpha)(U_\alpha)$ to its corresponding generator element in the linearized task sheaf, i.e., $1 \cdot \sigma \in \mathbb{Z}[\mathcal{F}(\alpha)(U_\alpha)] = \mathcal{L}(\alpha)$. ◀

▶ **Example 5.37 (Linearized stalks).** Recall the task sheaf $\mathcal{F}$ from Example 5.26 encoding the binary consensus task. Consider a global state $(0_a, 1_b, \leftrightarrow)$, whose corresponding stalk $\mathcal{F}((0_a, 1_b, \leftrightarrow))$ contains two valid output simplices, $(0_a, 0_b)$ and $(1_a, 1_b)$. The linearized stalk, by Definition 5.35, is then

$$\mathcal{L}((0_a, 1_b, \leftrightarrow)) = \mathbb{Z}[\{(0_a, 0_b), (1_a, 1_b)\}] \cong \mathbb{Z} \oplus \mathbb{Z}$$

where each possible decision corresponds to one generator. More precisely, each value from the stalk at $\mathcal{F}$ can be embedded in the linearized task sheaf $\mathcal{L}$ as a generator. For instance, the stalk at $\mathcal{L}$ of $(0_a, 1_b, \leftrightarrow)$ is the group generated by $\{x \cdot (0_a, 0_b) + y \cdot (1_a, 1_b) \mid x, y \in \mathbb{Z}\}$, where $\iota : (1_a, 1_b) \mapsto 0 \cdot (0_a, 0_b) + 1 \cdot (1_a, 1_b)$. ◀

The abelian group structure of the linearized task sheaf allows us to express the compatibility condition of sections as linear constraints. The data from the stalks on 0 and 1-cells form a graded abelian group with the possible global and local decisions at each level. A coboundary operator derived from the restriction maps connects then those structures by measuring the failure of an assignment to be consistent.

▶ **Definition 5.38 (Cochain groups, coboundary operator and cochain complex).** Let $\mathcal{L} : \mathbf{Cell}(\mathcal{S}) \to \mathbf{Ab}$ be the linearized task sheaf on a system slice $\mathcal{S}$. The 0- and 1-*cochain groups* are given by the direct sums of stalks at 0- and 1-cells:

$$C^0(\mathcal{L}) = \bigoplus_{\mathbf{c} \in \mathbf{Cell}(\mathcal{S})} \mathcal{L}(\mathbf{c}), \qquad C^1(\mathcal{L}) = \bigoplus_{(\mathbf{c},\mathbf{c}')_a \in \mathbf{Cell}(\mathcal{S})} \mathcal{L}((\mathbf{c}, \mathbf{c}')_a)$$

For an arbitrary orientation on each 1-cell, the *coboundary operator* $\delta^0 : C^0(\mathcal{L}) \to C^1(\mathcal{L})$ is defined component-wise by

$$\delta^0(\phi)_{(\mathbf{c},\mathbf{c}')_a} = \mathcal{L}(\mathbf{c}' \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)(\phi_{\mathbf{c}'}) - \mathcal{L}(\mathbf{c} \trianglelefteq (\mathbf{c}, \mathbf{c}')_a)(\phi_{\mathbf{c}})$$

Given an arbitrary ordering of 0-cells $(\mathbf{c}_1, \ldots, \mathbf{c}_m)$ and 1-cells $(e_1, \ldots, e_n)$, where $e_j = (\mathbf{c}_i, \mathbf{c}_{i'})_a$ is oriented $\mathbf{c}_i \to \mathbf{c}_{i'}$, the coboundary operator can be represented as a *difference matrix* $D$ with rows indexed by 1-cells and columns indexed by 0-cells. For $e_j = (\mathbf{c}_i, \mathbf{c}'_i)_a$ oriented $\mathbf{c}_i \to \mathbf{c}_{i'}$, the entries are

$$D_{j,i} = -\mathbb{Z}[\partial_{U_{\mathbf{c}_i},\{a\}}], \qquad D_{j,i'} = +\mathbb{Z}[\partial_{U_{\mathbf{c}_{i'}},\{a\}}], \qquad D_{j,k} = 0 \text{ for } k \notin \{i, i'\}$$

◀

This construction follows from the standard cellular chain in [Cur14]. Each element in the 0-cochain group assigns to each global state a linear combination with coefficients in $\mathbb{Z}$ of the decisions, while elements in the 1-cochain group assigns combinations of the agents' decisions to the edges. From such assignment, the coboundary map computes the difference of the values obtained through the restriction maps at each endpoint, in the direction of the orientation. That is, for $\phi \in C^0$, the equation $\delta^0(\phi) = 0$ is true precisely when adjacent stalks agree after restricting to the same agent's decision. In fact, this equality is the definition of the kernel of the coboundary operator, which is recovered in the zeroth cohomology group.

▶ **Definition 5.39 (Zeroth cohomology group).** Let $\mathcal{L} \colon \mathbf{Cell}(\mathcal{S}) \to \mathbf{Ab}$ be the linearized task sheaf, and let $\delta^0 \colon C^0(\mathcal{L}) \to C^1(\mathcal{L})$ be the associated coboundary operator. The *zeroth cohomology group* has as carrier

$$\mathcal{H}^0(\mathcal{L}) = \ker(\delta^0) = \{\phi \in C^0 \mid \delta^0(\phi) = 0\}$$

with operation and zero inherited from $C^0(\mathcal{L})$. ◀

▶ **Remark 5.40 (Functoriality of cohomology and holes).** More precisely, following the standard cohomology construction [Cur14], our linear space of decisions is a *cochain complex* $(C, \delta)$. That is, a graded abelian group defined by a collection of cochain groups $\{C^i\}_{i\in\mathbb{N}}$ and coboundary maps $\{\delta^i\}_{i\in\mathbb{N}}$. However, for the purpose of reasoning about the task sheaf, every cochain group for $i \geq 2$ and coboundary map for $i \geq 1$ is trivial and this abstract perspective does not add much information.

The importance of these constructions lies in the functoriality of cohomology, defined as a functor $\mathcal{H}^* \colon \mathbf{Ch}^\bullet(\mathbf{Ab}) \to \mathbf{grAb}$, where $\mathbf{Ch}^\bullet(\mathbf{Ab})$ is the category of

cochain complexes and $\mathbf{grAb}$ the category of graded abelian groups, with the following assignment

$$\mathcal{H}^i(C^i, \delta^i) \coloneqq \ker(\delta^i)/\mathrm{im}(\delta^{i-1})$$

which measures obstructions to extending local data globally, characterizing holes in a space at dimension $i$. In the case of task sheaves, we restrict ourselves to the zeroth cohomology, measuring failures of consistency at the level of decision assignments, where the "hole" detection analogy can be recovered in Figure 5.3 of Example 5.32. ◀

The zeroth cohomology group characterizes the assignments that satisfy the section compatibility condition from Definition 5.30, expressed in this linearized setting.

▶ **Lemma 5.41 (Isomorphism of sections and cohomology groups).** Let $\mathcal{L} : \mathbf{Cell}(\mathcal{S}) \to \mathbf{Ab}$ be a linearized task sheaf. There is a natural isomorphism

$$\mathcal{H}^0(\mathcal{L}) \cong \Gamma(\mathbf{Cell}(\mathcal{S}), \mathcal{L})$$

between the zeroth cohomology group and the sections of $\mathcal{L}$. ◀

*Proof.* Observe that an element $\phi \in C^0(\mathcal{L})$ is in $\mathcal{H}^0(\mathcal{L})$ if and only if $\delta^0(\phi) = 0$ (Definition 5.39). By definition 5.38, this means that, for every 1-cell $e = (\mathbf{c}, \mathbf{c}')_a$, the equation $\mathcal{L}(\mathbf{c} \trianglelefteq e)(\phi_{\mathbf{c}}) = \mathcal{L}(\mathbf{c}' \trianglelefteq e)(\phi_{\mathbf{c}'})$ holds, where $\phi_{\mathbf{c}} \in \mathcal{L}(\mathbf{c})$. This is precisely the condition for the compatibility of sections for $\mathcal{L}$ from Definition 5.30. As such, $\phi \in \Gamma(\mathbf{Cell}(\mathcal{S}), \mathcal{L})$. The group structure of $\Gamma(\mathbf{Cell}(\mathcal{S}), \mathcal{L})$ is inherited component-wise from the abelian group structure of the stalks valued in $\mathbf{Ab}$ and coincide as those form the cochain groups in Definition 5.38. ■

Lemma 5.41 is a special case of the standard identification of the zeroth cohomology with global sections [Bre12], which now reduces finding sections of the linearized task sheaf to computing the kernel of a coboundary operator. In order to reason about the original task sheaf, we can look at which elements of this kernel correspond to actual decisions instead of arbitrary linear combinations added during the linearization, an information that we have through the generator embedding of Definition 5.36. This allows us to reformulate characterization of Theorem 5.33 in terms of the associated zeroth cohomology.

▶ **Corollary 5.42 (Characterization of solvability by the zeroth cohomology).** Let $\mathbf{T}$ be a task and $\mathbf{S}$ a system with system frame $\mathcal{K}_{\mathbf{S}}$. There exists a terminating decision function solving $\mathbf{T}$ if and only if there exists an execution cut $A$ such that the zeroth cohomology group $\mathcal{H}^0(\mathcal{L})$ of the linearized task sheaf $\mathcal{L} \colon \mathbf{Cell}(\mathcal{S}_A) \to \mathbf{Ab}$ contains an element $\phi$ with $\phi_{\mathbf{c}} \in \mathrm{im}(\iota_{\mathbf{c}})$ for every 0-cell $\mathbf{c}$, where $\iota$ is a generator embedding. ◀

*Proof.* We want to connect task solvability to the cohomology of $\mathcal{L}$. By Theorem 5.33, the task $\mathbf{T}$ is solvable if and only if some terminal execution cut $A$ provides a task sheaf $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}_A) \to \mathbf{Cset}$ with $\Gamma(\mathcal{S}_A, \mathcal{F}) \neq \varnothing$. We proceed with each direction separately.

- $\Rightarrow$ Suppose that we have a section $s \in \Gamma(\mathcal{S}, \mathcal{F})$, i.e., a collection of csets $(s_\alpha \in \mathcal{F}(\alpha))_{\alpha \in \mathbf{Cell}(\mathcal{S})}$. We construct a corresponding element in $\mathcal{H}^0(\mathcal{L})$ by defining a collection $\phi \in C^0(\mathcal{L})$, for every 0-cell $\mathbf{c}$ in $\mathbf{Cell}(\mathcal{S})$, as the generator embedding $\phi_{\mathbf{c}} = \iota_{\mathbf{c}}(s_{\mathbf{c}}(U_{\mathbf{c}}))$, per Definition 5.36. By construction, we have that $\phi_{\mathbf{c}} \in \mathrm{im}(\iota_{\mathbf{c}})$. In order to verify that $\phi \in \mathcal{H}^0(\mathcal{L})$, we need to show that $\delta^0(\phi) = 0$. For that, consider any 1-cell $e = (\mathbf{c}, \mathbf{c}')_a$, which by the section condition for $s$ (Definition 5.30) says that $\mathcal{F}(\mathbf{c} \trianglelefteq e)(s_{\mathbf{c}}) = s_e = \mathcal{F}(\mathbf{c}' \trianglelefteq e)(s_{\mathbf{c}'})$. By the functoriality of the $\mathbb{Z}[-]$ (Definition 5.34), applying the free abelian group functor to this equation gives us $\mathbb{Z}[\mathcal{F}(\mathbf{c} \trianglelefteq e)](\phi_{\mathbf{c}}) = \mathbb{Z}[\mathcal{F}(\mathbf{c}' \trianglelefteq e)](\phi_{\mathbf{c}'})$, which is precisely the requirement that $\delta^0(\phi)_e = \mathcal{L}(\mathbf{c}' \trianglelefteq e)(\phi_{\mathbf{c}'}) - \mathcal{L}(\mathbf{c} \trianglelefteq e)(\phi_{\mathbf{c}}) = 0$ by Definition 5.38. As the 1-cell was arbitrary, we have that $\delta^0(\phi) = 0$, and hence $\phi \in \mathcal{H}^0(\mathcal{L})$.
- $\Leftarrow$ Suppose that $\phi \in \mathcal{H}^0(\mathcal{L})$ satisfies $\phi_{\mathbf{c}} \in \mathrm{im}(\iota_{\mathbf{c}})$ for all 0-cells $\mathbf{c}$. We want to extract a section $s \in \Gamma(\mathcal{S}, \mathcal{F})$ from this element. Since each $\phi_{\mathbf{c}}$ is in the image of the generator embedding, we know that there is some unique $\sigma_{\mathbf{c}} \in \mathcal{F}(\mathbf{c})(U_{\mathbf{c}})$ such that $\phi_{\mathbf{c}} = \iota_{\mathbf{c}}(\sigma_{\mathbf{c}})$, where the uniqueness follows from the injectivity of the embedding. We verify now that $s = (\sigma_{\mathbf{c}})_{\mathbf{c} \in \mathbf{Cell}(\mathcal{S})}$ is indeed a section. For that, consider any 1-cell $e = (\mathbf{c}, \mathbf{c}')_a$ and note that, Definition 5.39, $\phi \in \mathcal{H}^0(\mathcal{L})$, defined as $\delta^0(\phi) = 0$. That is, rewritten with Definition 5.38, $\mathbb{Z}[\partial_{U_{\mathbf{c}}, \{a\}}](\phi_{\mathbf{c}}) = \mathbb{Z}[\partial_{U_{\mathbf{c}'}, \{a\}}](\phi_{\mathbf{c}'})$. Since $\phi_{\mathbf{c}} = \iota_{\mathbf{c}}(\sigma_{\mathbf{c}})$ and $\phi_{\mathbf{c}'} = \iota_{\mathbf{c}'}(\sigma_{\mathbf{c}'})$ are both generators, i.e., not linear combinations, this equality in the free abelian group implies the equality of the underlying elements. That

is, $\partial_{U_{\mathbf{c}},\{a\}}(\sigma_{\mathbf{c}}) = \partial_{U_{\mathbf{c}'},\{a\}}(\sigma_{\mathbf{c}'})$. This is precisely the condition for a section, $\mathcal{F}(\mathbf{c} \trianglelefteq e)(\sigma_{\mathbf{c}}) = \mathcal{F}(\mathbf{c}' \trianglelefteq e)(\sigma_{\mathbf{c}'})$ from Definition 5.30. As such, $s = (\sigma_{\mathbf{c}})_{\mathbf{c}\in\mathbf{Cell}(\mathcal{S})}$ defines a section and, by Theorem 5.33, the task $\mathbf{T}$ is solvable.

■

Corollary 5.42 is an immediate consequence of framing task solvability in terms of cellular sheaves, and transforms the search for sections into a constructive procedure. First, we compute the kernel of the coboundary operator, which consists of solving a linear system over $\mathbb{Z}$. Second, we verify whether any element of the kernel has each of its components lying in the image of the generator embedding of Definition 5.36. Note that, since the task specification has finitely many simplices and the system slice have finitely many cells (Lemma 5.9), this verification consists of checking membership in a finite set for each 0-cell. While the kernel computation itself is efficient, the procedure involves searching through potentially many elements of the kernel.

We illustrate now this cohomology computation with the task sheaf for $\varepsilon$-agreement from Example 5.32.

▶ **Example 5.43 (Cohomology computation).** Consider the fragment of the task sheaf for $\varepsilon$-agreement with $\varepsilon = 0.5$ of Example 5.32. We compute explicitly the difference matrix and zeroth cohomology for 0-cells $\mathbf{c}_1 = (0_a, 1_b, \leftarrow)$, $\mathbf{c}_2 = (0_a, 1_b, \leftrightarrow)$, and 1-cell $e_1 = (\mathbf{c}_1, \mathbf{c}_2)_a$. Recall that the delivery patterns $\leftarrow$ and $\leftrightarrow$ indicate that in both global states agent $a$ received $b$'s message, while only in one did $b$ receive $a$'s. This means that $a$ cannot distinguish $\mathbf{c}_1$ and $\mathbf{c}_2$, witnessed by $e_1$, and the decision function must assign the same output for $a$ in both states. The stalks (Definition 5.23) at both 0-cells contain the valid outputs for the input $(0_a, 1_b)$ under $\varepsilon = 0.5$:

$$\mathcal{F}(\mathbf{c}_1)(\{a,b\}) = \mathcal{F}(\mathbf{c}_2)(\{a,b\}) = \left\{\binom{0}{0}, \binom{0}{0.5}, \binom{0.5}{0}, \binom{0.5}{0.5}, \binom{0.5}{1}, \binom{1}{0.5}, \binom{1}{1}\right\}$$

At the 1-cells, the stalk contains all possible local outputs for agent $a$:

$$\mathcal{F}(e)(\{a\}) = \{0, 0.5, 1\}$$

The 0-cells at the linearized task sheaf (Definition 5.35) have as stalks $\mathcal{L}(\mathbf{c}_i) = \mathbb{Z}[\mathcal{F}(\mathbf{c}_i)(\{a,b\})]$ the values obtained with the following seven generators (Defini-

tion 5.34)

$$g_1 = \binom{0}{0}\ g_2 = \binom{0}{0.5}\ g_3 = \binom{0.5}{0}\ g_4 = \binom{0.5}{0.5}\ g_5 = \binom{0.5}{1}\ g_6 = \binom{1}{0.5}\ g_7 = \binom{1}{1}$$

while the three generators of $\mathcal{L}(e) = \mathbb{Z}[\mathcal{F}(e)(\{a\})]$ are identified as

$$f_0 = 0\ f_{0.5} = 0.5\ f_1 = 1$$

The cochain groups (Definition 5.38) are then their amalgamation, expressed as

$$C^0(\mathcal{L}) = \mathcal{L}(\mathbf{c}_1) \oplus \mathcal{L}(\mathbf{c}_2) \cong \mathbb{Z}^7 \oplus \mathbb{Z}^7 \qquad C^1(\mathcal{L}) = \mathcal{L}(e) \cong \mathbb{Z}^3.$$

where the restriction maps $\mathcal{L}(\mathbf{c}_i \trianglelefteq e) = \mathbb{Z}[\partial_{\{a,b\},\{a\}}]$ project each output to the component of agent $a$, inducing the linear map sending each generator $g_j$ to the corresponding generator $f_k$:

$$\begin{aligned} &g_1 = \tbinom{0}{0} \mapsto f_0, \quad g_2 = \tbinom{0}{0.5} \mapsto f_0, \quad g_3 = \tbinom{0.5}{0} \mapsto f_{0.5}, \\ &g_4 = \tbinom{0.5}{0.5} \mapsto f_{0.5}, \quad g_5 = \tbinom{0.5}{1} \mapsto f_{0.5}, \quad g_6 = \tbinom{1}{0.5} \mapsto f_1, \quad g_7 = \tbinom{1}{1} \mapsto f_1. \end{aligned}$$

This can be expressed as a matrix with rows indexed by $\{f_0, f_{0.5}, f_1\}$ and columns indexed by $\{g_1, \ldots, g_7\}$, i.e.

$$R_a = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

We can derive now the coboundary operator (Definition 5.38) by orienting the edge as $\mathbf{c}_1 \to \mathbf{c}_2$, where it becomes $\delta^0 \colon C^0(\mathcal{L}) \to C^1(\mathcal{L})$ given by

$$\delta^0(\phi)_e = \mathcal{L}(\mathbf{c}_2 \trianglelefteq e)(\phi_{\mathbf{c}_2}) - \mathcal{L}(\mathbf{c}_1 \trianglelefteq e)(\phi_{\mathbf{c}_1}) = R_a\phi_{\mathbf{c}_2} - R_a\phi_{\mathbf{c}_1}.$$

whose corresponding difference matrix $D$, with columns indexed by $\{g_1^{\mathbf{c}_1}, \ldots, g_7^{\mathbf{c}_1}, g_1^{\mathbf{c}_2}, \ldots, g_7^{\mathbf{c}_2}\}$, is as follows.

$$D = \begin{pmatrix} -R_a & R_a \end{pmatrix} = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Note now that an element $\phi = (\phi_{\mathbf{c}_1}, \phi_{\mathbf{c}_2}) \in C^0(\mathcal{L})$ is in the kernel $\ker(D)$ if and only if $R_a\phi_{\mathbf{c}_1} = R_a\phi_{\mathbf{c}_2}$. Furthermore, a section corresponds to an element in the

kernel where both $\phi_{\mathbf{c}_1}$ and $\phi_{\mathbf{c}_2}$ are generators. For instance, for $\phi_{\mathbf{c}_1} = g_3 = \binom{0.5}{0}$ and $\phi_{\mathbf{c}_2} = g_4 = \binom{0.5}{0.5}$, both have $0.5$ as its $a$-component. As such,

$$R_a g_3 = R_a g_4 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = f_{0.5}.$$

which corresponds to $\phi \in C^0(\mathcal{L})$ being the vector with coefficient $1$ at positions $g_3^{\mathbf{c}_1}$ and $g_4^{\mathbf{c}_2}$, and zeros elsewhere. It can be verified directly then that $D\phi = 0$, confirming that $\phi \in \mathcal{H}^0(\mathcal{L})$. Since both components are generators, this element is in the image of $\iota$, and as such $(\binom{0.5}{0}, 0.5, \binom{0.5}{0.5})$ defines a valid local section to this piece of the task sheaf. ◀

The reformulation of Theorem 5.33 in terms of the cohomology of the linearized task sheaf in Corollary 5.42 gives us a constructive procedure for assessing task solvability.

▶ **Theorem 5.44 (Semidecidability of task solvability).** Let $\mathbf{T}$ be a task and $\mathbf{S}$ be a system with system frame $\mathcal{K}_{\mathbf{S}}$. The problem of determining whether there exists a terminating decision function solving $\mathbf{T}$ is semidecidable.

*Proof.* We describe a procedure that halts if and only if $\mathbf{T}$ is solvable, and then outputs a terminating decision function.

The procedure consists of enumerating execution cuts $A_1, A_2, \ldots$ in increasing depth, where depth is the maximum number of computational steps taken by any agent. For each cut $A_k$, we perform the following. First, we compute the causal closure $\mathrm{ccl}(A_k)$ (Definition 5.7) and construct the system slice (Definition 5.13). Second, we build the task sheaf $\mathcal{F} \colon \mathbf{Cell}(\mathcal{S}_{A_k}) \to \mathbf{Cset}$ (Definition 5.23) and linearize it as $\mathcal{L} \colon \mathbf{Cell}(\mathcal{S}_k) \to \mathbf{Ab}$ (Definition 5.35). Third, we use the restriction maps of $\mathcal{L}$ to assemble the coboundary operator $\delta^0 \colon C^0 \to C^1$ as a difference matrix $D$ over $\mathbb{Z}$ (Definition 5.38). Fourth, we compute a basis for $\ker(\delta^0)$, i.e., the zeroth cohomology group $\mathcal{H}^0(\mathcal{L})$ (Definition 5.39), by some method such as Smith normal form. Finally, for each basis element $\phi \in \mathcal{H}^0(\mathcal{L})$, we check whether $\phi_{\mathbf{c}} \in \mathrm{im}(\iota_{\mathbf{c}})$ for every $0$-cell $\mathbf{c} \in \mathrm{ccl}(A_k)$, where $\iota$ is the generator embedding of Definition 5.36. If such $\phi$ is found, let $\sigma_{\mathbf{c}}$ be the unique simplex with $\iota_{\mathbf{c}}(\sigma_{\mathbf{c}}) = \phi_{\mathbf{c}}$, which exists since $\phi_{\mathbf{c}} \in \mathrm{im}(\iota_{\mathbf{c}})$. We define then a decision function $d$ by $d(\pi_a(\mathbf{c})) = \partial_{U_{\mathbf{c}},\{a\}}(\sigma_{\mathbf{c}})$ for each agent $a \in U_{\mathbf{c}}$. This procedure then halts and outputs $d$.

In order to assert correctness, suppose $\mathbf{T}$ is solvable. By Theorem 5.33, there exists a terminal execution cut $A^*$ whose task sheaf $\mathcal{F}$ admits a section $s \in \Gamma(\mathcal{S}_{A^*}, \mathcal{F})$. As the enumeration eventually reaches $A^*$, Corollary 5.42 guarantees that $\iota(s) \in \mathcal{H}^0(\mathcal{L})$, where the image $\iota(s) \in \mathcal{H}^0(\mathcal{L})$ has each component $\iota_{\mathbf{c}}(s_{\mathbf{c}})$ in the image of $\iota_{\mathbf{c}}$, meaning that the procedure halts. The output $d$ is a terminating decision function by construction, as shown in Corollary 5.42. Consider now that the procedure halts with some $\phi$, in which case Corollary 5.42 provides a section, hence $\mathbf{T}$ is solvable by Theorem 5.33. ∎

Theorem 5.44 provides a constructive procedure that is guaranteed to provide a positive answer to task solvability in finite time, while a negative answer would require an unbounded search. The procedure described reduces the geometrical question of existence of sections to a linear algebra computation over $\mathbb{Z}$, where the kernel computation of the coboundary operator is efficient [Sto00] and captures all assignments that are consistent across adjacent cells in the system slice. This is followed by a generator embedding test, consisting of a finite membership check. The kernel elements passing this test are in bijection with the sections by Corollary 5.42, making it possible to synthesize a terminating decision function directly from each such element, as shown in Theorem 5.33.

Furthermore, because of the agent task sheaf decomposition shown to uniquely merge as a global task sheaf in Theorem 5.29, the cohomology condition for task solvability of Corollary 5.42 can be similarly reformulated to agent-wise conditions. Note that the coboundary operator decomposes along the agent indices of the 1-cells, and the generator embedding test can be performed independently for each agent component. As such, the computation described in Theorem 5.44 could be meaningfully done in a distributed manner, with local kernel computations composed into a global section when one exists. Such parallelization reflects the locality of the cellular sheaf structure, where the consistency constraints are local to an agent and verifying them only needs information local to the adjacent 1-cells. This mirrors the distributed nature of the original coordination problem, where agents must reach consistent decisions using only local information.

## 5.5 CONCLUSION

This chapter established a sheaf-theoretic framework for reasoning about distributed task solvability, transforming the question of whether a terminating decision function exists into whether local causal consistency constraints can be simultaneously satisfied. The framework complements the combinatorial topology approach [HKR14], where the connectivity of the execution state space remains a key property of analysis, by providing instead a compositional and computationally adapted perspective. Three results make this precise. The decomposition of Theorem 5.29 shows that task sheaves arise uniquely from agent-wise components, enabling a modular reasoning about individual constraints. The characterization of Theorem 5.33 identifies task solvability with the existence of global sections, a central object in sheaf theory. The cohomological reformulation of Corollary 5.42 and the semidecidability of Theorem 5.44 convert this existence question into a linear algebra problem with a constructive procedure.

By first defining a system slice (Definition 5.13), the properties of determinism and termination of a decision function are externalized into structural conditions. Constructed as a finite substructure of the system frame via an execution cut (Definition 5.3), the slice limits the scope of evaluation by a frontier where every execution must pass, while the causal closure (Definition 5.7) adds all global states whose local information could constraint decisions at the frontier, which in turn are encoded in a unified causal consistency relation (Definition 5.11). The resulting system slice naturally forms a cell category (Definition 5.17), providing the combinatorial domain required for the definition of cellular sheaves. The task specification is then restricted to the constraints pertaining individual global states, defining the space of decision functions (Definition 5.20), effectively separating the structural requirements of the system's execution from the data requirements of the task.

The task sheaf (Definition 5.23) formalizes the relationship between system and task as a cellular sheaf, assigning to each global state the possible decision values, with restriction maps enforcing the agent-wise consistency. This sheaf decomposes uniquely into agent task sheaves that can be defined independently beforehand and then composed (Theorem 5.29). These structures are the site for the main characterization result: a terminating decision function solving a task ex-

ists if and only if there exists a task sheaf with a global section (Theorem 5.33). That is, the task solvability question is transformed into a local-to-global consistency problem. By linearizing the task sheaf (Definition 5.35), the problem of section existence becomes the computation of the kernel of a coboundary operator over the integers (Corollary 5.42). This zeroth cohomology group captures precisely those assignments that satisfy all consistency constraints. Checking which kernel elements correspond to decisions yields a procedure to construct a terminating decision function when one exists, making task solvability a semidecidable problem (Theorem 5.44).

It should be noted that task sheaves developed here are limited to distributed tasks definable as a static relation of input and output values, where each execution leads to a final decision. Stabilizing tasks, where the output may vary within a valid set, would require relaxing the termination condition encoded in causal consistency, and generalizing the decision values to accommodate the intermediary stabilizing states. For context-dependent validity, such as with reactive systems, a sheaf of decision functions rather than decision values may be more appropriate. In such cases, the decision process would depend not only on agents but on their context, making the representability argument of Lemma 5.22 less direct unless it too is externalized into the base category[5].

The assumption of locally finite system slices also warrants attention, as it restricts the analysis to bounded executions defined by terminal cuts. While the execution cut enforces the finiteness required for computations, it excludes systems where global states may remain indistinguishable indefinitely, such as certain asynchronous ones. For those cases, additional assumptions such as bounded delays are required to ensure the existence of an informative finite cut. A separate concern arises from the computational complexity. The semidecidability of Theorem 5.44 relies on checking the generator embedding for elements of the kernel, and the stalks may grow exponentially with the number of agents according to the task specification. Exploiting the structure through representability (Lemma 5.22) and the agent-wise decomposition (Theorem 5.29) remains important for practical applications, while a proper complexity analysis is warranted for further

[5]A hint of this generalization is mentioned in Chapter 6 as chromatic semi-simplicial sets with decision values.

considerations.

The sheaf-theoretic framework introduced in this chapter opens multiple avenues for future work. A natural first direction is to lift the procedure of testing different system slices to a categorical level, seeing the choice of deeper execution cuts as morphisms between task sheaves. The sequence of inclusions between slices should induce a corresponding sequence of sheaves, and solvability may be expressible as a property of this sequence rather than of individual slices. The structure of the task sheaf could also incorporate higher-dimensional cells into the base cell category, which currently contains only $(n-1)$- and $0$-dimensional cells, corresponding to global state and agent-wise constraints. Since it was shown in [Gou+23] that higher-dimensional simplices encode forms of group knowledge in csets, incorporating these intermediary cells may capture constraints that emerge from the interaction of multiple agents, beyond pairwise consistency. Furthermore, it would be interesting to explore the flexibility offered by the space of decision functions in Definition 5.20, which requires only a simplicial map defining assignments.

Another direction comes from understanding the relationship with the other topological characterizations of distributed task solvability. The point-set topological approach of [NSW24] defines a distance between executions based on how long processes take to distinguish them, with consensus characterized by the disconnectedness of this space. This distance naturally induces a filtration on the execution space, where executions that diverge later are closer than those that diverge earlier. The sequence of system slices we construct, parametrized by the depth of execution cuts, mirrors this structure, where deeper cuts provide finer slices that can distinguish more states across executions. This suggests investigating how the cohomology of these sheaves evolves as the slices take later states. As the finer system slices should only gain in capacity to distinguish global states, seen analogously in Chapter 4 with the spatial knowledge persistence of Theorem 4.13, the obstructions present at a given depth can vanish, but not appear further along an execution. A question comes as whether this sequence stabilizes, or whether a vanishing cohomology at some finite depth could characterize solvability.

Finally, the spectral cellular sheaf theory developed since [Han20], where a sheaf Laplacian operator models diffusion of data across the sheaf, suggests a

quantitative approach to task solvability. Minimizing an appropriate sheaf Laplacian would correspond to finding assignments that best satisfy the consistency constraints, even when exact sections do not exist. This interpretation frames solvability as a distributed optimization problem, where a possible decision function is iteratively adjusted towards a consistent assignment, and distance between an exact solution and the closest approximate solution could quantify the "difficulty" of a task.

*"O passado não reconhece o seu lugar: está sempre presente."*

"The past does not recognize its place: it is always present."

— Mário Quintana, Caderno H [Qui73]

# 6

# ITERATED ALGEBRAS OF DISTRIBUTED PROTOCOLS

We have so far reasoned about multi-robot systems as static objects obtained from their possible behaviors: system frames in Chapter 4 define relations between the possible executions of a fixed protocol, while task sheaves in Chapter 5 define topological constraints on these executions. Both forget the generative process behind such structures, the executions and system runs from Chapter 3, and impose properties such as the knowledge persistence in communication rounds. This chapter provides an algebraic framework that unifies protocol specification, execution generation, and decision computation within a single categorical model. More precisely, this chapter describes a categorical model of iterated distributed protocols, defined through an endofunctor algebra subsuming multiple protocols in the literature. These algebras form models for a different temporal-epistemic logic, allowing us to reason about the knowledge throughout the iterations of distributed systems, including multi-robot systems.

The key insight lies in the modeling of communication protocols as structure-preserving transformations rather than operational procedures. The understanding of a protocol as a specification of the possible communication patterns across rounds of execution naturally takes the form of a functor, which maps each set

of participating agents to the (distributed) state space they can reach. While a single application of this protocol functor describes one iteration of communication, they can be endowed with an algebra structure in order to relate successive rounds, specifying how to embed the output of one round back into its input. The respective free algebra encapsulates any successive protocol applications, collecting all reachable states in a finite number of rounds. We further leverage the simplicial structure of csets and assume a similar epistemic interpretation of [Gou+23] as encoding the indistinguishability of global states, now with a temporal component provided by the free algebra. As such, the persistence of knowledge that was assumed as an axiom in models of system frames, now arises as a theorem about morphisms of semi-simplicial models (Theorem 6.29). This framework can also be extended beyond communication to concrete protocols that compute and update numerical data. By enriching the state space with decision values, protocol functors can be lifted to track explicit values where local decision functions specify how agents update them at each round. Decisions allow us to model robot protocols, and their interpretation as decision predicates connects back to the spatial task specifications from Section 4.3.

This chapter is adapted from [Gou+25], a joint work with Eric Goubault, Roman Kniazev, Jérémy Ledent and Sergio Rajsbaum.

**Related work** Our algebraic framework builds on the protocol complex approach to task solvability of Herlihy, Kozlov and Rajsbaum [HKR14], reformulating their geometric constructions on simplicial complexes as functors on chromatic semi-simplicial sets, which offer a better behaved categorical setting. We shift from an operational to a categorical perspective, where protocol complex (functors) arise as left Kan extensions of simpler functors on agent subsets, where Yoneda extensions [Mac71] generalize them to the entire distributed state space. The universal properties of free algebras then show that protocol iteration admits a canonical construction via varietors [Trn+75].

The connection between simplicial structures and epistemic logic was established by Goubault, Ledent and Rajsbaum [GLR21], where simplicial complex models are shown equivalent to Kripke models. Their framework extends to dynamic epistemic logic [vDHK07] through simplicial action models. We follow a similar path by interpreting temporal-epistemic formulas on free algebras, where

the iterative structure is embedded in the algebra rather than requiring an external operational semantics. In the case of immediate snapshot protocols, our temporal operators capture updates analogous to public announcements [BMS98]. We incorporate fault models similar to recent works [GLR22; Gou+23] providing an epistemic interpretation to crash failures. Dynamic networks [KO11] provide the main communication model that we generalize, recovering immediate snapshot as a special case.

The enrichment of chromatic semi-simplicial sets, or csets, with decision values follows from the topos-theoretic perspective [MM94], where csets enriched with decision values arise as a slice categories over decision assignments. This allows us to model concrete protocols where agents update numerical data, such as averaging for fault-tolerant coordination [TLV24]. The connection to multi-robot systems follows from Alcántara et al. [Alc+19], who show the equivalence between asynchronous luminous robot models and wait-free shared memory, revealing their common protocol topology.

**Chapter organization** Section 6.1 reviews the protocol complex approach in distributed computing and shows that they naturally give rise to protocol functors of chromatic semi-simplicial sets (Definition 6.6). Multiple examples instantiate this framework: immediate snapshot protocols, dynamic networks, and synchronous protocols with crash failures (Examples 6.9 to 6.12 and 6.14), showing how different communication models arise as instances of the same functorial structure.

Section 6.2 reviews endofunctor algebras as models for transition systems, leading to the definition of an algebra of protocol complex functors modeling the iterations of a distributed system. Those are proven to induce a natural algebra structure (Lemma 6.15) and to be varietors (Corollary 6.21), hence admitting free algebras that canonically capture all configurations reachable through unbounded iteration from any initial state, allowing to recast the simplicial task solvability condition in our framework (Example 6.22).

Section 6.3 defines a temporal-epistemic logic whose semantics is given by semi-simplicial models over free algebras (Definitions 6.25 and 6.27), where the knowledge gain property (Theorem 6.29) for positive epistemic formulas emerges directly from the categorical structure.

Section 6.4 shows that the chromatic semi-simplicial sets can be enriched with decision values via protocol functors with decisions (Definition 6.36), allowing us to model multi-robot systems (Example 6.37). Furthermore, local decision functions are also integrated into concrete protocols (Definition 6.40), capturing iterative decision updates, such as averaging protocols (Example 6.42). The algebraic model is then extended with decision predicates, providing exploration and gathering task specifications (Examples 6.45 and 6.46).

Concluding remarks and future directions are discussed in Section 6.5.

## 6.1 FROM TRANSITION SYSTEMS TO PROTOCOL FUNCTORS

Up to now, we extracted information from mathematical objects encoding the possible executions of a distributed multi-robot system. Chapter 3 provided the trajectories and their execution abstractions, and those gave rise to system frames, which enhanced with spatial information in Chapter 4 defined models of knowledge over time. The same frame construction was then lifted to task sheaves in Chapter 5, formally relating its causal and epistemic structures to task solvability. We now turn to the iterative process behind these execution structures, formalizing the one-step state transformation of a distributed system as an endofunctor algebra on a structured state space.

Endofunctor $\mathcal{F}$-algebras formalize algebraic transition systems, which we will later lift to the distributed setting.

▶ **Definition 6.1 ($\mathcal{F}$-algebra).** Let $\mathcal{F}\colon \mathbf{C} \to \mathbf{C}$ be an endofunctor on a category $\mathbf{C}$. An *$\mathcal{F}$-algebra* is a pair $(C, a)$, where $C \in \mathbf{C}$ is an object and $a\colon \mathcal{F}(C) \to C$ is a morphism in $\mathbf{C}$, called the *algebra structure* of $C$.

Let $(C, a)$ and $(C', a')$ be two $\mathcal{F}$-algebras. Then $f$ is a *morphism of $\mathcal{F}$-algebras* from $(C, a)$ to $(C', a')$ if and only if $f : C \to C'$ is a morphism in $\mathbf{C}$ and the following diagram commutes:

$$\begin{array}{ccc} \mathcal{F}(C) & \xrightarrow{a} & C \\ \downarrow{\scriptstyle \mathcal{F}(f)} & & \downarrow{\scriptstyle f} \\ \mathcal{F}(C') & \xrightarrow{a'} & C' \end{array}$$

◀

We start by noting that the standard deterministic transition system, where the transition function can be understood as a stepping operation that provides a new state for each state and action pair, in this algebraic structure.

▶ **Example 6.2 (Deterministic transition system as an $\mathcal{F}$-algebras).** A deterministic transition system consists of a set of states $S$, a set of actions $A$ and a transition function $\tau : S \times A \to S$ specifying what state $s' \in S$ is obtained from each pair $(s, a)$ of state and action.

The transition system $(S, A, \tau)$ can be seen as an $\mathcal{F}$-algebra $(S, a)$ over an endofunctor over the category of sets **Set** defined on objects as $\mathcal{F} : S \mapsto S \times A$. A function $a : S \times A \to S$ matches the transition function $\tau$ by assigning to each to a pair $(s, a)$ a state $s'$.

The algebra structure $(C, a)$ decomposes the *transition system* in a state space (the carrier object $C$) and its transition dynamics (the functor $\mathcal{F}$ and map $a$). The functor $\mathcal{F}$ sends a set of states to the possible pairs of states and actions, defining the signature of the possible operations. The map $a$ completes it by giving its interpretation, as it specifies what happens (which state is reached) if one does follow the transition from some pair of state and action. ◀

A distributed multi-robot system lifts this structure from sets to chromatic semi-simplicial sets. Rather than having states represent a single computational entity, they are an amalgamation of local states with enough structure to encode how they relate to each other. Analogously, the operation must now encode the local stepping operation of each robot. This leads us to lift this encoding from sets of states to chromatic semi-simplicial sets of states, as shown in Section 2.3.3 to be a useful algebraic encoding of such distributed structures.

Recall from Definition 2.35 that a chromatic semi-simplicial set, is a functor $X : \Gamma^{\mathrm{op}} \to \mathbf{Set}$, mapping subsets of agents[1] to sets of global states, also known as configurations of the system. For a set of agents $\mathcal{A}$, the category $\Gamma$ has as objects the subsets $U \subseteq \mathcal{A}$, and a single morphism $\delta_{U,V} : U \to V$, whenever $U \subseteq \mathcal{A}$. For each

[1]We maintain the generalization of the language introduced in Remark 5.1 to agents, which is specialized to either processes or robots, according to the applications that will follow.

subset of agents $U \in \Gamma$, the cset specifies the set of $U$-simplices $X(U)$ representing all global states where precisely the agents in $U$ participate. The cset defines face maps $\partial_{U,V} \colon X(V) \to X(U)$, from the inclusions $\delta_{U,V}$, where the sub global state $X(U)$ is extracted from $X(V)$ by forgetting the agents in $V \setminus U$. The set of all global states where only $n$ agents are present is denoted $X_n$, which refers to all $U$-simplices with cardinality $n$, i.e., $\{X(U) \mid U \subseteq \mathcal{A}, |U| = n\}$.

The chromatic semi-simplicial sets will serve as our distributed state space, as sets did in Example 6.2. Before defining the algebra structure that accompanies them, we need to specify the requirements of a distributed "stepping operator". This operator is the role of the protocol in distributed computing, the procedure that transforms sets of states through communication rounds. In the topological approach of Herlihy, Kozlov and Rajsbaum [HKR14], the protocol complex captures all possible resulting configurations of states after applying a communication protocol, while output values are obtained in a final decision layer (see Section 2.2 for a review of this approach). We lift this notion to a functorial setting, where protocols become structured transformations of csets.

We describe now the properties that a protocol is assumed to present when applied to a global state, which acts by providing all new global states it can jointly transition into.

▶ **Assumption 6.3 (Round-based communication).** *round-based* if it models exactly one discrete step of the protocol, during which every active agent sends and receives messages (or accesses a shared memory) exactly once, producing a new set of local states. For a functor $\mathcal{F} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ modeling a protocol, this means that $\mathcal{F}(U)$, for $U \in \Gamma$, represents the cset of all configurations reachable when agents in $U$ step one round. ◀

▶ **Assumption 6.4 (Oblivious communication).** A communication protocol is *oblivious* if it does not depend on the local states of the agents. A functor $\mathcal{F} : \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ is oblivious if its application to any subset of agents $U \subseteq A$, $\mathcal{F}(U)$, is entirely determined by which $U$ agents participate, making no reference to agents outside $U$. ◀

▶ **Assumption 6.5 (Wait-free communication).** A protocol is *wait-free* if it takes into consideration the possibility that only a subset of the active agents at a time will

participate in communication. That is, if $A$ is a set of active agents at a round of execution, but only $B \subseteq A$ participate, due to the others being too slow or having crashed, then the resulting configurations of applying the protocol in $A$ must include the effects of communication having been done only on the smaller set $B$.

For a cset $X \colon \Gamma^{\mathrm{op}} \to \mathbf{Set}$ and a functor $\mathcal{F} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ modeling the effect of a protocol on every set of active agents, wait-freedom means that the map $\mathcal{F}(\delta_{B,A}) \colon \mathcal{F}(B) \to \mathcal{F}(A)$ is a monomorphism, embedding configurations for participating agents $B$ into the space of configurations for all active agents $A$. In other words, the possible configurations for $\mathcal{F}(A)$ must be valid independently of non-participating agents, preserving inclusions. ◀

Assumptions 6.3 to 6.5 motivate the categorical structure of a protocol functor. A round-based protocol acts on one time step at a time. An oblivious protocol depends only on which agents participate, not on their specific states, suggesting a domain indexed by subsets of agents rather than state configurations. A wait-free protocol must produce valid outputs even when only a subset of agents participate, requiring that restrictions to smaller groups remain consistent. These considerations leads us to model a protocol as a functor $\mathcal{F} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$, where $\Gamma$ is the category of agent subsets. We now encode these three requirements into a single categorical definition.

▶ **Definition 6.6 (Protocol functor).** A *protocol functor* is a functor $\mathcal{F} : \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ that preserves monomorphisms. ◀

The preservation of monomorphisms encodes wait-freedom, while round-based and oblivious properties come from $\mathcal{F}$ being defined on $\Gamma$. The protocol functor assigns to each $U \in \Gamma$ a cset $\mathcal{F}(U)$ representing all possible combined sets of global states that may result from the protocol specification after one round.

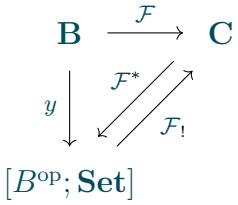
▶ **Lemma 6.7.** Every protocol functor satisfies Assumption 6.3 (round-based), Assumption 6.4 (obliviousness) and Assumption 6.5 (wait-freedom). ◀

*Proof.* Let $\mathcal{F} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ be a protocol functor and $A$ a set of agents. For Assumption 6.5, consider $B \subseteq A$ with the inclusion $\delta_{B,A}$. Since $\mathcal{F}$ preserves monomorphisms and every morphism in $\Gamma$ is monic, $\mathcal{F}(\delta_{B,A}) \colon \mathcal{F}(B) \to \mathcal{F}(A)$ is a monomorphism of csets. This means that each component $\mathcal{F}(\delta_{B,A})(V) \colon \mathcal{F}(B)(V) \to \mathcal{F}(A)(V)$

is injective, so configurations reachable by agents in $B$ embed into those reachable by agents in $A$, establishing wait-freedom. By construction, $\mathcal{F}(A)$ specifies the cset for agents in $A$ for exactly one round, satisfying Assumption 6.3. The functor is defined on sets of agents, ignoring non-participating agents and satisfying Assumption 6.4. ■

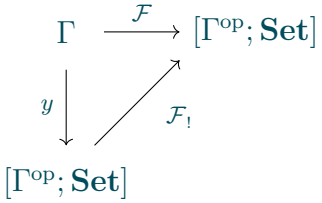
Note that the protocol functor operates on individual sets of agents. In order to apply it to the entire distributed state space in a cset, the functor $\mathcal{F}\colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ must be extended to the entire category of csets. This can be achieved by extending $\mathcal{F}$ along the Yoneda embedding, which sends each $U$ to the standard $U$-simplex $\Gamma[U]$, defined as $y\colon U \mapsto \hom_\Gamma(-, U)$. This corresponds to a Yoneda extension, a left Kan extension along the Yoneda embedding.

▶ **Proposition 6.8 (Existence of Yoneda extensions [Mac71, Section X.3]).** Let $\mathbf{B}$ be a small category, $\mathbf{C}$ be a category. Every functor $\mathcal{F} : \mathbf{B} \to \mathbf{C}$ defines a functor $\mathcal{F}^* : \mathbf{C} \to [B^{\mathrm{op}}; \mathbf{Set}]$ defined on the objects $c \in \mathbf{C}$ by $\mathcal{F}^*(c) = \hom_{\mathbf{C}}(\mathcal{F}-, c)$.




If $\mathbf{C}$ has small colimits, then $\mathcal{F}^*$ has a left adjoint $\mathcal{F}_! : [\mathbf{B}^{\mathrm{op}}; \mathbf{Set}] \to \mathbf{C}$, called the *Yoneda extension* of $\mathcal{F}$, which is the left Kan extension of $\mathcal{F}$ along the Yoneda embedding $y : \mathbf{B} \to [\mathbf{B}^{\mathrm{op}}; \mathbf{Set}]$. This extension $\mathcal{F}_!$ is the unique functor preserving colimits such that $\mathcal{F}_!(\mathbf{B}[b]) = \mathcal{F}(b)$ for all $b \in \mathbf{B}$. ◀

In our case, by taking $\mathbf{B} = \Gamma$ and $\mathbf{C} = [\Gamma^{\mathrm{op}}; \mathbf{Set}]$, the Yoneda extension $\mathcal{F}_!\colon [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ extends the protocol functor to all csets via the universal property expressed by the following commutative diagram.



The functor $\mathcal{F}_! \colon [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ is called the *protocol complex functor*. It maps any input cset $X$ to a protocol complex $\mathcal{F}_!(X)$, which represents all possible configurations reachable after one step of the protocol, starting from any configuration in $X$, generalizing the protocol complex construction of Definition 2.25 from chromatic simplicial complexes to csets. The protocol complex functor can be computed pointwise as a colimit for cset $X \colon \Gamma^{\mathrm{op}} \to \mathbf{Set}$ as

$$\mathcal{F}_!(X) \cong \mathrm{colim}_{(U,\sigma)\in \mathrm{el}(X)} \mathcal{F}(U)$$

where $\mathrm{el}(X)$ is the category of elements of the cset $X$, whose objects are pairs $(U, \sigma)$ for each object $U \in \Gamma$ and simplex $\sigma \in X(U)$, with morphisms $(U, \sigma) \to (V, \tau)$ given by inclusions $\delta_{U,V} \colon U \to V$ satisfying $\partial_{U,V}(\tau) = \sigma$.

Each simplex $\sigma \in X(U)$ represents a global state where agents in $U$ are active. The protocol functor $\mathcal{F}(U)$ provides all successor global states this set of agents reaches after one communication round. The colimit construction glues these local protocol outcomes while preserving the relations between the different global states present in $X$. More precisely, for each set $U \in \Gamma$, the set $\mathcal{F}_!(X)(V)$ contains all $U$-simplices reachable by applying the protocol to some configuration in $X$.

The protocol complex functor $\mathcal{F}_!$ provides a general framework for modeling iterated distributed protocols. We now instantiate protocols from the distributed computing literature, first with an abstract definition of their corresponding protocol functors, followed by their concrete realizations. We start with the asynchronous immediate snapshot (IS) model, a version of the shared memory model introduced in Section 2.2.1. We later generalize it to arbitrary dynamic networks and incorporate crash failures.

▶ **Example 6.9 (Protocol functor: asynchronous immediate snapshot).** Consider a shared memory system with processes $\Pi = \{p_1, \ldots, p_n\}$ that activate asynchronously in rounds and execute a full information immediate snapshot communication protocol. Each process $p_i$ has a private local state $v_i$, and communicates by performing a WRITE operation asynchronously in a shared memory at position $x_i^k$, for round $k$. As part of the immediate snapshot protocol, the array of each round starts clean, so to ensure that processes at different rounds do not interfere. Because the protocol is *full information*, each READ operation retrieves the entire shared memory of the corresponding round. That is, at round $k$, the process $p_i$ executes the following steps.

1. WRITE the local state $v_i$ in the shared memory, i.e., $x_i^k \leftarrow v_i$;
2. READ the entire shared memory, i.e., all values $x_1^k, \ldots, x_n^k$;
3. COMPUTE locally an update to its local state, i.e., $v_i \leftarrow f(x_1^k, \ldots, x_n^k)$.

A full information immediate snapshot protocol can be modeled as a functor $\mathcal{F}_{\text{IS}} \colon \Gamma \to [\Gamma^{\text{op}}; \mathbf{Set}]$ as follows. Let object $\mathcal{A} \in \Gamma$ be a set of active processes at some round. Then, $\mathcal{F}_{\text{IS}}(A)$ is a cset composed of sets of simplices for every $B = \{b_1, \ldots, b_m\} \subseteq A$ of participating processes (those not too slow or crashed). Its vertices are pairs $(b_i, v_i)$ where $v_i$ is the view of the shared memory by $b_i$. Its $B$-simplices are compatible sets of such views, i.e.,

$$\mathcal{F}_{\text{IS}}(\mathcal{A})(B) = \left\{ \{(b_0, v_0), \ldots, (b_m, v_m)\} \;\middle|\; \begin{array}{l} v_i \subseteq \mathcal{A} \\ \forall i,\ 0 \leqslant i \leqslant m,\ b_i \in v_i \\ \forall i, j,\ 0 \leqslant i, j \leqslant m,\ (v_i \subseteq v_j) \text{ or } (v_j \subseteq v_i) \\ \forall i, j,\ 0 \leqslant i, j \leqslant m,\ b_i \in v_j \Rightarrow v_i \subseteq v_j \end{array} \right\} \tag{6.1}$$

and such that the inclusions of participating processes $B' \subseteq B$ are respected, i.e.,

$$\begin{array}{rlcl} \mathcal{F}_{\text{IS}}(\mathcal{A})(\delta_{B',B}) : & \mathcal{F}_{\text{IS}}(\mathcal{A})(B) & \longrightarrow & \mathcal{F}_{\text{IS}}(\mathcal{A})(B') \\ & \{(b_0, v_0), \cdots, (b_m, v_m)\} & \longmapsto & \{(b_i, v_i) \mid b_i \in B'\} \end{array}$$

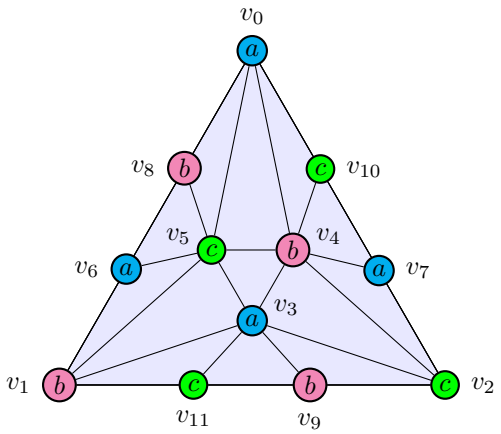
The functor $\mathcal{F}_{\text{IS}}$ is defined on inclusions of active processes $\mathcal{A}' \subseteq \mathcal{A}$, the morphism $\delta_{\mathcal{A}',\mathcal{A}} \in \Gamma$ as follows.

$$\begin{array}{rlcl} \mathcal{F}_{\text{IS}}(\delta_{\mathcal{A}',\mathcal{A}})(B) : & \mathcal{F}_{\text{IS}}(\mathcal{A}')(B) & \longrightarrow & \mathcal{F}_{\text{IS}}(\mathcal{A})(B) \\ & \{(b_0, v_0), \cdots, (b_m, v_m)\} & \longmapsto & \{(b_0, v_0), \ldots, (b_m, v_m)\} \end{array}$$

From Proposition 6.8, the protocol functor $\mathcal{F}_{\text{IS}} \colon \Gamma \to [\Gamma^{\text{op}}; \mathbf{Set}]$ can be extended to a protocol complex functor $\mathcal{F}_{!\text{IS}} \colon [\Gamma^{\text{op}}; \mathbf{Set}] \to [\Gamma^{\text{op}}; \mathbf{Set}]$ describing one round of the full information immediate snapshot for the entire distributed state space given by an input cset $X$. More concretely, for each simplex $\sigma$ of $X$ with participating processes $U \subseteq \mathcal{A}$, $\mathcal{F}_{!\text{IS}}$ creates one copy of $\mathcal{F}_{\text{IS}}(U)$ and glues all copies according to the face maps of the $X$. ◀

We show now a specific instantiation of the functorial definition with its execution by three agents.

▶ **Example 6.10 (Immediate snapshot with three processes).** Consider three processes $\Pi = \{a, b, c\}$ executing the full information immediate snapshot protocol with initial states $x_a^0 = \{a\}$, $x_b^0 = \{b\}$ and $x_c^0 = \{c\}$. The protocol is modeled as the functor $\mathcal{F}_{\text{IS}}$ from Example 6.9. The figure below shows the cset $\mathcal{F}_{\text{IS}}(A)$ from applying the protocol to the global state corresponding to $A$. The 12 vertices $v_0, \ldots, v_11$ are the values obtained from Equation (6.1), while the 13 triangles correspond to the sets of compatible views among the vertices.




$v_0 = (a, \{a\})$ $\quad$ $v_6 = (a, \{a, b\})$
$v_1 = (b, \{b\})$ $\quad$ $v_7 = (a, \{a, c\})$
$v_2 = (c, \{c\})$ $\quad$ $v_8 = (b, \{a, b\})$
$v_3 = (a, \{a, b, c\})$ $\quad$ $v_9 = (b, \{b, c\})$
$v_4 = (b, \{a, b, c\})$ $\quad$ $v_{10} = (c, \{a, c\})$
$v_5 = (c, \{a, b, c\})$ $\quad$ $v_{11} = (c, \{b, c\})$

Each triangle corresponds to one possible execution of the system, where the operations WRITE and READ of each process are intercalated respecting the immediate snapshot constraints. For instance, triangle $\{v_3, v_4, v_5\}$ represents any sequence of three WRITE operations, followed by any sequence of three READ operations, all obtaining the same full view. Triangles with growing views, such as $\{v_0, v_5, v_8\}$, correspond to a sequential execution where process $a$ performs a WRITE followed by a READ, obtaining no new information, process $b$ does the same sequence without interruption and acquires the shared value of $a$ and itself, and likewise process $c$ obtains the values of $a$, $b$ and itself.

Note that for every subset of participating processes $A' \subseteq A$, $\mathcal{F}_{\text{IS}}(A')$ is also represented in the figure as a sub-cset of $\mathcal{F}_{\text{IS}}(A)$, captured by the morphism of csets $\mathcal{F}_{\text{IS}}(\delta_{A',A}) : \mathcal{F}_{\text{IS}}(A') \to \mathcal{F}_{\text{IS}}(A)$. For instance, for $A' = \{a, b\}$, $\mathcal{F}_{\text{IS}}(A')$ consists of the vertices $\{v_0, v_1, v_6, v_8\}$ and the three edges between them. ◀

The immediate snapshot of Example 6.9 is an instance of a broader class of protocol functors within our framework, that of dynamic networks [KO11]. Here, the communication channels between agents may change (in directionality or existence) at each round of execution of a protocol. Recall that this is modeled

through time-varying communication graph, as in Definition 2.13, chosen by a message adversary $M_U$ that defines the set of possible graphs for each subset of agents $U \subseteq \mathcal{A}$. This allows to consider scenarios where agents have crashed and the communication graphs are relevant only to those that remain. We cast now the dynamic network model as a protocol functor.

▶ **Example 6.11 (Protocol functor: oblivious dynamic network).** Consider a dynamic network system with agents $\mathcal{A}$, specified by a family of oblivious message adversaries $(M_U)_{U\subseteq\mathcal{A}}$ for each subset of agents $U \subseteq \mathcal{A}$. Each $M_U$ defines the set of possible communication graphs when agents $U$ are active, modeling time-varying connectivity. Recall from Definition 2.13 that the view obtained by agent $a$ in communication graph $G$ is denoted $\mathrm{view}_G(a)$, while $\mathrm{active}(G)$ denotes the set of active agents.

The protocol functor $\mathcal{F}_{\mathrm{DN}} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ associates to each object $U$ of $\Gamma$ (a possible group of agents) a cset $\mathcal{F}_{\mathrm{DN}}(U) : \Gamma^{\mathrm{op}} \to \mathbf{Set}$. The cset $\mathcal{F}_{\mathrm{DN}}(U)$ defines sets of $V$-simplices, each representing one possible global state of a set of active agents $V \subseteq U$, as follows.

$$\mathcal{F}_{\mathrm{DN}}(U)(V) = \begin{cases} \varnothing & \text{if } V \not\subseteq U \\ \{\{(v, \mathrm{view}_G(v)) \mid v \in V\} \mid G \in M_{U'} \text{ for} & \text{otherwise} \\ \quad \text{some } U' \subseteq U, \text{ and } V \subseteq \mathrm{active}(G)\} & \end{cases} \tag{6.2}$$

Where each simplex corresponds to one possible communication pattern where, among agents $U$ under consideration, $U'$ are the agents that may communicate according to the message adversary that chose $M_{U'}$, and $V \subseteq U'$ are the agents that actually participate in this round. The set is empty if it exceeds the agents in $U$.

For each map $V \subseteq W$ of active agents, the following face maps are defined in the cset $\mathcal{F}_{\mathrm{DN}}(U)$. These maps forget agents while preserving their communication patterns, i.e., agents in $V$ have the same views that they had when $W$ was active. Note that they are well-defined by transitivity of active agents in sub-communication graphs.

$$\begin{array}{rcccc} \mathcal{F}_{\mathrm{DN}}(U)(\delta_{V,W}) & : & \mathcal{F}_{\mathrm{DN}}(U)(W) & \longrightarrow & \mathcal{F}_{\mathrm{DN}}(U)(V) \\ & & \{(w, \mathrm{view}_G(w)) \mid w \in W\} & \longmapsto & \{(v, \mathrm{view}_G(v)) \mid v \in V\} \end{array}$$

We look now at the possible morphisms between sets of agents in $\Gamma$, $\delta_{U,T} : U \hookrightarrow T$, which are mapped to morphisms of csets $\mathcal{F}_{\text{DN}}(\delta_{U,T}) \colon \mathcal{F}_{\text{DN}}(U) \to \mathcal{F}_{\text{DN}}(T)$, defined as a family of functions as follows.

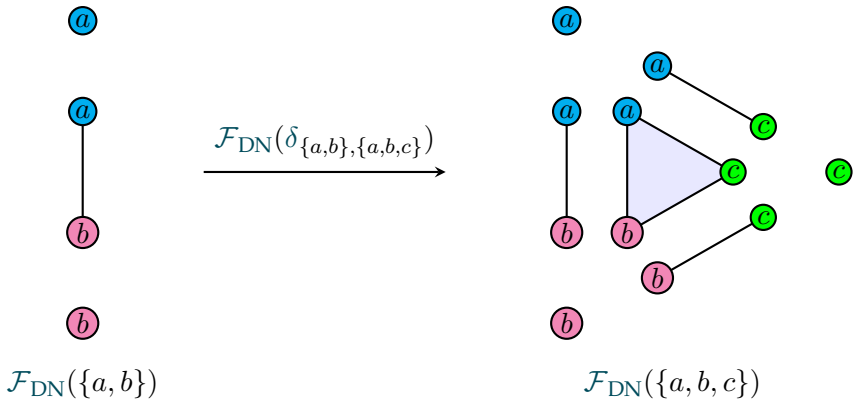
$$\begin{array}{lcccc} \mathcal{F}_{\text{DN}}(\delta_{U,T})(V) & : & \mathcal{F}_{\text{DN}}(U)(V) & \longrightarrow & \mathcal{F}_{\text{DN}}(T)(V) \\ & & \{(v, \text{view}_G(v)) \mid v \in V\} & \longmapsto & \{(v, \text{view}_G(v)) \mid v \in V\} \end{array}$$

Where these are well-defined because any communication graph $G \in M_{U'}$ for $U' \subseteq U \subseteq T$ remains valid when considering the larger set $T$. ◀

We illustrate now the protocol functor for dynamic networks in Example 6.11 using three agents in a reliable network, where no agents may crash and no communication messages may be lost.

▶ **Example 6.12 (Dynamic network with three agents).** Consider three agents $\mathcal{A} = \{a, b, c\}$ executing a reliable dynamic network protocol. The fact that the protocol is reliable translates to a message adversary that defines, for each subset of agents $U \subseteq \mathcal{A}$, a single communication graph $M_U = K_U$, the complete graph over $U$ vertices. Note that, even though the protocol does not allow for crashes or messages being lost, in a single possible execution, the fact that we model wait-free protocols, as per Assumption 6.5, forces us to consider the different subsets of agents that may be active. This requirement can be seen in the definition of the protocol functor $\mathcal{F}_{\text{DN}}(U)$ on an object $U$ (Equation (6.2)), which chooses a subset $U' \subseteq U$ of agents, and only those reliably communicate their views, which are collected by the other active agents.

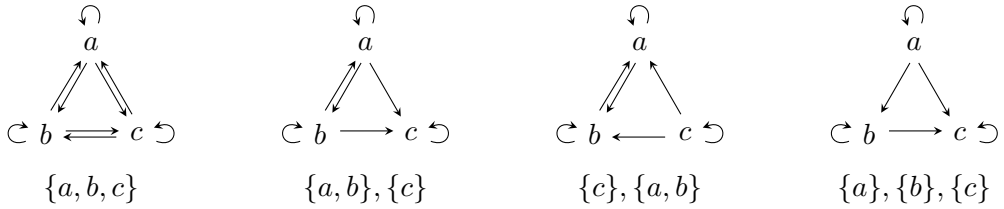
Concretely, rather than a single simplex, an application of $\mathcal{F}_{\text{DN}}$ gives us one maximal simplex for each subset $U' \subseteq U$. The figure below shows the morphism $\mathcal{F}_{\text{DN}}(\delta_{\{a,b\},\{a,b,c\}}) : \mathcal{F}_{\text{DN}}(\{a, b\}) \to \mathcal{F}_{\text{DN}}(\{a, b, c\})$, an inclusion of csets.




◀

▶ **Remark 6.13 (Immediate snapshot as a complete dynamic network).** The immediate snapshot appears then as the special case of the dynamic network model, where the possible communication graphs of a set of agents $U$ form an ordered partition. That is, for a set of agents $U \subseteq \mathcal{A}$, its ordered partition is a sequence of subsets $S = (S_1, \ldots, S_k)$, such that $\cup_i S_i = U$ and $S_i \cap S_j = \varnothing$ for all $i \neq j$. The communication graph $G_S$ associated to an ordered partition $S$ has an edge $a \to b$, for $a, b \in U$ whenever $a \in S_i$ and $b \in S_j$ with $i \leq j$.

The immediate snapshot model is then cast as a dynamic network with message adversary $M_U = \{G_S \mid S \text{ is an ordered partition of } U\}$. The figure below shows four of the thirteen possible communication graphs for agents $\mathcal{A} = \{a, b, c\}$, alongside their respective ordered partitions.



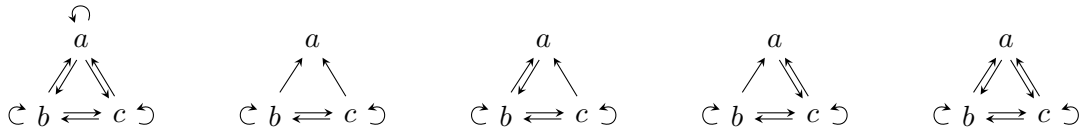

◀

The examples so far have shown how to model wait-free, asynchronous systems with dynamic networks as protocol functors, without explicitly handling failures yet. We now close this gap with the following example, which shows how our framework naturally extends to model crashes with restrictions to the message adversaries.

▶ **Example 6.14 (Synchronous fault-tolerant dynamic network).** Consider a dynamic network system with agents $\mathcal{A} = \{a_1, \ldots, a_n\}$ that communicate via synchronous message-passing. Each agent $a_i$ has a private local state $v_i$, an output buffer $x_i^i$, and an input buffer $x_i^j$, for $j \in \{1, \ldots, \widehat{i}, \ldots, n\}$. Note that the input and output buffers of an agent form a single array of $n$ values. At every round $k$, with communication graph $G_k$, agent $a_i$ executes the following:

1. SEND the values contained in its output buffer $x_i^i$ to all other agents $a_j$, i.e., $x_j^i \leftarrow x_i^i$, for every $a_i \to a_j$;
2. RECEIVE the data that has reached it from each other agent and store it in its input buffer, i.e., $x_i^j \leftarrow x_j^i$, for every $a_j \to a_i$, and $x_i^j \leftarrow \perp$ otherwise;
3. COMPUTE locally an update to its local state, i.e., $v_i \leftarrow f(x_i^1, \ldots, x_i^n)$, for some computable function $f$.

The fact that the system is synchronous means that agents wait until all SEND operations are complete before continuing to their next round. We consider also crash failures, which may happen at any point during the round, before or after sending any amount of messages, and are permanent, i.e., an agent remains crashed in the following rounds. In a synchronous system, crashes can be detected by not receiving a message at the end of the round from an agent that executed a SEND in the previous round. A crashed agent $a_i$ is represented in the communication graph by the absence of its reflexive edge $a_i \to a_i$. See below five possible communication graphs for agents $\mathcal{A} = \{a, b, c\}$ with message adversary $M_{\{a,b,c\}} \supset \{G_1, G_2, G_3, G_4, G_5\}$.



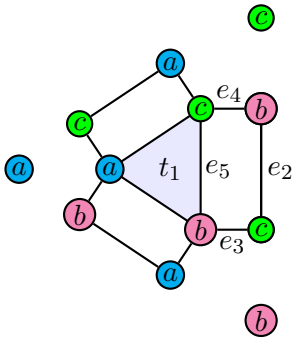

$G_1$ $G_2$ $G_3$ $G_4$ $G_5$

Communication graph $G_1$ shows one round of execution where no crashes occurred and all messages were successfully delivered. All other four graphs display agent $a$ as the single crashed agent after sending $0$ messages ($G_2$), $1$ message

($G_3$ and $G_4$), and 2 messages ($G_5$). Note that in the latter case of $G_5$, agent $a$ finished all of its SEND phase. This message adversary is defined for any subset of agents $U \subseteq \mathcal{A}$ by constraining all agents that are not crashed to execute all SEND operations. That is,

$$M_U = \{G \subseteq U \times U \mid \text{for all } a \in U,\ a \to a \text{ implies that for all } b \in U,\ a \to b\}$$

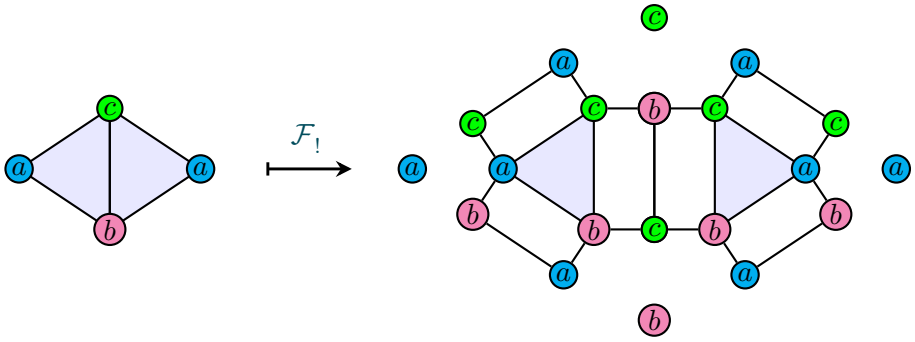
We consider now the synchronous dynamic network protocol with crash faults as a protocol functor $\mathcal{F}_{\text{CF}} : \Gamma \to [\Gamma^{\text{op}}; \mathbf{Set}]$. The figure below shows the cset obtained for $\mathcal{F}_{\text{CF}}(\{a, b, c\})$.




The triangle $t_1$ corresponds to the execution through the communication graph $G_1$, where all three vertices obtain the same view $\{a, b, c\}$ because there were no crashes. Each of the edges $e_2$, $e_3$, $e_4$ and $e_5$, correspond to the executions through graph $G_2$, $G_3$, $G_4$ and $G_5$, respectively, where agent $a$ crashes and does not belong to the configurations anymore. Similarly, all isolated vertices represent the single possible global state (in fact a local state) where two agents have crashed. Note that, as the crash in $G_5$ happened after all SEND operations were executed, it is indistinguishable[2] to the crashless execution.

Consider now the extension of $\mathcal{F}_{\text{CF}}$ to $\mathcal{F}_{!\text{CF}} : [\Gamma^{\text{op}}; \mathbf{Set}] \to [\Gamma^{\text{op}}; \mathbf{Set}]$ by Proposition 6.8, defined on csets rather than sets of agents. The figure below depicts its application on an initial cset composed of two triangles (two global states) glued along their $bc$-colored edge, where $a$ has two possible initial states that are compatible with those of $b$ and $c$. The cset on the right can be identified as two copies of the previous figure glued along their common edges and vertices.

[2]The consequences of interpreting such relations between simplices as epistemically indistinguishable is related to epistemic logic, as seen in Section 2.3.2, and will be further developed in Section 6.3.




◀

The protocol functors defined above provide the stepping operators needed for formalizing distributed transitions systems in an algebraic setting. An input cset $X$ equipped with an algebra structure $a\colon \mathcal{F}_!(X) \to X$ forms a distributed analog of Example 6.2. While the cset $X$ constitutes the distributed state space, the protocol complex functor $\mathcal{F}_!$ generates all possible next global states, and the algebra map $a$ relates the global states across subsequent rounds.

We formalize next these algebraic structures and the properties of their iterations, recovering the execution of a distributed system.

## 6.2 ALGEBRAS OF PROTOCOLS

Similar to how the deterministic transition systems shown in Example 6.2 induce an algebra $(S, a\colon S \times A \to S)$, the protocol complex functor can be equipped with an algebra structure in order to model the distributed iteration of a protocol. Given a compatible cset $X\colon \Gamma^{\mathrm{op}} \to \mathbf{Set}$, a protocol complex functor $\mathcal{F}_!\colon [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ induces an algebra carried by $X$ that encodes how to generate and select possible successor global states. Note how the state space is now encoded in a cset, enjoying of extra structure that allows us to model the distributed global state as glued local states.

While any morphism $a\colon \mathcal{F}_!(X) \to X$ induces an $\mathcal{F}_!$-algebra, for some cset $X$, we can construct a canonical natural transformation $\varepsilon\colon \mathcal{F}_! \to \mathrm{id}$ that maps uniformly the distributed state space at the image of a protocol complex functor, i.e., after one round, back to its starting point. Given a cset $X$, the algebra map $\varepsilon_X\colon \mathcal{F}_!(X) \to X$ formally relates the effects of a communication protocol

through an iteration. This mapping is done by "forgetting" the protocol details and remembering only which agents participated in the communication.

▶ **Lemma 6.15 (Protocol complex functor induces an algebra).** Let $\mathcal{F}\colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ be a protocol functor and let $\mathcal{F}_! : [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ be its Yoneda extension. There exists a natural transformation $\varepsilon\colon \mathcal{F}_! \to \mathrm{id}[\Gamma^{\mathrm{op}}; \mathbf{Set}]$, such that for every cset $X$, the pair $(X, \varepsilon_X)$ is an $\mathcal{F}_!$-algebra. ◀

*Proof.* We prove first that, for any $U \in \Gamma$, we can construct a canonical morphism of csets $f_U : \mathcal{F}(U) \to \Gamma[U]$, from a $U$-simplex to the standard $U$-simplex. This morphism encodes forgetting the protocol complex structure obtained after one round while maintaining which agents participated. For each $U \in \Gamma$, define $f_U : \mathcal{F}(U) \to \Gamma[U]$ as the function, for every $V \subseteq U$, $f_U(V)$ from the set $\mathcal{F}(U)(V)$ of $V$-simplices of $\mathcal{F}(U)$ to the $V$-simplices of $\Gamma[U]$. By definition, $\Gamma[U]$ is the Yoneda embedding functor $y : \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$, and so $\Gamma[U](V) = \hom_\Gamma(V, U)$, which is a singleton as $\Gamma$ is posetal. As such, there is only one choice, making it easy to verify that $f_U$ respects restriction maps and is indeed a morphism of csets.

We show now that all such morphisms $f_U$ are natural in $U$, i.e., the collection of morphisms $\{f_U : \mathcal{F}(U) \to \Gamma[U]\}_{U \in \Gamma}$ forms a natural transformation $f : \mathcal{F} \Rightarrow y$. Consider any inclusion $\delta_{U,U'} : U \to U'$ in $\Gamma$. We have

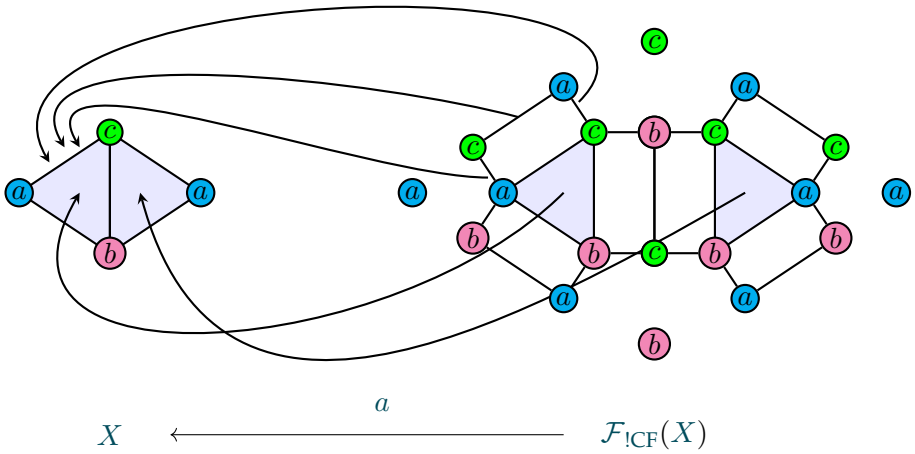
$$\begin{array}{ccc} \mathcal{F}(U) & \xrightarrow{f_U} & \Gamma[U] \\ \downarrow{\scriptstyle \mathcal{F}(\delta_{U,U'})} & & \downarrow{\scriptstyle y(\delta_{U,U'})} \\ \mathcal{F}(U') & \xrightarrow{f_{U'}} & \Gamma[U'] \end{array}$$

which commutes because $\Gamma[U](V)$, for any $V \subseteq U'$, is a singleton and any two functions into it are equal, thus the two paths above must be the same. As such, $f : \mathcal{F} \Rightarrow y$ is natural.

Having established that $f$ is natural in $U$, we consider now the extension of $f : \mathcal{F} \Rightarrow y$ from standard $U$-simplices to all csets with a Yoneda extension as in Proposition 6.8. Since $\mathcal{F}_! = \mathrm{Lan}_y(\mathcal{F})$ and $\mathrm{Lan}_y(y) \cong \mathrm{id}_{[\Gamma^{\mathrm{op}}; \mathbf{Set}]}$, by the universal property of the left Kan extension, there is a unique natural transformation $\varepsilon : \mathcal{F}_! \Rightarrow \mathrm{id}$ whose restriction along $y$ is $f$, i.e., $f = \varepsilon \circ y$. Since $\varepsilon$ exists with components $\varepsilon_X$, for $X \in [\Gamma^{\mathrm{op}}; \mathbf{Set}]$, the pair $(X, \varepsilon_X)$ defines an $\mathcal{F}_!$-algebra. ■

The algebra structure $(X, \varepsilon_X)$ defines then a distributed transition system in our categorical framework. The protocol complex functor generates all possible successor global states after one round of communication, while the map $\varepsilon_X : \mathcal{F}_!(X) \to X$ projects those back to states at the beginning of the round. As such, the algebra map tracks how the states relates across rounds, abstracting away the internal functioning of the communication protocol.

▶ **Example 6.16 (Crash fault protocol complex functor algebra).** Consider the protocol complex functor $\mathcal{F}_{!\text{CF}} \colon [\Gamma^{\text{op}}; \mathbf{Set}] \to [\Gamma^{\text{op}}; \mathbf{Set}]$ for a synchronous protocol with crash faults, and the cset $X$ with two triangles below, both from Example 6.14. In the $\mathcal{F}_{!\text{CF}}$-algebra $(X, a)$, the map $a \colon \mathcal{F}_{!\text{CF}}(X) \to X$ is a morphism of csets sending simplices in $\mathcal{F}_{!\text{CF}}(X)$ to simplices in $X$, such that the face maps are respected. While the protocol complex functor generates the distributed state space after one round of execution, the map $a$ connects the simplices (the distributed states) to their operational specification. Some of the arrows in $a$ are depicted in the figure below. Note that the two triangles in $\mathcal{F}_{!\text{CF}}(X)$ are mapped to their respective ones in $X$, while all sets of three edges with two agents are collapsed into a single originating edge.




◀

We consider now the executions of such algebraic distributed systems. Just as system runs, solution sets, system frames and task sheaves defined execution structures for the dynamical, computational (Chapter 3), logical (Chapter 4) and

topological (Chapter 5) abstractions of multi-robot systems, we now make use of free algebras as their analogue for our current algebraic abstraction. The main insight is that an $\mathcal{F}_!$-algebra defines a single step of execution that can be arbitrarily iterated.

▶ **Definition 6.17 (Free $\mathcal{F}$-algebra).** Let $\mathcal{F}\colon \mathbf{C} \to \mathbf{C}$ be an endofunctor and $I$ be an object in $\mathbf{C}$. An $\mathcal{F}$-algebra $(I^\infty, \phi_I)$ together with a morphism $\eta\colon I \to I^\infty$ is a *free $\mathcal{F}$-algebra generated by $I$* if the following holds: for any $\mathcal{F}$-algebra $(Q, q)$ and any morphism $f\colon I \to Q$, there is a unique morphism of $\mathcal{F}$-algebras $f^*\colon (I^\infty, \phi_I) \to (Q, q)$ such that $f = f^* \circ \eta$, i.e., the following diagram commutes.

$$\begin{array}{ccccc} \mathcal{F}(I^\infty) & \xrightarrow{\phi_I} & I^\infty & \xleftarrow{\eta} & I \\ \downarrow{\scriptstyle \mathcal{F}(f^*)} & & \downarrow{\scriptstyle f^*} & \swarrow{\scriptstyle f} & \\ \mathcal{F}(Q) & \xrightarrow{q} & Q & & \end{array}$$

The universal property of free algebras ensures that $(I^\infty, \phi_I)$ is initial among $\mathcal{F}$-algebras carried by $I$. ◀

Free algebras formalize the unbounded iteration of a protocol starting from an initial configuration given by a cset $\mathcal{I}$.

▶ **Example 6.18 (Free algebra for deterministic transition system).** Consider a deterministic transition system and the induced $\mathcal{F}$-algebra $(S, \tau)$, as seen in Example 6.2. Its free $\mathcal{F}$-algebra $(S^\infty, \phi_S)$, generated by $S$, is given by the formula

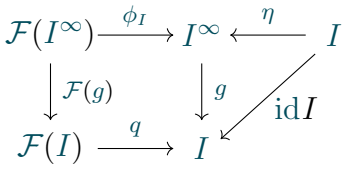
$$S^\infty = S \cup \bigcup_{n>0} S \times A^n$$

where each component $S \times A^n$ represents an execution trace of length $n$.

The morphism $\phi_S\colon \mathcal{F}(S^\infty) \to S^\infty$, that is, $\phi_S\colon S^\infty \times A \to S^\infty$, acts on a pair $((s, (a_1, \ldots, a_n)), a)$ of a state $s \in S$, a sequence of actions $(a_i)_{i \le n} \in A^n$, and a new action $a$, by appending it to the sequence, i.e.

$$\phi_S\colon ((s, (a_1, \ldots, a_n)), a) \mapsto (s, (a_1, \ldots, a_n, a))$$

We can apply this construction to any set of initial states $I$ through an $\mathcal{F}$-algebra, with $q\colon \mathcal{F}(I) \to I$. By the universal property of the free algebra, there exists a unique $\mathcal{F}$-algebra morphism $g\colon (I^\infty, \phi_I) \to (I, q)$ such that the following diagram commutes.




The morphism $g$ can be decomposed on each component $\mathcal{F}^i(I)$ as follows.

$$g_i = q \circ \mathcal{F}(q) \circ \cdots \circ \mathcal{F}^{i-1}(q)\colon \mathcal{F}^i(I) \to I$$

which become then our append operation iterated $i$ times for the deterministic transition system above,

$$g_i((s, (a_1, a_2, \ldots, a_i))) = q(q(\ldots q(q(s, a_1), a_2), \ldots, a_i))$$

◀

Example 6.18 shows the usefulness of the free algebra as a tool to iterate the transition function, as it provides the canonical algebra that extends the original set of states with all states that are reachable by taking a finite but unbounded number of transitions.

Trnková et al. [Trn+75] show that sufficiently well-behaved endofunctors $\mathcal{F}\colon \mathbf{C} \to \mathbf{C}$ admit free algebras generated by any object $I \in \mathbf{C}$. Such endofunctors are called *varietors*, whose existence is characterized as follows.

▶ **Proposition 6.19 (Sufficient conditions for the existence of free $\mathcal{F}$-algebras [Trn+75]).** Let $\mathcal{F} : \mathbf{C} \to \mathbf{C}$ be an endofunctor that commutes with colimits and preserves monomorphisms. Then, for any object $I \in \mathbf{C}$, there exists a free $\mathcal{F}$-algebra generated by $I$, given as follows. Let $I^\infty$ be the infinite coproduct:

$$I^\infty = I \coprod \mathcal{F}(I) \coprod \ldots \coprod \mathcal{F}^n(I) \coprod \mathcal{F}^{n+1}(I) \ldots$$

As $\mathcal{F}$ commutes with colimits, $\mathcal{F}(I^\infty)$ is isomorphic to $\coprod_{i \geqslant 1} \mathcal{F}^i(I)$, and we denote by $\phi_I$ the map induced by the identities on each $\mathcal{F}(\mathcal{F}^{n-1}(I)) = \mathcal{F}^n(I) \subseteq \mathcal{F}(I^\infty)$ to $\mathcal{F}^n(I) \subseteq I^\infty$. ◀

These conditions are in fact satisfied by our protocol complex functors.

▶ **Lemma 6.20.** Every protocol complex functor $\mathcal{F}_! : [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ preserves both monomorphisms and colimits. ◀

*Proof.* We prove the preservation of each property separately. First, note that by Proposition 6.8, the protocol complex functor $\mathcal{F}_!$ is the left Kan extension of $\mathcal{F}$ along the Yoneda embedding $y : \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$. It is known that left Kan extensions, as they are left adjoints, preserve all colimits.

Now, particularly for the left Kan extension $\mathcal{F}_!$ of a functor $\mathcal{F} : \mathbf{C} \to \mathcal{E}$ with $\mathcal{E}$ a topos and $\mathcal{F}$ preserving a class of finite limits, then $\mathcal{F}_!$ preserves those limits (see [MM94, Chapter X]). This result applies to our case, with $\mathcal{E} = [\Gamma^{\mathrm{op}}; \mathbf{Set}]$. Note that monomorphisms are characterized by certain pullback diagrams, hence they are a specific type of finite limits. By Definition 6.6, protocol functors $\mathcal{F}$ preserve monomorphisms (they send inclusion maps $\delta_{U,V} : U \hookrightarrow V$ in $\Gamma$ to inclusions of csets $\mathcal{F}(U) \hookrightarrow \mathcal{F}(V)$), meaning they preserve these specific pullback diagrams. As such, the left Kan extension $\mathcal{F}_!$ of $\mathcal{F}$ preserves monomorphisms. Thus, any protocol complex functor $\mathcal{F}_!$ preserves both colimits and monomorphisms. ■

As the protocol complex functor $\mathcal{F}_!$ preserves colimits and monomorphisms by Lemma 6.20, the construction of Proposition 6.19 gives us a free $\mathcal{F}_!$-algebra generated by an arbitrary cset $X$.

▶ **Corollary 6.21.** Every protocol complex functor $\mathcal{F}_! : [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ is a varietor, i.e., it admits a free algebra $(\mathcal{I}^\infty, \phi_\mathcal{I})$ generated by any cset $\mathcal{I}$. ◀

*Proof.* By Lemma 6.20, any protocol complex functor preserves monomorphisms and colimits. By Proposition 6.19, an endofunctor satisfying these conditions is a varietor. ■

By establishing protocol complex functors as varietors, Corollary 6.21 shows that our framework can capture the free iteration of communication protocols for any cset of initial global states. We have now a canonical way to "unfold" any protocol from any input configuration. The free algebra $(\mathcal{I}^\infty, \phi_\mathcal{I})$ accumulates all finite rounds of $\mathcal{F}_!$ by embedding successive rounds via the algebra map $\phi_\mathcal{I} : \mathcal{F}_!(\mathcal{I}^\infty) \to \mathcal{I}^\infty$. The structure $\mathcal{I}^\infty = \coprod_{n \geqslant 0} \mathcal{F}_!{}^n(\mathcal{I})$ collects all configurations

reachable after exactly $n$ rounds, with the algebra map embedding each round's output into an execution space.

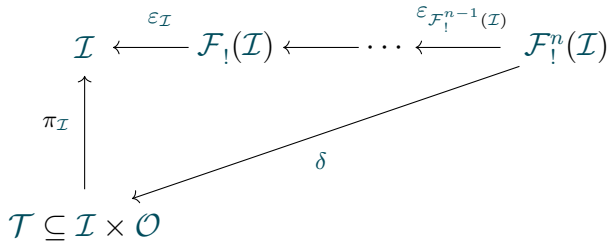
This algebraic framework can in fact capture the framework for task solvability of [HKR14], briefly reviewed in Section 2.3, where we use csets rather than simplicial complexes.

▶ **Example 6.22 (Distributed task specification and solvability).** Recall from Definitions 2.24 to 2.26 that a distributed task is a triple $\mathbf{T} = (\mathcal{I}, \mathcal{O}, \Delta)$, where $\mathcal{I}$ and $\mathcal{O}$ are chromatic simplicial complexes and $\Delta \colon \mathcal{I} \to 2^{\mathcal{O}}$ is a task specification chromatic carrier map. A simplicial protocol $\Psi$ is a chromatic carrier map, whose $n$-iterated application $\Psi^n$ produces a protocol complex containing all reachable states after $n$ rounds. The protocol $\Psi$ solves $\mathbf{T}$ if there exists a decision map $\delta \colon \Psi^n(\mathcal{I}) \to \mathcal{O}$ such that $\delta(\Psi(X)) \subseteq \Delta(X)$ for all $X \in \mathcal{I}$.

We can lift this framework from chromatic simplicial complexes to csets. A task specification becomes a sub-cset $\mathcal{T} \subseteq \mathcal{I} \times \mathcal{O}$ of the product of input and output csets, defined as the functor $\mathcal{T} \colon \Gamma^{\mathrm{op}} \to \mathbf{Set}$ given by

$$\mathcal{T}(U) = \{(\sigma, \tau) \in \mathcal{I}(U) \times \mathcal{O}(U) \mid (\sigma, \tau) \text{ satisfies the task specification}\}.$$

A communication protocol modeled as a protocol complex functor $\mathcal{F}_!$ generates a free $\mathcal{F}_!$-algebra $(\mathcal{I}^\infty, \phi_{\mathcal{I}})$, with $\mathcal{I}^\infty \cong \coprod_{k \geq 0} \mathcal{F}_!{}^k(\mathcal{I})$ (Proposition 6.19). $\mathcal{F}_!{}^n(\mathcal{I})$ contains the states reachable after exactly $n$ rounds. *Task solvability* at round $n$ requires then a morphism of csets $\delta \colon \mathcal{F}_!{}^n(\mathcal{I}) \to \mathcal{T}$, making the following diagram commute.




The composition of $\varepsilon$ maps from Lemma 6.15 along the horizontal chain tracks which input $\sigma \in \mathcal{I}$ generated each reachable global state in $\mathcal{F}_!{}^n(\mathcal{I})$. We obtain again the task solvability condition asking whether there is, for some round $n$, a decision map $\delta \colon \mathcal{F}_!{}^n(\mathcal{I}) \to \mathcal{T}$ such that $\pi_{\mathcal{T}} \circ \delta$ equals this composition ◀

We proceed next by making use of this algebraic structure as a model for a temporal-epistemic logic, allowing us to reason about knowledge of agents throughout the iterations of a distributed communication protocol.

## 6.3 TEMPORAL-EPISTEMIC SEMANTICS ON PROTOCOL ALGEBRAS

We return now to the central question of Chapter 4: reasoning about the knowledge in distributed multi-robot[3] systems over time. This question was tackled then by enriching system frames, Kripke-like structures with temporal and epistemic relations, with task-relevant space propositions. While those system frames were themselves generated from system runs arising from the I/O automata of Chapter 3, this generation remained external to the logical framework. We approach this question now using the free algebra construction of Section 6.2. As such, the generative structures of protocol functors becomes first-class objects within our framework, and properties of specific protocols, such as history preservation in wait-free systems, follow directly from the structures used, instead of requiring separate verification. While the validity of a system frame for a model required us to check their satisfaction of desired axioms, the free algebras generated by a protocol complex functor automatically inherit most of the desired properties. Furthermore, such categorical framework provides a uniform language for comparing and extending models, as observed in Remark 6.13, and will be made evident in Section 6.4 when we lift the framework to contain explicit decision values, preserving the algebraic structures.

In order to reason about these algebraically-generated structures, we introduce iteration propositions.

▶ **Definition 6.23 (Iteration propositions).** Let $\mathcal{A}$ be a finite set of agents and $U \subseteq \mathcal{A}$. The set of possible *iteration propositions* is given by $p \in At = \mathrm{S}_{\text{task}} \cup \{\mathrm{alive}(a), \mathrm{dead}(a) \mid a \in \mathcal{A}\}$, where $\mathrm{S}_{\text{task}}$ are task-specific propositions, and

$$\mathrm{dead}(a) \coloneqq K_a \mathrm{false} \qquad \mathrm{alive}(a) \coloneqq \neg \mathrm{dead}(a)$$
$$\mathrm{dead}(U) \coloneqq \bigwedge_{a \in U} \mathrm{dead}(a) \qquad \mathrm{alive}(U) \coloneqq \neg \mathrm{dead}(U)$$

[3] We preserve the agent language of the previous sections, but note again that robotic systems remain contained as specialized cases.

◀

The iteration propositions, as the previous space propositions, capture atomic facts that are evaluated throughout the rounds of execution. They can either be task-specific, such as "robot r is in position $x$" in a robotic context, or contain agent participation information in fault-tolerant settings, as done in [GLR22].

We adapt the temporal-epistemic logic of Chapter 4 as follows.

▶ **Definition 6.24 (Language of iterated knowledge).** Let $At$ be a set of iteration propositions and $\mathcal{A}$ a finite set of agents. The *language* $\mathcal{L}_{\text{iter}}$ is defined by the following grammar:

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid C_B\varphi \mid D_B\varphi \mid X\varphi \mid \Diamond\varphi \mid \Box\varphi$$

where $p \in At$, $a \in \mathcal{A}$, and $B \subseteq \mathcal{A}$. ◀

The temporal operators $\Diamond$ ("*eventually*") and $\Box$ ("*always*") quantify over rounds of the execution of a protocol, while $X$ (next-step) consider exactly one round forward. The epistemic operators $K_a$ ("agent $a$ *knows*"), $C_B$ ("*common knowledge* among agents $B$"), and $D_B$ ("*distributed knowledge* among agents $B$"), capture what agents can deduce from their local states within the structure of a cset.

We are interested in interpreting formulas $\varphi \in \mathcal{L}_{\text{iter}}$ in the algebraic execution structure within a free $\mathcal{F}_!$-algebra $(\mathcal{I}^\infty, \phi_{\mathcal{I}})$. That is, we want to model the truth value of formulas within the cset $\mathcal{I}^\infty$, carrier of a free algebra, for an arbitrary set of input global states $\mathcal{I}$. It contains in a single object all of the possible executions of the protocol complex functor $\mathcal{F}_!$. For this purpose, we use a semi-simplicial model, a variation of the "maximal epistemic covering model" introduced in [Gou+23].

▶ **Definition 6.25 (Semi-simplicial model).** A *semi-simplicial model* is a pair $M[X] = (X, \ell)$, where $X$ is a cset and $\ell : X_0 \to 2^{At}$ is a *valuation function* that associates to each vertex $v \in X_0$ a set of atomic propositions $\ell(v) \subseteq At$ that hold in $v$. The valuation function is defined for simplices $w \in X_n$, for $n > 0$, as follows. For an $n$-simplex $w$, the face maps $\partial_i \colon X_n \to X_{n-1}$ in the cset extract the lower-dimensional faces, and composing them repeatedly extracts the vertices.

$$\ell(w) = \bigcup_{0 \leqslant i_1 < i_2 < \ldots < i_n \leqslant n} \{\ell(\partial_{i_1} \circ \partial_{i_2} \circ \ldots \circ \partial_{i_n}(w))\} \tag{6.3}$$

That is, $\ell(w)$ collects all propositions holding at any vertex $W$, accessed via all possible chain maps.

For two semi-simplicial models $M[X] = (X, \ell)$ and $M[Y] = (Y, \ell')$, a *morphism of semi-simplicial models* $f : X \to Y$ is a morphism of csets that does not create new valuations, i.e., $\ell'(f(s)) \subseteq \ell(s)$. The *category of semi-simplicial models* with atomic propositions $At$ is denoted $\mathsf{Cset}_{At}$. ◀

Note that the algebraic framework of Section 6.2 extends to the semi-simplicial models. A protocol functor $\mathcal{F} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ acts on the cset component of the semi-simplicial model, while tracking the propagation of valuations.

▶ **Remark 6.26 (Protocol functors over semi-simplicial models).** For a protocol functor $\mathcal{F} \colon \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ and $\mathcal{F}_! \colon [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ its Yoneda extension, we can lift $\mathcal{F}_!$ pointwise to semi-simplicial models, providing an endofunctor $\mathcal{F}_{!At} \colon \mathsf{Cset}_{At} \to \mathsf{Cset}_{At}$. Consider a model $M[X] = (X, \ell)$. The protocol complex functor $\mathcal{F}_{!At}$ acts on simplices $\sigma \in X[U]$, for some set of agents $U \in \Gamma$ as

$$\mathcal{F}_{!At}(M[\sigma]) \coloneqq \big(\mathcal{F}(\sigma), \ell'\big)$$

Where for each vertex $v \in \mathcal{F}(\sigma)(\{a\})$, with color corresponding to the agent $a$ there, we have

$$\ell'(v) \coloneqq \ell\big(\partial_{\{a\},U}(\sigma)\big)$$

In other words, a vertex in the image of $\mathcal{F}_{At}(\sigma)$ inherits the propositions from the corresponding $a$-vertex in $\sigma$, which is obtained through the face map $\partial_{\{a\},U} \colon X(U) \to X(\{a\})$. This valuation is extended to all simplices as per Equation (6.3). ◀

When agent $a$ participates in a round of the protocol, its resulting local state inherits the propositions that were true at its previous local state. This preservation made explicit in Remark 6.26 ensures that the basic facts about the system propagate as expected throughout the iterations.

For the remaining of this section, we will work on objects $M[X]$ of the category of semi-simplicial models, which envelop the cset $X$ with the model structure required to interpret formulas from Definition 6.24. The precise interpretation is given by the following semantics, which adapts the one used in system frames (Definition 4.9) for the algebraic structure of semi-simplicial models. In a semi-simplicial model, all simplices, of any dimension, are worlds in the sense of the more classical Kripke semantics, reviewed in Section 2.3.2.

▶ **Definition 6.27 (Semantics of semi-simplicial models).** Let $At$ be a set of iteration propositions for agents $\mathcal{A}$. Let $\mathcal{F}_! \colon \mathsf{Cset}_{At} \to \mathsf{Cset}_{At}$ be a protocol complex functor, and let $(\mathcal{I}^\infty, \phi_\mathcal{I}) \cong \coprod_{n\geq 0} \mathcal{F}_!{}^n(\mathcal{I})$ its free $\mathcal{F}_!$-algebra. For a subset of agents $B \subseteq \mathcal{A}$, let $\mathrm{lnk}_B(x,y)$ be the *link predicate* testing if simplices $x, y \in \mathcal{I}^\infty$ share a common $B$-simplex $\sigma \in \mathcal{I}^\infty(B)$ as a face.

$$\mathrm{lnk}_B(x,y) \coloneqq \exists \sigma \in \mathcal{I}^\infty(B) : x \xrightarrow{\partial_{x,\sigma}} \sigma \xleftarrow{\partial_{y,\sigma}} y$$

The *semantics* of the semi-simplicial model $M[\mathcal{I}^\infty] = (\mathcal{I}, \ell)$ is defined for each simplex $w \in \mathcal{I}^\infty$ as follows.

$$
\begin{array}{lllll}
(At) & M[\mathcal{I}^\infty], w \models p & \text{iff} & p \in \ell(w) \\
(\neg) & M[\mathcal{I}^\infty], w \models \neg\varphi & \text{iff} & M[\mathcal{I}^\infty], w \not\models \varphi \\
(\wedge) & M[\mathcal{I}^\infty], w \models (\varphi \wedge \psi) & \text{iff} & M[\mathcal{I}^\infty], w \models \varphi \text{ and } M[\mathcal{I}^\infty], w \models \psi \\
(K) & M[\mathcal{I}^\infty], w \models K_a\varphi & \text{iff} & \forall w' \in \mathcal{I}^\infty \text{ s.t. } \mathrm{lnk}_{\{a\}}(w,w'),\ M[\mathcal{I}^\infty], w' \models \varphi \\
(D) & M[\mathcal{I}^\infty], w \models D_B\varphi & \text{iff} & \forall w' \in \mathcal{I}^\infty \text{ s.t. } \mathrm{lnk}_B(w,w'), M[\mathcal{I}^\infty], w' \models \varphi \\
(C) & M[\mathcal{I}^\infty], w \models C_B\varphi & \text{iff} & \forall w' \in \mathcal{I}^\infty \text{ s.t. } \exists (w_0,\dots,w_n), w_0 = w, w_n = w' \\
 & & & \qquad \wedge_{i=1}^n \left(\mathrm{lnk}_B(w_i, w_{i-1})\right),\ M[\mathcal{I}^\infty], w' \models \varphi \\
(X) & M[\mathcal{I}^\infty], w \models X\varphi & \text{iff} & \forall w' \in \mathcal{I}^\infty \text{ s.t. } w \in \mathcal{F}_!{}^n(\mathcal{I}),\ w' \in \mathcal{F}_!{}^{n+1}(\mathcal{I}), \\
 & & & \qquad \phi_\mathcal{I}(w') = w, M[\mathcal{I}^\infty], w' \models \varphi \\
(\Diamond) & M[\mathcal{I}^\infty], w \models \Diamond\varphi & \text{iff} & \exists i \geqslant 0 \text{ s.t. } M[\mathcal{I}^\infty], w \models X^i\varphi \\
(\Box) & M[\mathcal{I}^\infty], w \models \Box\varphi & \text{iff} & \forall i \geqslant 0 \text{ s.t. } M[\mathcal{I}^\infty], w \models X^i\varphi
\end{array}
$$

◀

The semantics connects the categorical and logical frameworks through the free algebra structure. For a simplex $w \in \mathcal{I}^\infty$, its membership in component $\mathcal{F}_!{}^n(\mathcal{I})$ captures that it is present at round $n$ of the iteration of the protocol complex functor $\mathcal{F}_!$. The rules epistemic and temporal operators make use of the simplicial and algebraic structures, respectively, to derive their meaning:

- $(K)$, $(D)$ and $(C)$ use the same interpretation as in Goubault et al. [Gou+23]. These operators are modeled using the simplicial structure of $\mathcal{I}^\infty$, where connected simplices (worlds) are said to be indistinguishable by the vertices

(agents) or simplices (groups of agents) that they have in common. The link predicate $\mathrm{lnk}_B(w, w')$ captures when a group of agents $B$ cannot distinguish between global states $w$ and $w'$ as precisely when those share a common $B$-simplex[4]. Knowledge by an agent (resp. distributed knowledge by a group of agents) of a formula $\varphi$ translates then to requiring that $\varphi$ holds in all worlds linked through an $\{a\}$-simplex (resp. $B$-simplex). Common knowledge extends this requirement to the transitive closure of the link relation, capturing knowledge through a chain of indistinguishabilities.

- $(X)$ defines the *next-step* temporal modality as that which is true at some world $w \in {\mathcal{F}_!}^n(\mathcal{I})$ of round $n$ and remains true in all successor worlds $w' \in {\mathcal{F}_!}^{n+1}(\mathcal{I})$ during round $n+1$. The successor worlds are those satisfying $w \in {\phi_{\mathcal{I}}}^{-1}(w')$ where the algebra map embeds the possible states from round $n+1$ into the cumulative execution state space. The rules $(\Diamond)$ and $(\Box)$ come then from the immediate usage of the stepping operator.

An important feature of our framework is that a semi-simplicial model over a cset $\mathcal{I}^\infty$ unfolds the complete execution tree. By embedding new rounds into a cumulative execution space, the algebra map $\phi_{\mathcal{I}} \colon \mathcal{F}_!(\mathcal{I}^\infty) \to \mathcal{I}^\infty$ ensures that the information in round $n$ is not lost in round $n+1$. Unlike models based on Kripke frames, that need to track events and values explicitly, the free algebra construction ensures that every possible execution of a protocol is represented with all intermediate states preserved. As such, the temporal operators in Definition 6.27 are quantified over complete histories, instead of arbitrary time points.

As a consequence of this preservation property, our interpretation of the temporal-epistemic logic from Definition 6.24 only allows agents to learn more information throughout an execution. That is, the indistinguishability of global states, assessed by the link predicate in the simplicial structure of $\mathcal{I}^\infty$, may be refined as agents communicate. Take for instance the synchronous protocols that generate holes in the protocol complex, as seen in Example 6.14, where the agents learn to distinguish states based on messages not being received. This leads to the

[4]This geometric condition corresponds to the indistinguishability relation that arises when agents in $B$ have compatible local states, generalizing the classical approach from Section 2.3.2.

following knowledge gain result, similar to previous work by Goubault, Ledent and Rajsbaum [GLR21], which now formalizes that morphisms of semi-simplicial models cannot create new knowledge about positive epistemic formulas.

▶ **Definition 6.28 (Positive epistemic formula).** Let $At$ be a finite set of propositions and $\mathcal{A}$ a set of agents. A *positive epistemic formula* $\varphi$ is build according to the following grammar, for $a \in \mathcal{A}$ and $B \subseteq \mathcal{A}$.

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid K_a\varphi \mid D_B\varphi \mid C_B\varphi$$

The set of all positive epistemic formulas is denoted $\mathcal{L}_{\mathrm{K}}^{+}$ ◀

Positive epistemic formulas do not contain negations, except possibly in front of atomic propositions, and use only epistemic operators.

▶ **Theorem 6.29 (Knowledge gain).** Let $M[X] = (X, \ell)$ and $M[Y] = (Y, \ell')$ be two semi-simplicial models and $f : M[X] \to M[Y]$ a morphism between them such that $f(w) = w'$, where $w \in X$ and $w' \in Y$. Let $\varphi \in \mathcal{L}_{\mathrm{K}}^{+}$ be a positive epistemic formula. Then, $M[Y], w' \models \varphi$ implies $M[X], w \models \varphi$.

*Proof.* The proof follows by structural induction on the positive epistemic formula $\varphi \in \mathcal{L}_{\mathrm{K}}^{+}$. We assume that the claim holds for all subformulas $\psi$ of $\varphi$ and prove it for $\varphi$.

**Base case.** For $\varphi = p$, an atomic proposition, suppose $M[Y], w' \models p$, for $p \in At$. By the semantics of semi-simplicial models, we know that $p \in \ell'(w')$. As $f$ is a morphism of semi-simplicial models and $f(w) = w'$, we have that $\ell'(w') = \ell'(f(w)) \subseteq \ell(w)$. This means that $p \in \ell(w)$, thus $M[X], w \models p$.

**Negation.** For $\varphi = \neg p$, the proof follows is analogous to that for $\varphi = p$.

**Conjunction.** For $\varphi = \varphi_1 \wedge \varphi_2$, suppose $M[Y], w' \models \varphi_1 \wedge \varphi_2$. This means that $M[Y], w' \models \varphi_1$ and $M[Y], w' \models \varphi_2$. By the induction hypothesis, $M[X], w \models \varphi_1$ and $M[X], w \models \varphi_2$, thus $M[X], w \models \varphi_1 \wedge \varphi_2$.

**Knowledge.** For $\varphi = K_a\psi$, suppose $M[Y], w' \models K_a\psi$. This means, by the semantics, that for all $w'' \in Y$ with $\mathrm{lnk}_{\{a\}}(w', w'')$, we have $M[Y], w'' \models \psi$. We want to show that $M[X], w \models K_a\psi$, and for that we consider any

$s \in X$ with $\mathrm{lnk}_{\{a\}}(w, s)$. By the definition of the link predicate, this means that there exists $\sigma \in X(\{a\})$ and face maps $\partial_{w,\sigma} \colon w \to \sigma$ and $\partial_{s,\sigma} \colon s \to \sigma$. As $f$ is a morphism of semi-simplicial models, it is a morphism of csets, which commutes with face maps. Therefore, $f(\partial_{w,\sigma}) \colon f(w) \to f(\sigma)$ and $f(\partial_{s,\sigma}) \colon f(s) \to f(\sigma)$ are face maps in $Y$, with $f(\sigma) \in Y(\{a\})$. That is, $\mathrm{lnk}_{\{a\}}(w', f(s))$. By the assumption, we have that $M[Y], f(s) \models \psi$. By the induction hypothesis, we have that $M[X], s \models \psi$. As the choice of $s$ was arbitrary, we conclude that $M[X], w \models K_a\psi$.

**Distributed knowledge.** For $\varphi = D_B\psi$, the proof follows the same structure as for $\varphi = K_a\psi$, where $\{a\}$ is replaced by $B$.

**Common knowledge.** For $\varphi = C_B\psi$, suppose $M[Y], w' \models C_B\psi$. By the semantics, for all $w'' \in Y$ reachable from $w'$ via chain of worlds indistinguishable by some agent in $B$, $w'_0, \ldots, w'_n$, with $w'_0 = w'$, $w'_n = w''$ and $\mathrm{lnk}_B(w'_i, w'_{i-1})$ for all $i \in \{1, \ldots, n\}$, we have $M[Y], w'' \models \psi$. As we want to show that $M[X], w \models C_B\psi$, we take any $s \in X$ that is reachable from $w$ by a $B$-chain $(s_0, \ldots, s_m)$ with $s_0 = w$, $s_m = s$ and $\mathrm{lnk}_B(s_i, s_{i-1})$ for all $i \in \{1, \ldots, m\}$. By the same argument for the knowledge, $f$ preserves link predicates, that is, $\mathrm{lnk}_B(s_i, s_{i-1})$ in $X$ implies $\mathrm{lnk}_B(f(s_i), f(s_{i-1}))$ in $Y$. As such, $(f(s_0), \ldots, f(s_m))$ forms a $B$-chain in $Y$ from $w' = f(w) = f(s_0)$ to $f(s) = f(s_m)$. By assumption, $M[Y], f(s) \models \psi$. By the induction hypothesis, $M[X], s \models \psi$. As the choice of $s$ was arbitrary, we conclude that $M[X], w \models C_B\psi$.

■

The algebra map $\phi_{\mathcal{I}} : \mathcal{F}_!(\mathcal{I}^\infty) \to \mathcal{I}^\infty$ is itself a morphism of semi-simplicial models by the construction of Remark 6.26. As such, Theorem 6.29 applies directly to protocol iterations, i.e., if a positive epistemic formula holds for some simplex $w \in \mathcal{F}_!{}^n(\mathcal{I})$ at round $n$, it continues to hold at all successor simplices in round $n + 1$. We formalize next this temporal persistence.

▶ **Lemma 6.30 (Persistence of positive formulas).** Let $\mathcal{F}_! \colon \mathsf{Cset}_{At} \to \mathsf{Cset}_{At}$ be a protocol complex functor and $(\mathcal{I}^\infty, \phi_{\mathcal{I}})$ be a free $\mathcal{F}_!$-algebra for any cset $\mathcal{I}$. For any positive

epistemic formula $\varphi \in \mathcal{L}_{\mathrm{K}}^{+}$ and simplex $w \in {\mathcal{F}_!}^n(\mathcal{I}^\infty)$, the following are satisfied.

$$M[\mathcal{I}^\infty], w \models \varphi \;\Rightarrow\; M[\mathcal{I}^\infty], w \models X\varphi \tag{6.4}$$

$$M[\mathcal{I}^\infty], w \models \varphi \;\Rightarrow\; M[\mathcal{I}^\infty], w \models \Box\varphi \tag{6.5}$$

$$M[\mathcal{I}^\infty], w \models \varphi \;\Rightarrow\; M[\mathcal{I}^\infty], w \models \Diamond\varphi \tag{6.6}$$

◀

*Proof.* We derive each property individually.

- For $\varphi \in \mathcal{L}_{\mathrm{K}}^{+}$ a positive epistemic formula, as $\phi_{\mathcal{I}} \colon \mathcal{F}_!(\mathcal{I}^\infty) \to \mathcal{I}^\infty$ is a morphism of semi-simplicial models, we get from Theorem 6.29 that $M[\mathcal{I}^\infty], w \models \varphi$ implies $M[\mathcal{I}^\infty], w' \models \varphi$ when $\phi_{\mathcal{I}}(w') = w$. Similarly, ${\mathcal{F}_!}^n(\phi_{\mathcal{I}}) \colon {\mathcal{F}_!}^{n+1}(\mathcal{I}) \to {\mathcal{F}_!}^n(\mathcal{I})$ is a morphism of semi-simplicial models, and as such it holds that $M[{\mathcal{F}_!}^n(\mathcal{I})], x \models \varphi$ implies $M[{\mathcal{F}_!}^{n+1}(\mathcal{I})], y \models \varphi$ when ${\mathcal{F}_!}^n(\phi_{\mathcal{I}})(y) = x$, i.e., $M[\mathcal{I}^\infty], x \models X\varphi$. By the semantics of $X\varphi$, we conclude Equation (6.4).
- Equation (6.5) can be obtained by induction on the number of applications of the algebra map $\phi_{\mathcal{I}}$.
- Equation (6.6) is obtained by noting that $\Box\varphi$ implies $\Diamond\varphi$.

■

Lemma 6.30 can be understood as the fact that a distributed system with its communication protocol modeled using a protocol complex functor never forgets true statements about the initial values of the agents. This preservation property contrasts our algebraic model from the system frames of Chapter 4, where history preservation had to be imposed as an external axiom (Assumption 4.12). Note that the persistence of positive epistemic formulas emerges from the categorical structure itself, rather than an axiom imposed on the logic. From this result, combined with the epistemic axioms of $KB4$ for distributed knowledge and standard axioms for linear temporal logic, yields the following sound axiom system.

▶ **Lemma 6.31 (Sound axioms for semi-simplicial models).** Let $\mathcal{A}$ be a set of agents, with $a \in \mathcal{A}$ and $B \subseteq \mathcal{A}$. Let $\varphi, \psi \in \mathcal{L}_{\mathrm{iter}}$ and $\varphi^+ \in \mathcal{L}_{\mathrm{K}}^{+}$. The following axioms are sound

in all free algebras over csets:

| | |
|---|---|
| (**K**) | $D_B(\varphi \Rightarrow \psi) \Rightarrow (D_B\varphi \Rightarrow D_B\psi)$ |
| (**4**) | $D_B\varphi \Rightarrow D_B D_B\varphi$ |
| (**B**) | $\varphi \Rightarrow D_B\neg D_B\neg\varphi$ |
| (**Mono**) | $B \subseteq B' \Rightarrow (D_B\varphi \Rightarrow D_{B'}\varphi)$ |
| (**Union**) | $\mathrm{alive}(B) \wedge \mathrm{alive}(B') \Rightarrow \mathrm{alive}(B \cup B')$ |
| (**NE**) | $\bigvee_{a\in\mathcal{A}} \mathrm{alive}(a)$ |
| (**P**) | $\mathrm{alive}(B) \wedge \mathrm{dead}(B^c) \wedge \varphi \Rightarrow D_B(\mathrm{dead}(B^c) \Rightarrow \varphi)$ |
| (**Max**) | $B \neq \varnothing \Rightarrow (\mathrm{alive}(B) \Rightarrow \neg D_B\neg\mathrm{dead}(B^c))$ |
| (**GFP**) | $C_{\mathcal{A}}(\varphi \Rightarrow \bigwedge_{a\in\mathcal{A}} K_a\varphi) \Rightarrow (\varphi \Rightarrow C_{\mathcal{A}}\varphi)$ |
| (**KG**) | $\varphi^+ \Rightarrow X\varphi^+$ |
| (**KG**$\Box$) | $\varphi^+ \Rightarrow \Box\varphi^+$ |
| (**KG**$\Diamond$) | $\varphi^+ \Rightarrow \Diamond\varphi^+$ |
| (**AX**) | $\Box\varphi \Rightarrow X\Box\varphi$ |
| (**XX**) | $\Box\varphi \Rightarrow \varphi$ |
| (**AI**) | $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$ |
| (**E**) | $\Diamond\varphi \Leftrightarrow \neg\Box\neg\varphi$ |

◀

*Proof.* Let $\mathcal{F}_!\colon \mathsf{Cset}_{At} \to \mathsf{Cset}_{At}$ be a protocol complex functor and $(\mathcal{I}^\infty, \phi_{\mathcal{I}})$ be its free $\mathcal{F}_!$-algebra. We verify the soundness for the axioms for an arbitrary $M[\mathcal{I}^\infty]$.

**(K), (4), (B), (Mono)** are standard axioms known as $KB4 + Mono + Union$ in [Gou+23] and are known to hold for cset.

**(NE), (P), (Max)** follow from [Gou+23, Theorem 3], as they are known for semi-simplicial models.

**(GFP)** is a standard greatest fixed point characterization of common knowledge, as in [Fag+95].

**(KG), (KG□), (KG◊)** follow immediately from Lemma 6.30.

**(AX), (XX), (AI), (E)** follow from the semantics of temporal operators. Axiom **(AX)** holds since $\Box\varphi$ means $iX[i]\varphi$, for all $i \geq 0$, and so $X(X^j\varphi)$ for all $j \geqslant 0$. Similarly, axiom **(XX)** considers $i = 0$. Axiom **(AI)** applies modus ponens at each step. Axiom **(E)** is the known duality of the eventually and always modalities.

■

Through the temporal-epistemic logic defined here, we can formalize the conditions for solvability of specific distributed tasks. This complements the solvability condition of Example 6.22, as our algebraic framework allows us to express both geometric requirements with csets and logical ones as we see now.

▶ **Example 6.32 (Binary consensus specification).** Consider the binary consensus task (Example 2.17), where $n$ agents $\mathcal{A} = \{a_1, \ldots, a_n\}$, each with input values $v_i \in \{0, 1\}$, need to exchange messages in order to eventually decide on an output value $d_i \in \{0, 1\}$ such that they respect TERMINATION, VALIDITY and CONSENSUS. For each agent $a_i \in \mathcal{A}$, let $p_i^0, p_i^1 \in At$ be atomic propositions representing that agent $a_i$'s input is $0$ or $1$, respectively. The binary consensus task can then be specified as a formula in $\mathcal{L}_{\text{iter}}$:

$$\Diamond\left(C_{\mathcal{A}}\left(\bigvee_{i\in\mathcal{A}} p_i^0\right) \vee C_{\mathcal{A}}\left(\bigvee_{i\in\mathcal{A}} p_i^1\right)\right)$$

The eventually modality above encodes TERMINATION, as it ensures the agents reach some decision in a finite amount of rounds. The disjunction of common knowledge formulas states that all agents reach agreement on which input value to coordinate on. When combined with a protocol that uses common knowledge as a decision trigger, it ensures CONSENSUS and VALIDITY. ◀

## 6.4 INCORPORATING LOCAL DECISION FUNCTIONS

Note that the csets we have used so far can only track structural information of the protocol execution, such as the participation of agents, their views and the compatibility of local and global states. As a consequence, the logic modeled

on them could only express temporal and epistemic properties about the agents' input values, such as their knowledge over time. We extend now the protocol functor defined in Section 6.1 by incorporating explicit decision values into the state of the agents. This allows us to model concrete protocols where agents update numerical data, such as in averaging algorithms for robot coordination, and to relate the algebraic framework back to the spatial tasks model of Chapter 4.

We enrich the category of subsets of agents $\Gamma$ (Definition 2.34) to the category $\Psi$ by indexing with a set of decision values $V$. Objects become pairs $(B, d)$ of agent sets and decision functions.

▶ **Definition 6.33 (Category $\Psi$).** Let $V$ be a set of *decision values*. The category $\Psi$ has:

- Objects: pairs $(B, d)$, where $B \in \Gamma$ and $d \colon B \to V \cup \{\bot\}$ is a *decision function*;
- Morphisms: $f \colon (B, d) \to (C, e)$ are inclusions $\delta_{B,C} \colon B \subseteq C$ in $\Gamma$ such that for all $b \in B$, $e(b) = d(b)$.

Composition and identity are given by the transitivity and reflexivity of inclusion maps. ◀

As with the category $\Gamma$, we use presheaves on $\Psi$ to model the distributed state space of a distributed system, now including decision values in the local states.

▶ **Definition 6.34 (Chromatic semi-simplicial sets with decision values).** A *chromatic semi-simplicial set with decision values*, abbreviated as dcset, is a functor $X \colon \Psi^{\mathrm{op}} \to \mathbf{Set}$. The category of dcsets is the presheaf category $[\Psi^{\mathrm{op}}; \mathbf{Set}]$, whose morphisms are natural transformations. ◀

Each dcset assigns to every pair $(B, d) \in \Psi$ of agents with decision functions a set of simplices, where the face maps preserve both the simplicial structure and the decision values.

▶ **Remark 6.35.** By the fundamental theorem of topoi [MM94], the category of dcsets $[\Psi^{\mathrm{op}}; \mathbf{Set}]$ is equivalent to the slice category $[\Gamma^{\mathrm{op}}; \mathbf{Set}]/D$, where $D$ is the cset of all possible decision value assignments to agents.

Concretely, for a set of agents $\mathcal{A}$ and a set of decision values $V$, the cset $D$ has $U$-simplices $s = ((a_0, d_0), \ldots, (a_n, d_n))$, for $a_i \in U \subseteq \mathcal{A}$ and $d_i \in V$. The equivalence $[\Psi^{\mathrm{op}}; \mathbf{Set}] \cong [\Gamma^{\mathrm{op}}; \mathbf{Set}]/D$ means that every dcset can be viewed as a cset equipped with a morphism $X \to D$ that assigns decision values to agents in each simplex, such that face maps are respected. ◀

### 6.4.1 PROTOCOL FUNCTORS

The protocol functors, used to model communication protocols in Section 6.1, lift to functors valued in dcsets that preserve both the simplicial structure and the decision values.

▶ **Definition 6.36 (Protocol functor with decisions).** A *protocol functor with decisions* is a functor $\mathcal{F}\colon \Psi \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ that preserves monomorphisms. ◀

Similarly to Definition 6.6, the protocol functor with decisions assigns to each pair $(B, d) \in \Psi$ a dcset $\mathcal{F}((B, d))$ with all possible configurations reachable after one round of communication when agents in $B$ start with decision functions encoded in $d$. The property of preservation of monomorphisms encodes wait-freedom (Assumption 6.5). That is, for $B' \subseteq B$ with $d|_{B'} = d'$, then $\mathcal{F}(\delta_{B',B})\colon \mathcal{F}(B', d) \to \mathcal{F}(B, d)$ is a monomorphism, capturing that configurations valid for fewer participating agents remain valid when additional agents are present.

As provided by Proposition 6.8, the protocol functor with decisions $\mathcal{F}\colon \Psi \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ also accepts a Yoneda extension $\mathcal{F}_!\colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$. This construction is called a *protocol complex functor with decisions*, extending the functorial definition of the protocol to arbitrary dcsets. The following example shows the first benefit of this extension, as we can now model multi-robot systems using our algebraic framework, where robot positions are encoded as decision values

▶ **Example 6.37 (Protocol functor with decisions: multi-robot systems).** There is a similarity between computational and robotic systems endowed with communication faults and potential lack of synchronicity, as observed in Section 2.2.3. Other problems may be introduced, requiring assumptions such as physical realizability (Assumption 3.35) for moving freely between computational and dynamical models. We abstract those now and consider the LOOK-COMPUTE-MOVE cycle for a *multi-robot system*[5], reviewed in Section 2.2.2. Each robot $\mathbf{r}_i \in \mathcal{R}$ has a position $x_i \in \mathcal{X} \subset \mathbb{R}^p$ and is equipped with a set of visible lights used for communication and memory. At each round, they perform the following sequence of actions.

[5]Note that the LCM cycle was instantiated in Section 3.3 using I/O automata, the briefer version of Chapter 2 is sufficient for this example.

1. LOOK: They acquire the positions and lights of the robots $\mathbf{r}_j$ present within a fixed visibility radius $r$;
2. COMPUTE: They compute their next position within a set of reachable ones $\mathcal{N}_{\mathcal{X}}(x_i)$, as well as their next configuration of lights;
3. MOVE: They update their positions and lights to the computed values;

We now model this system via a protocol complex functor with decisions $\mathcal{F}_{!lcm}$: $[\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ where decision values are positions in space, i.e., $V = \mathcal{X}$. $\mathcal{F}_{!lcm}$ is defined as the Yoneda extension of the functor $\mathcal{F}_{lcm} \colon \Psi \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$. For robots $\mathbf{r}_i \in A \subseteq \mathcal{R}$, an object $(A, d) \in \Psi$, where $d\mathbf{r}_i$ gives the current position of $\mathbf{r}_i$, the dcset $\mathcal{F}_{lcm}((A, d))$ has $(B, e)$-simplices, with $e$ giving the next positions of robots in $B$ after one round of the protocol:

- $\varnothing$ if $B \not\subseteq A$ or $e(\mathbf{r}_i) \notin \mathcal{N}_{\mathcal{X}}(d(\mathbf{r}_i))$ for some $\mathbf{r}_i \in B$;
- Otherwise, when $B = \{\mathbf{r}_0, \ldots, \mathbf{r}_m\} \subseteq A$ and $e(\mathbf{r}_i) \in \mathcal{N}_{\mathcal{X}}(d(\mathbf{r}_i))$ for all $\mathbf{r}_i \in B$:

$$\left\{ (\mathbf{r}_0, W_0, e(\mathbf{r}_0)), \cdots, (\mathbf{r}_m, W_m, e(\mathbf{r}_m)) \;\middle|\; \begin{array}{l} W_i \subseteq A \\ \forall i,\ 0 \leqslant i \leqslant m,\ \mathbf{r}_i \in W_i \\ \forall i, j,\ 0 \leqslant i, j \leqslant m,\ (W_i \subseteq W_j) \text{ or } (W_j \subseteq W_i) \\ \forall i, j,\ 0 \leqslant i, j \leqslant m,\ \mathbf{r}_i \in W_j \Rightarrow W_i \subseteq W_j \\ \forall i, j,\ \|e(\mathbf{r}_i) - e(\mathbf{r}_j)\| \leqslant r \end{array} \right\}$$

The sets $W_i \subseteq B$ are the views obtained by $robot_i$ during the LOOK operation, representing the robots observed within the visibility radius. The first four conditions encode the immediate snapshot structure of Example 6.9. That is, participating robots in a round are a subset of the ones previously participating, and views are both nested and must respect inclusions. The latter condition encodes the visibility constraint, relating the geometric conditions to the combinatorial structure. The only views obtained are of those robots that are visible. ◀

The similarities between Examples 6.9 and 6.37 are evidence of the relationship between the protocol functors from Section 6.1 and their enriched with decision values. More precisely, there is an adjunction between $[\Gamma^{\mathrm{op}}; \mathbf{Set}]$ and $[\Psi^{\mathrm{op}}; \mathbf{Set}]$ defined as follows.

▶ **Definition 6.38 (Adjunction of protocol functors).** Let $\Psi$ be the category of subsets of agents $\mathcal{A}$ with decision values $V$. Consider the functor $c_\pi \colon \Psi \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ where:

- Objects $c_\pi((A, d)) = \Gamma[A]$;
- Morphisms $\delta_{A,B} \colon (A, d) \to (B, e)$ are mapped to morphisms $\Gamma[A] \to \Gamma[B]$ induced by inclusions $A \subseteq B$.

By Proposition 6.8, $c_\pi$ extends to $c_{!\pi} \colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ via a Yoneda extension, which forgets decision values. This extension has a right adjoint:

$$c_{!\pi} \colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \rightleftarrows [\Gamma^{\mathrm{op}}; \mathbf{Set}] \colon d_{!\pi}$$

◀

The functor $c_{!\pi}$ forgets the decision values, while $d_{!\pi}$ enriches csets with constant decision functions. For any protocol functor with decisions $\mathcal{F} \colon \Psi \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$, the composition $c_\pi \circ \mathcal{F}$ provides the underlying protocol functor representing only the communication structure, and discarding the remaining computational information.

▶ **Remark 6.39 (From distributed robotics to distributed computing).** The adjunction in Definition 6.38 opens way for a systematic reduction of problems from distributed robotics to distributed computing. Such connection has been observed in Alcántara et al. [Alc+19], where task solvability of LCM models with unbounded visibility reduces to task solvability of immediate snapshot models. Indeed, the protocol functor with decisions $\mathcal{F}_{lcm}$ in Example 6.37 reduces via $c_\pi$ to the protocol functor $\mathcal{F}_{\mathrm{IS}}$ from Example 6.9. ◀

While the forgetful functor $c_{!\pi} \colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ shows how to reduce protocols with decisions to their underlying communication problems, a well-developed perspective within the distributed computing community, we consider now the dual question of systematically constructing a protocol functor with decisions from its communication structure. This construction is the role of a concrete protocol, which specifies how decision values evolve, augmenting a state-independent protocol functor.

▶ **Definition 6.40 (Concrete protocol).** Let $\mathcal{A}$ be a finite set of agents. A *concrete protocol* $\mathcal{F}_{f!} \colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ is a protocol complex functor with decisions constructed from a protocol complex functor $\mathcal{F}_! \colon [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ and a family of *local decision maps* $f_a \colon 2^{\mathcal{A}} \times V^{\mathcal{A}} \to V$, for each $a \in \mathcal{A}$.

The construction defines a functor $\mathcal{F}_f \colon \Psi \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ by decorating $\mathcal{F}_!$ as follows:

- For $(U, e) \in \Psi$, we associate $\mathcal{F}_!(\Gamma[U])$ decorated to become an element of $[\Psi^{\mathrm{op}}; \mathbf{Set}]$. Each $V$-simplex $\sigma \in \mathcal{F}_!(\Gamma[U])(V)$ becomes a $(V, d)$-simplex where

$$d(a) = f_a(V, e|_V)$$

  for all $a \in V$. This decoration is well-defined by Assumption 6.4, as $V$-simplices of $\mathcal{F}_!(\Gamma[U])$ involve only agents in $V \subseteq U$.

- For a morphism $\delta_{U,U'} \colon (U, e) \to (U', e')$ in $\Psi$, we associate $\mathcal{F}_!(\delta_{U,U'})$, seen as a morphism in $[\Psi^{\mathrm{op}}; \mathbf{Set}]$, which preserves the decision values.

The concrete protocol $\mathcal{F}_{f!}$ is the Yoneda extension of $\mathcal{F}_f$ to all dcsets, given by Proposition 6.8. ◀

The concrete protocol construction, and protocol complex functors with decisions in general, inherit the properties of the protocol complex functor. As the $\mathcal{F}_!$ component is a varietor by Corollary 6.21, and $\mathcal{F}$ decorates $\mathcal{F}_!$ with local decision maps that respect face restrictions, the free algebra of Lemma 6.20 lifts directly.

▶ **Lemma 6.41 (Protocol complex functors with decisions are varietors).** Let $\mathcal{F} \colon \Psi \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ be a protocol functor with decisions and $\mathcal{F}_! \colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ be its Yoneda extension. Then, for any dcset $\mathcal{I}$, $\mathcal{F}_!$ admits a free algebra $(\mathcal{I}^\infty, \phi_{\mathcal{I}})$, i.e., it is a varietor. ◀

*Proof.* By Definition 6.36, $\mathcal{F}$ preserves monomorphisms. As $\mathcal{F}_!$ results from a Yoneda extension (Proposition 6.8), it preserves colimits. By Proposition 6.19, such conditions are sufficient for $\mathcal{F}_!$ to be a varietor, i.e., free algebras exist for any dcset $\mathcal{I}$. ■

Lemma 6.41 ensures that concrete protocols admit free algebras, allowing us to iterate protocols with explicit decision values. We illustrate this construction

with a concrete averaging protocol, where the decision values evolve at each round according to the local decision function to the average of the participating agents.

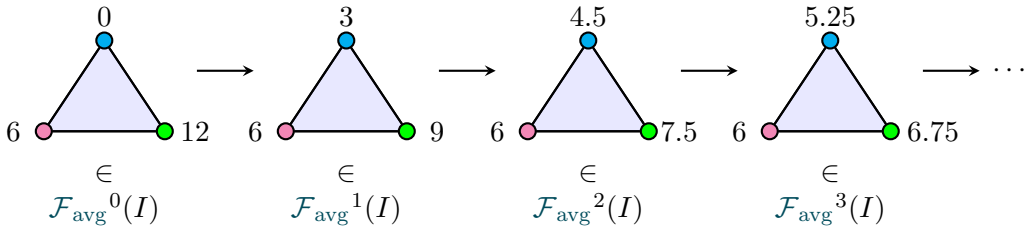
▶ **Example 6.42 (Concrete averaging protocol).** Consider the asynchronous immediate snapshot protocol complex functor $\mathcal{F}_{!\mathbb{S}} \colon [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ of Example 6.9. We construct a concrete protocol where agents iteratively update their decision values towards their local average.

Let $\mathcal{A}$ be a set of agents, and $X$ the dcset with $(V, d)$-simplices, from a set of decision values $V = \mathbb{R}$. Consider the local decision function $\mathrm{avg}_a \colon 2^{\mathcal{A}} \times V^{\mathcal{A}} \to V$, defined for agent $a \in V \subseteq \mathcal{A}$ and parameter $0 < \alpha < 1$ as follows.

$$\mathrm{avg}_a(V, d) = d(a) + \alpha \sum_{b \in V} (d(b) - d(a)) \tag{6.7}$$

This map updates the value of agent $a$ by moving a fraction $\alpha$ in the direction of the average of values in its view $V$. An *averaging protocol functor* $\mathcal{F}_{\mathrm{avg}} \colon [\Psi^{\mathrm{op}}; \mathbf{Set}] \to [\Psi^{\mathrm{op}}; \mathbf{Set}]$ is the concrete protocol given by $\mathcal{F}_{!\mathbb{S}}$ and local decision maps $\{\mathrm{avg}_a\}_{a \in \mathcal{A}}$, as per Definition 6.40.

By Lemma 6.41, $\mathcal{F}_{\mathrm{avg}}$ admits a free algebra $(\mathcal{I}^\infty, \phi_{\mathcal{I}})$, for any initial dcset $\mathcal{I}$. The free algebra $\mathcal{I}^\infty = \coprod_{k \geqslant 0} {\mathcal{F}_{\mathrm{avg}}}^k(\mathcal{I})$, whose component ${\mathcal{F}_{\mathrm{avg}}}^k(\mathcal{I})$ contains the reachable states at exactly $k$ rounds. Consider three agents $\mathcal{A} = \{a, b, c\}$, with initial values $d_0(a) = 0$, $d_0(b) = 6$, and $d_0(c) = 12$, and averaging parameter $\alpha = \frac{1}{2}$. The figure below shows one synchronous line of execution of $\mathcal{I}^\infty$.




Each triangle represents all agents participating in each round $k$, i.e., a $(\{a, b, c\}, d)$-simplex within the corresponding components ${\mathcal{F}_{\mathrm{avg}}}^k(\mathcal{I})$. The sequence shows the values converging towards the initial average of 6 ◀

The averaging protocol functor models a common approach to approximate agreement in distributed computing, as variations of averaging have been shown to guarantee convergence of values, including the cases of asynchronicity with

fairness assumptions and crash or Byzantine [TLV24] faults. In Example 6.42, such guarantees translate to the averaging protocol $\mathcal{F}_{\text{avg}}$, which satisfies both VALIDITY, as the decision values of agents remain in the convex hull of the initial values, and AGREEMENT, as the distance of the decision values decreases geometrically at each round of communication. Just as this approach is also a common one for gathering tasks (see [Kia+19]), we recover its applications to robot protocols, as in Example 6.37, by encoding robot positions as decision values.

### 6.4.2 SEMANTICS

The temporal-epistemic logic of Section 6.3 extends naturally to reason about decision values in free algebras over dcsets. This allows us to express properties not just about the structure of the communication and their initial values, but about the data being computed and updated by the agents.

We augment the language $\mathcal{L}_{\text{iter}}$ from Definition 6.24 by extending its set of atomic iteration propositions $At$ with decision predicates. Those assert facts about the decision values. For instance, the predicate $d_a = 5$ asserts that the decision of agent $a$ is $5$.

▶ **Definition 6.43 (Decision predicates).** Let $V$ be a set of decision values and $\mathcal{A}$ a finite set of agents. A *decision predicate* is a formula $q((d_a)_{a\in B})$, where $B \subseteq \mathcal{A}$ and $q$ is a predicate on the tuple of decision values corresponding to the agents in $B$. ◀

We interpret again formulas within the algebraic execution structure of free algebras, where the semi-simplicial models are now built from dcsets. We work with objects in the *category of semi-simplicial models with decisions* with iteration propositions $At$ (now extended with decision predicates), denoted $\mathsf{dcset}_{At}$. A protocol functor with decisions is lifted to work with semi-simplicial models analogously to Remark 6.26.

Likewise, the semantics in Definition 6.27 is extended to interpret the decision predicates. All rules for the base language, as well as for the temporal and epistemic modalities, remain identical, as they depend only on the simplicial and algebraic structure that is preserved.

▶ **Definition 6.44 (Semantics of decision predicates on free algebras).** Let $At$ be a set of iteration propositions extended with decision predicates. Let $\mathcal{F}_{!At}$ be a protocol

complex functor with decisions and $M[\mathcal{I}^\infty]$ be a free $\mathcal{F}_{!At}$-algebra generated by the initial model $M[\mathcal{I}]$. The semantics from Definition 6.27 is extended with the following rule for any decision predicate $q$.

$$(q) \quad M[\mathcal{I}^\infty], w \models q((d_a)_{a\in B}) \quad \text{iff} \quad w \text{ is a } (B,d)\text{-simplex and the predicate holds for the tuple of decision values } (d(a))_{a\in B}$$

◀

The semantic framework connects the algebraic model to the spatial robot task models of Chapter 4. By choosing decision values $V$ to be spatial data, the spatial propositions from Definition 4.6 become decision predicates.

▶ **Example 6.45 (Robot exploration task).** Consider the exploration task from Section 4.3.1, where a group of robots $\mathcal{R}$ must collectively visit every region of a mission space $\mathcal{X}$. Given a protocol complex functor with decisions $\mathcal{F}_!$ modeling the robots' communication and exploration protocol, we work within its free algebra $(\mathcal{I}^\infty, \phi_\mathcal{I})$ with decision values $V = \tau_\mathcal{X}$, where each decision $d(\mathbf{r})$ is an open subset of $\mathcal{X}$, representing the set of regions explored by a robot $\mathbf{r}$.

The iteration propositions $At$ for exploration are composed of decision predicates defined for each open region $U \in \tau_\mathcal{X}$ by

$$M[\mathcal{I}^\infty], w \models \mathsf{s}(\mathsf{U}) \text{ iff } w \text{ is a } (B,d)\text{-simplex satisfying } U \subseteq \bigcup_{\mathbf{r}\in B} d(\mathbf{r})$$

It becomes evident that the requirement that the entire mission space is explored translates again to $M[\mathcal{I}^\infty] \models \Diamond\mathsf{s}(\mathcal{X})$, while the task specification with termination requires identical properties to those in Definition 4.16, e.g., parallel termination is expressed as

$$M[\mathcal{I}^\infty] \models \Diamond D_\mathcal{R}\mathsf{s}(\mathcal{X})$$

requiring that the entire mission space is covered, and that this is distributively known. As such, the algebraic framework with decision predicates encompasses the exploration semantics from Chapter 4. ◀

▶ **Example 6.46 (Robot gathering task).** The gathering task with robots $\mathcal{R}$ in a mission space $\mathcal{X}$, from Section 4.3.3, is modeled analogously to Example 6.45. Given

a protocol complex functor with decisions $\mathcal{F}_!$ modeling the gathering protocol (such as the averaging protocol in Example 6.42), we work within its free algebra $(\mathcal{I}^\infty, \phi_\mathcal{I})$ with decision values $V = \mathcal{X}$, where each $d(\mathbf{r}) \in \mathcal{X}$ represents robot $\mathbf{r}$'s current position. The gathering regions $\mathsf{gather}(\mathcal{X})$ from Definition 4.32 define what constitutes "close enough" for the task.

Iteration propositions now consist of individual decision predicates $\mathsf{p}_\mathbf{r}(\mathsf{U})$, defined by

$$M[\mathcal{I}^\infty], w \models \mathsf{p}_\mathbf{r}(\mathsf{U}) \text{ iff } w \text{ is a } (B, d)\text{simplex with } d(\mathbf{r}) \in U$$

and the initial position propositions $\mathsf{i}_\mathbf{r}(\mathsf{U})$ are evaluated in the $\mathcal{I}$ component of $\mathcal{I}^\infty$, where $\mathcal{F}_!$ has not been applied yet.

The conditions from Definition 4.32 translate directly to formulas in the language of iteration. For instance, eventual gathering is expressed as

$$M[\mathcal{I}^\infty] \models \bigvee_{U \in \mathsf{gather}(\mathcal{X})} \Diamond\Box \left( \bigwedge_{\mathbf{r} \in \mathcal{R}} \mathsf{p}_\mathbf{r}(\mathsf{U}) \right)$$

while starting validity uses the initial position propositions to relate initial global states to the VALIDITY condition of GATHERING. ◀

Examples 6.45 and 6.46 show how spatial task specifications from Chapter 4 can be encoded within the algebraic framework using decision predicates. The task properties translate directly into the language of iteration, and the free algebras provide execution structures analogous to system frames.

## 6.5 CONCLUSION

The previous chapters extracted structures from system executions. This chapter inverts this perspective by introducing protocol functors as a functorial definition of communication protocols. This categorical encoding makes explicit the structure-preserving properties of protocols, enabling their systematic comparison and iterative composition. The repeated rounds of communication are captured by free algebras, which are uniquely defined for each pair of protocol functor and initial cset configuration space. The relation between epistemic and simplicial structures, introduced in [GLR21; vDit+22], is revisited, now with a temporal

dimension that is internalized by the algebra structure. The versatility of our approach is attested by incorporating decision values and local decision functions to the protocol definition, where all previous results are almost immediately lifted to the more complex structure, up to the interpretation of semi-simplicial models.

Protocol functors (Definition 6.6) encode wait-freedom, obliviousness and round-based communication as functors $\mathcal{F} : \Gamma \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$, with examples ranging from immediate snapshot protocols to dynamic networks with crash failures. The Yoneda extension constructs protocol complex functors $\mathcal{F}_! : [\Gamma^{\mathrm{op}}; \mathbf{Set}] \to [\Gamma^{\mathrm{op}}; \mathbf{Set}]$ that apply protocols to entire csets while respecting face maps (Proposition 6.8). This endofunctor signature enables defining an algebra structure for iteration (Lemma 6.15), while being a varietor (Corollary 6.21) guarantees that free algebras exist for any initial cset. Such free algebra is canonical and collects all reachable global states through unbounded iteration, providing at last an execution space, with which Example 6.22 recasts the simplicial task solvability condition of Herlihy et al. [HKR14] in terms of csets. Semi-simplicial models (Definition 6.25) provide then the semantic structure to interpret formulas of temporal-epistemic logic, where the epistemic operators leverage the simplicial structure through link predicates, and temporal operators quantify over the round components of the free algebra. The knowledge gain property (Theorem 6.29) for positive epistemic formulas emerges directly from the categorical structure rather than requiring specific axioms, showing that the algebra maps preserve indistinguishability relations.

The framework extends naturally to protocol functors with decisions (Definition 6.36), enriching csets to dcsets that track explicit decision values. Concrete protocols (Definition 6.40) integrate local decision functions specifying numerical updates at each round, exemplified by averaging protocols (Example 6.42). This enrichment proves itself general enough to model multi-robot systems (Example 6.37), where positions serve as decision values. The adjunction between csets and dcsets formalizes the intuitive procedure of reducing robotic problems to computational ones (Remark 6.39). Decision predicates extend the logic to reason about the decisions, and by setting them to be spatial properties we recover analogous specifications to exploration and gathering robot tasks from Section 4.3.

Several theoretical directions remain open for extending this algebraic framework. The relationship between the temporal operators on free algebras and action models of dynamic epistemic logic requires precise formulation. While our

approach internalizes iterations within the algebra structure itself, it is unknown if the algebra maps capture the same dynamics on information as public announcements, or provides a genuinely different perspective on knowledge updates. Also on the algebra structure, we note that the dual perspective of coalgebras offers a complementary view to modeling protocols that should be explored, as they are known in the literature to model non-determinism in similar transition systems [Rut00] and may be used to model possible observations instead of state transitions. Enriching decision values with a metric or order structure could enable analyzing convergence properties categorically, as the recent work on categorical diffusion [Ghr+25] suggests that viewing decision spaces as quantale-enriched categories might relate our protocol iterations to a Laplacian-style dynamics on weighted lattices. The connection to task sheaves from Chapter 5 also deserves exploration, as each dcset equipped with decision predicates already resembles a sheaf-theoretic task specification. The equivalence between Kripke models and simplicial complex models from [GLR21] hints at potential functorial relationships bridging the logical and algebraic perspectives of Chapters 4 and 6.

An orthogonal direction for future work lies in exploring the transfer of results between distributed computing and distributed robotics, hinted by the adjunction between csets and dcsets (Definition 6.38). The equivalence between asynchronous luminous robot and wait-free shared memory models observed in [Alc+19] could be generalized through our framework by expanding the reduction technique suggested in Remark 6.39. Establishing such transfer theorems would enable applying the extensive distributed computing literature to robot coordination problems, while the geometric insights from robotics could inform protocol design. One approach would be to extend the decision values of dcsets, which are specified by deterministic functions defined on subsets of agents, to probability distributions instead. Similarly, the deterministic updates of local decision functions could be generalized to operation on these distributions, or include faulty behavior as in the case of Byzantine agents. Each such instantiation would inherit the free algebra construction and knowledge gain properties, providing a uniform categorical language for comparing and analyzing distributed models.

# 7

# Conclusion

This thesis started from a desire to reconcile the disconnect between how applied and theoretical roboticists reason about multi-robot systems. Multiple efforts have already been made across this gap, either by bringing algorithms into the control of dynamical systems or by adding mobility to distributed systems. One of the first translations of models came with Alcántara et al. [Alc+19], who demonstrated that the limitations to coordination found in distributed systems due to a lack of information were also found in robotic systems, once the physics was abstracted away. The key observation was that such limitations could be made evident by analyzing the distributed state space and its possible transformations through time, following Herlihy and Shavit [HS99] for distributed computing. This thesis systematically extended this observation by identifying common structures between hybrid dynamical systems and compositional I/O automata, staple models within mobile robotics and distributed computing, and demonstrated its usefulness by developing three complementary frameworks under this common perspective.

The first contribution, in Chapter 3, was to establish when reasoning within a computational abstraction transfers meaningfully to the dynamical model. This was possible by separating the physical and computational descriptions of a robot into connected subsystems. This convenient reshuffling of variables allowed us to define a system abstraction simulation relation, where the computational content

of both models could be matched, while the dynamics of all robots were abstracted into a common environment. With a common definition of local and global time, encoded geometrically in a time path, we showed that such a simulation relation was preserved along executions, justifying the adoption of the compositional model for the subsequent chapters.

The concept of a system frame provided the common execution structure we sought in Chapter 4. Constructed from the global states reachable by valid robot executions, its causality and indistinguishability relations encode how information propagates and where uncertainty remains. This geometric model of knowledge through time, extending the framework of Fagin et al. [Fag+95] with a branching-time structure, enabled the two complementary analyses developed in Chapters 4 and 5.

First, in Chapter 4, we asked what robots must know to reach different levels of coordination. A temporal-epistemic logic with spatial propositions derived from the topology of a compact mission space allowed us to separate physical coverage from communication requirements. For exploration, surveillance and gathering, we derived sufficient epistemic conditions for the tasks to be completed, formalizing the intuition that more complicated forms of collective behavior require more information, a perspective already largely studied in distributed computing.

Second, in Chapter 5, we asked when knowledge was enough to act upon. By externalizing the constraints of terminating decision functions into a finite sub frame, a system slice, we turned task solvability into a local constraint satisfaction problem. That is, we asked whether consistent decisions specific to each global state could be extended into a valid decision function. The existence of sections in the corresponding task sheaf characterized solvability, while its linearization reduced this question to looking at its zeroth cohomology, a proxy for the characterization computable through linear algebra with integers.

Chapter 6 inverted the approach: instead of analyzing a given set of executions, we used a protocol complex functor to generate them. By restricting our attention to oblivious wait-free communication, the specification of what can happen to a single global state for one round became sufficient to derive all reachable configurations through unbounded iteration. The functorial encoding of such protocols as endofunctors on chromatic semi-simplicial sets admits free

algebras that canonically collect the distributed state space in a single object. The temporal-epistemic logic interpreted on these algebras then revealed knowledge persistence as a property of the algebra structure, rather than requiring an external axiom, as in Chapter 4. We extended these to protocol complex functors with decision values, recovering spatial task specifications analogous to those of earlier chapters.

The generality of these approaches comes with assumptions that limit their current applicability. Physical realizability was introduced in Chapter 3 in order to transfer results from the computational abstraction back to dynamical models, as discrete executions may not be immediately realizable. This means that protocols generated in the compositional model in Chapter 5 should be verified as fitting the original robot dynamics, while the unidirectional analysis of Chapters 4 and 6 is unaffected. Perfect recall, while common in impossibility proofs under full-information assumptions, has been used throughout Chapter 4 in order to focus on the modeling of knowledge dynamics rather than carving specific bounds. Its relaxation to oblivious protocols is of interest, as robot dynamics may provide information that compensates for a shorter history, just as a temporal cut behavior avoided requiring a continuous coverage of the mission space for surveillance. Further limitations include the use of homogeneous systems throughout, where all robots share identical sensing and acting capabilities, and the explicit use of static task specifications in Chapters 5 and 6, excluding changing goals of reactive systems. Extending the frameworks to these settings would require non-trivial modifications, especially to the base space of the task sheaves, which encode specific system properties, though the generality of the space of decision functions suggests the possibility.

None of these frameworks originated in distributed robotics, yet each finds natural application once the execution structure is made explicit. What has been established is not a finished theory, but a collection of tools that were previously isolated, now available in a common language, admitting analysis through epistemic logic, sheaf theory and categorical algebra. The exercise of this versatility is a natural next step, such as using sheaf cohomology to compute solutions to concrete coordination problems and lifting impossibility results from distributed computing to constrained robotic settings. Exploring the spectral theory of cellular sheaves for iterative approximation of solutions is a promising theoretical

avenue, given the significant attention it has recently received in several domains with similar distributed behavior, from opinion dynamics and economic modeling to multi-agent coordination. The taxonomy of multi-robot systems from the knowledge perspective remains largely open. Which physical constraints create epistemic limitations absent in purely computational settings, and which are merely implementation details that disappear under the appropriate abstraction?

# REFERENCES


[AB11] S. Abramsky and A. Brandenburger. "The Sheaf-Theoretic Structure of Non-Locality and Contextuality". *New Journal of Physics* 13 (2011). doi:10.1088/1367-2630/13/11/113036 (cit. on pp. 32, 102).

[Alc+19] M. Alcántara, A. Castañeda, D. Flores-Peñaloza, and S. Rajsbaum. "The Topology of Look-Compute-Move Robot Wait-Free Algorithms with Hard Termination". *Distributed Computing* 32 (2019). doi:10.1007/s00446-018-0345-3 (cit. on pp. 2, 22, 26, 38, 95, 147, 181, 188, 189).

[ARL20] J. Armenta-Segura, S. Rajsbaum, and J. Ledent. "Two-Agent Approximate Agreement from an Epistemic Logic Perspective". In: Latin American Workshop on New Methods of Reasoning. 2020 (cit. on p. 67).

[ABD95] H. Attiya, A. Bar-Noy, and D. Dolev. "Sharing Memory Robustly in Message-Passing Systems". *J. ACM* 42 (1995). doi:10.1145/200836.200869 (cit. on p. 17).

[ACN23] H. Attiya, A. Castañeda, and T. Nowak. *Topological Characterization of Task Solvability in General Models of Computation*. 2023. URL: http://arxiv.org/abs/2301.13837 (cit. on p. 101).

[AC84] J.-P. Aubin and A. Cellina. *Differential Inclusions: Set-Valued Maps and Viability Theory*. Vol. 264. 1984. doi:10.1007/978-3-642-69512-4 (cit. on p. 47).

[BvDG22] P. Balbiani, H. van Ditmarsch, and S. F. González. "Asynchronous Announcements". *ACM Transactions on Computational Logic* 23 (2022). doi:10.1145/3481806 (cit. on p. 66).

[Bal+22] A. Baltag, N. Bezhanishvili, A. Özgün, and S. Smets. "Justified Belief, Knowledge, and the Topology of Evidence". *Synthese* 200 (2022). doi:10.1007/s11229-022-03967-6 (cit. on p. 67).

[BMS98] A. Baltag, L. S. Moss, and S. Solecki. "The Logic of Public Announcements, Common Knowledge, and Private Suspicions". In: *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge*. 1998 (cit. on p. 147).

[BMS16] A. Baltag, L. S. Moss, and S. Solecki. "The Logic of Public Announcements, Common Knowledge, and Private Suspicions". In: *Readings in Formal Epistemology: Sourcebook*. 2016. doi:10.1007/978-3-319-20451-2_38 (cit. on p. 66).

[Bat+20] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri. "Networks beyond Pairwise Interactions: Structure and Dynamics". *Physics Reports* 874 (2020). doi:10.1016/j.physrep.2020.05.004 (cit. on p. 12).

[BM15] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. 2015. doi:10.1007/978-3-319-17933-9 (cit. on p. 10).

[Bod+23] C. Bodnar, F. Di Giovanni, B. P. Chamberlain, P. Liò, and M. M. Bronstein. *Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs*. 2023. doi:10.48550/arXiv.2202.04579 (cit. on p. 102).

[BG93] E. Borowsky and E. Gafni. "Immediate Atomic Snapshots and Fast Renaming". In: PODC. 1993. doi:10.1145/164051.164056 (cit. on p. 15).

[Bre12] G. E. Bredon. *Sheaf Theory*. 2012 (cit. on p. 135).

[Cha91] B. Charron-Bost. "Concerning the Size of Logical Clocks in Distributed Systems". *Information Processing Letters* 39 (1991). doi:10.1016/0020-0190(91)90055-m (cit. on p. 40).

[Cig+25] G. Cignarale, S. Felber, E. Goubault, B. Hummes Flores, and H. Rincon Galeana. *Knowledge in Multi-Robot Systems: An Interplay of Dynamics, Computation and Communication*. 2025. doi:10.48550/arXiv.2501.18309 (cit. on pp. 3, 4, 38, 66).

[CFR24] G. Cignarale, S. Felber, and H. Rincon Galeana. *A Sufficient Epistemic Condition for Solving Stabilizing Agreement*. 2024. doi:10.48550/arXiv.2403.01025 (cit. on pp. 67, 95, 96).

[Cla+18] *Handbook of Model Checking*. 2018. doi:10.1007/978-3-319-10575-8 (cit. on pp. 66, 75).

[CMB06] J. Cortes, S. Martinez, and F. Bullo. "Robust Rendezvous for Mobile Autonomous Agents via Proximity Graphs in Arbitrary Dimensions". *IEEE Transactions on Automatic Control* 51 (2006). doi:10.1109/TAC.2006.878713 (cit. on p. 20).

[CGP15] É. Coulouma, E. Godard, and J. Peters. "A Characterization of Oblivious Message Adversaries for Which Consensus Is Solvable". *Theoretical Computer Science* 584 (2015). doi:10.1016/j.tcs.2015.01.024 (cit. on p. 16).

[Cur14] J. M. Curry. "Sheaves, Cosheaves and Applications". University of Pennsylvania, 2014 (cit. on pp. 32, 100, 101, 111, 113, 114, 131, 134).

[Das+16] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. "Autonomous Mobile Robots with Lights". *Theoretical Computer Science* 609 (2016). doi:10.1016/j.tcs.2015.09.018 (cit. on pp. 20, 38).

[De +20] V. De Iuliis, A. D'Innocenzo, A. Germani, and C. Manes. "Stability Conditions for Linear Discrete-Time Switched Systems in Block Companion Form". *IET Control Theory & Applications* 14 (2020). doi:10.1049/iet-cta.2020.0754 (cit. on p. 38).

[DI17] S. M. Dibaji and H. Ishii. "Resilient Consensus of Second-Order Agent Networks: Asynchronous Update Rules with Delays". *Automatica* 81 (2017). doi:10.1016/j.automatica.2017.03.008 (cit. on pp. 2, 12, 46).

[DBO17] H. T. T. Doan, F. Bonnet, and K. Ogata. "Model Checking of a Mobile Robots Perpetual Exploration Algorithm". In: *Structured Object-Oriented Formal Language and Method*. Vol. 10189. 2017. doi:10.1007/978-3-319-57708-1_12 (cit. on p. 66).

[DBO18] H. T. T. Doan, F. Bonnet, and K. Ogata. "Model Checking of Robot Gathering". *LIPIcs, Volume 95, OPODIS 2017* 95 (2018). doi:10.4230/LIPICS.OPODIS.2017.12 (cit. on p. 66).

[Dol+86] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. "Reaching Approximate Agreement in the Presence of Faults". *Journal of the ACM* 33 (1986). doi:10.1145/5925.5931 (cit. on p. 19).

[EF82] T. Elrad and N. Francez. "Decomposition of Distributed Programs into Communication-Closed Layers". *Science of Computer Programming* 2 (1982). doi:10.1016/0167-6423(83)90013-8 (cit. on p. 17).

[Fab+21] F. Fabiano, B. Srivastava, M. B. Ganapini, J. Lenchner, L. Horesh, and F. Rossi. "E-PDDL: A Standardized Way of Defining Epistemic Planning Problems". In: International Conference on Automated Planning and Scheduling. 2021 (cit. on p. 66).

[Fag+95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. 1995. doi:10.7551/mitpress/5803.001.0001 (cit. on pp. 3, 26, 28, 65, 67, 68, 74, 95, 98, 176, 190).

[Faj+16] L. Fajstrup, E. Goubault, E. Haucourt, S. Mimram, and M. Raussen. *Directed Algebraic Topology and Concurrency*. 2016. doi:10.1007/978-3-319-15398-8 (cit. on pp. 38, 40).

[FHR25a] S. Felber, B. Hummes Flores, and H. Rincon Galeana. *A Sheaf-Theoretic Characterization of Tasks in Distributed Systems*. 2025. doi:10.48550/arXiv.2503.02556 (cit. on pp. 4, 101).

[FHR25b] S. Felber, B. Hummes Flores, and H. Rincon Galeana. "Brief Announcement: A Sheaf-Theoretic Characterization of Tasks in Distributed Systems". In: International Colloquium on Structural Infor-

mation and Communication Complexity. 2025. doi:10.1007/978-3-031-91736-3_26 (cit. on pp. 4, 101).

[Fid88] C. Fidge. "Timestamps in Message-Passing Systems That Preserve the Partial Ordering". In: Australian Computer Science Conference. Vol. 10. 1988 (cit. on pp. 38, 40).

[FLP85] M. Fischer, N. Lynch, and M. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". *Journal of the ACM* (1985) (cit. on pp. 2, 19).

[FPS19] P. Flocchini, G. Prencipe, and N. Santoro. "Moving and Computing Models: Robots". In: *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*. 2019. doi:10.1007/978-3-030-11072-7_1 (cit. on p. 20).

[Flo+05] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. "Gathering of Asynchronous Robots with Limited Visibility". *Theoretical Computer Science* 337 (2005). doi:10.1016/j.tcs.2005.01.001 (cit. on p. 20).

[FP23] P. Fraigniaud and A. Paz. "The Topology of Local Computing in Networks" (2023) (cit. on p. 101).

[FKS21] K. Fruzsa, R. Kuznets, and U. Schmid. "Fire!" *Electronic Proceedings in Theoretical Computer Science* 335 (2021). doi:10.4204/EPTCS.335.13 (cit. on p. 66).

[Ghr+25] R. Ghrist, M. Lopez, P. R. North, and H. Riess. *Categorical Diffusion of Weighted Lattices*. 2025. URL: http://arxiv.org/abs/2501.03890 (cit. on pp. 35, 102, 188).

[GR22] R. Ghrist and H. Riess. "Cellular Sheaves of Lattices and the Tarski Laplacian". *Homology, Homotopy and Applications* 24 (2022). doi:10.4310/HHA.2022.v24.n1.a16 (cit. on p. 35).

[God18] J.-L. Godard. *Le Livre d'image*. Casa Azul Films, 2018 (cit. on p. 37).

[GJ09] P. G. Goerss and J. F. Jardine. *Simplicial Homotopy Theory*. 2009. doi:10.1007/978-3-0346-0189-4 (cit. on p. 30).

[GM20] G. Goren and Y. Moses. "Silence". *J. ACM* 67 (2020). doi:10.1145/3377883 (cit. on p. 66).

[Gou+25] E. Goubault, B. Hummes Flores, R. Kniazev, J. Ledent, and S. Rajsbaum. *A Categorical and Logical Framework for Iterated Protocols*. 2025. doi:10.48550/arXiv.2505.10071 (cit. on pp. 5, 31, 146).

[Gou+23] É. Goubault, R. Kniazev, J. Ledent, and S. Rajsbaum. "Semi-Simplicial Set Models for Distributed Knowledge". In: ACM/IEEE Symposium on Logic in Computer Science. 2023. doi:10.1109/LICS56636.2023.10175737 (cit. on pp. 29–31, 67, 99, 101, 115, 143, 146, 147, 169, 171, 176).

[GLR21] É. Goubault, J. Ledent, and S. Rajsbaum. "A Simplicial Complex Model for Dynamic Epistemic Logic to Study Distributed Task Computability". *Information and Computation* 278 (2021). doi:10.1016/j.ic.2020.104597 (cit. on pp. 28, 29, 101, 146, 173, 186, 188).

[GLR22] É. Goubault, J. Ledent, and S. Rajsbaum. "A Simplicial Model for KB4_n: Epistemic Logic with Agents That May Die". *LIPIcs, Volume 219, STACS 2022* 219 (2022). doi:10.4230/LIPICS.STACS.2022.33 (cit. on pp. 147, 169).

[GMT18] É. Goubault, S. Mimram, and C. Tasson. "Geometric and Combinatorial Views on Asynchronous Computability". *Distributed Computing* 31 (2018). doi:10.1007/s00446-018-0328-4 (cit. on p. 38).

[HM90] J. Y. Halpern and Y. Moses. "Knowledge and Common Knowledge in a Distributed Environment". *J. ACM* 37 (1990). doi:10.1145/79147.79161 (cit. on pp. 27, 66).

[Han+25] T. Hanks, H. Riess, S. Cohen, T. Gross, M. Hale, and J. Fairbanks. *Distributed Multi-agent Coordination over Cellular Sheaves*. 2025. doi:10.48550/arXiv.2504.02049 (cit. on p. 102).

[Han20] J. Hansen. "Laplacians of Cellular Sheaves: Theory and Applications". University of Pennsylvania, 2020 (cit. on pp. 32, 34, 101, 143).

[HG19a] J. Hansen and R. Ghrist. "Distributed Optimization with Sheaf Homological Constraints". In: *2019 57th Annual Allerton Conference on*

*Communication, Control, and Computing (Allerton)*. Annual Allerton Conference on Communication, Control, and Computing. 2019. doi:10.1109/ALLERTON.2019.8919796 (cit. on pp. 32, 102).

[HG21] J. Hansen and R. Ghrist. "Opinion Dynamics on Discourse Sheaves". *SIAM Journal on Applied Mathematics* 81 (2021). doi:10.1137/20M1341088 (cit. on pp. 32, 34, 102).

[HG19b] J. Hansen and R. Ghrist. "Toward a Spectral Theory of Cellular Sheaves". *Journal of Applied and Computational Topology* 3 (2019). doi:10.1007/s41468-019-00038-7 (cit. on pp. 34, 35).

[Hat01] A. Hatcher. *Algebraic Topology*. 2001 (cit. on pp. 32, 132).

[Her65] F. Herbert. *Dune*. 1965 (cit. on p. 65).

[HKR14] M. Herlihy, D. N. Kozlov, and S. Rajsbaum. *Distributed Computing through Combinatorial Topology*. 2014 (cit. on pp. 23, 26, 100, 101, 141, 146, 150, 167, 187).

[HS93] M. Herlihy and N. Shavit. "The Asynchronous Computability Theorem for T-Resilient Tasks". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing - STOC '93*. Symposium on Theory of Computing (STOC). 1993. doi:10.1145/167088.167125 (cit. on p. 101).

[HS99] M. Herlihy and N. Shavit. "The Topological Structure of Asynchronous Computability". *Journal of the ACM* 46 (1999). doi:10.1145/331524.331529 (cit. on pp. 2, 189).

[Hin62] J. Hintikka. *Knowledge and Belief*. 1962 (cit. on pp. 26, 66).

[IWF22] H. Ishii, Y. Wang, and S. Feng. "An Overview on Multi-Agent Consensus under Adversarial Attacks". *Annual Reviews in Control* 53 (2022). doi:10.1016/j.arcontrol.2022.01.004 (cit. on p. 38).

[Jau+01] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. 2001 (cit. on p. 10).

[Kas84] M. Kashiwara. "The Riemann–Hilbert Problem for Holonomic Systems". *Publications of the Research Institute for Mathematical Sciences* 20 (1984). doi:10.2977/prims/1195181610 (cit. on p. 32).

[Kha02] H. K. Khalil. *Nonlinear Systems*. 3rd ed. 2002 (cit. on pp. 7, 10).

[Kia+19] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez. "Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms". *IEEE Control Systems* 39 (2019). doi:10.1109/MCS.2019.2900783 (cit. on pp. 46, 184).

[Kni13] S. Knight. "The Epistemic View of Concurrency Theory". École Polytechnique, 2013 (cit. on p. 67).

[KO11] F. Kuhn and R. Oshman. "Dynamic Networks: Models and Algorithms". *Theoretical Computer Science* 42 (2011). doi:10.1145/1959045.1959064 (cit. on pp. 16, 147, 155).

[Kul+14] S. Kulkarni, M. Demirbas, D. Madappa, B. Avva, and M. Leone. "Logical Physical Clocks". In: International Conference on Principles of Distributed Systems. 2014. doi:10.1007/978-3-319-14472-6_2 (cit. on pp. 38, 40).

[Lam78] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". *Communications of the ACM* 21 (1978) (cit. on pp. 2, 16, 40).

[Lan02] S. Lang. *Algebra*. Vol. 211. 2002. doi:10.1007/978-1-4613-0041-0 (cit. on p. 132).

[LeB+13] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram. "Resilient Asymptotic Consensus in Robust Networks". *IEEE Journal on Selected Areas in Communications* 31 (2013). doi:10.1109/JSAC.2013.130413 (cit. on p. 46).

[Lee21] K. Lee. "Effect of Asynchronous Communications on Stationary Solutions for Discrete-Time Multiagent Systems". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51 (2021). doi:10.1109/TSMC.2019.2926696 (cit. on p. 13).

[LM22] K. Lee and G. Macias. "Asynchronous Distributed Average Consensus: A Switched System Framework for Biased Average Analysis". *International Journal of Control, Automation and Systems* 20 (2022). doi:10.1007/s12555-021-0114-0 (cit. on p. 38).

[Lib03] D. Liberzon. *Switching in Systems and Control*. 2003. doi:10.1007/978-1-4612-0017-8 (cit. on pp. 13, 14).

[LFM05] Z. Lin, B. Francis, and M. Maggiore. "Necessary and Sufficient Graphical Conditions for Formation Control of Unicycles". *IEEE Transactions on Automatic Control* 50 (2005). doi:10.1109/TAC.2004.841121 (cit. on p. 12).

[LSV03] N. Lynch, R. Segala, and F. Vaandrager. "Hybrid I/O Automata". *Information and Computation* 185 (2003). doi:10.1016/S0890-5401(03)00067-1 (cit. on p. 39).

[Lyn97] N. A. Lynch. *Distributed Algorithms*. 1997 (cit. on pp. 14, 17, 39, 41, 49, 51, 52, 54).

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*. 1971 (cit. on pp. 146, 152).

[MM94] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. 1994. doi:10.1007/978-1-4612-0927-0 (cit. on pp. 147, 166, 178).

[Mat88] F. Mattern. "Virtual Time and Global States of Distributed Systems". In: International Workshop on Parallel and Distributed Algorithms. 1988 (cit. on pp. 38, 40).

[Mos16] Y. Moses. "Relating Knowledge and Coordinated Action: The Knowledge of Preconditions Principle". *Electronic Proceedings in Theoretical Computer Science* 215 (2016). doi:10.4204/EPTCS.215.17 (cit. on p. 67).

[Nis24] S. Nishimura. "Defining Logical Obstruction with Fixpoints in Epistemic Logic". *Journal of Applied and Computational Topology* 8 (2024). doi:10.1007/s41468-023-00151-8 (cit. on p. 29).

[NSW24] T. Nowak, U. Schmid, and K. Winkler. "Topological Characterization of Consensus in Distributed Systems". *Journal of the ACM* 71 (2024). doi:10.1145/3687302 (cit. on pp. 101, 143).

[Nuñ+22] E. Nuño, A. Loría, E. Panteley, and E. Restrepo. "Rendezvous of Nonholonomic Robots via Output-Feedback Control Under Time-Varying Delays". *IEEE Transactions on Control Systems Technology* 30 (2022). doi:10.1109/TCST.2022.3144031 (cit. on p. 13).

[OM04] R. Olfati-Saber and R. Murray. "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays". *IEEE Transactions on Automatic Control* 49 (2004). doi:10.1109/TAC.2004.834113 (cit. on p. 12).

[OFM07] R. Olfati-Saber, J. A. Fax, and R. M. Murray. "Consensus and Cooperation in Networked Multi-Agent Systems". *Proceedings of the IEEE* 95 (2007). doi:10.1109/JPROC.2006.887293 (cit. on pp. 1, 38).

[Pnu77] A. Pnueli. "The Temporal Logic of Programs". In: *18th Annual Symposium on Foundations of Computer Science (Sfcs 1977)*. Annual Symposium on Foundations of Computer Science. 1977. doi:10.1109/SFCS.1977.32 (cit. on p. 66).

[Poi02] H. Poincaré. *La science et l'hypothèse*. 1902 (cit. on p. 100).

[Pos+24] *Hybrid and Networked Dynamical Systems: Modeling, Analysis and Control*. Vol. 493. 2024. doi:10.1007/978-3-031-49555-7 (cit. on pp. 11, 13, 38).

[Qui73] M. Quintana. *Caderno H*. 1973 (cit. on p. 145).

[RB05] W. Ren and R. Beard. "Consensus Seeking in Multiagent Systems under Dynamically Changing Interaction Topologies". *IEEE Transactions on Automatic Control* 50 (2005). doi:10.1109/TAC.2005.846556 (cit. on p. 12).

[Rie11] E. Riehl. "A Leisurely Introduction to Simplicial Sets" (2011) (cit. on p. 31).

[Rie23] H. Riess. "Lattice Theory in Multi-Agent Systems". University of Pennsylvania, 2023. URL: http://arxiv.org/abs/2304.02568 (cit. on pp. 34, 35).

[RRS22] H. Rincon Galeana, S. Rajsbaum, and U. Schmid. "Continuous Tasks and the Asynchronous Computability Theorem". In: *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Innovations in Theoretical Computer Science Conference (ITCS). Vol. 215. 2022. doi:10.4230/LIPIcs.ITCS.2022.73 (cit. on p. 101).

[Rob20] M. Robinson. "Assignments to Sheaves of Pseudometric Spaces". *Compositionality* 2 (2020). doi:10.32408/compositionality-2-2 (cit. on p. 102).

[Rob17] M. Robinson. "Sheaves Are the Canonical Data Structure for Sensor Integration". *Information Fusion* 36 (2017). doi:10.1016/j.inffus.2016.12.002 (cit. on pp. 32, 102).

[Rut00] J. J. M. M. Rutten. "Universal Coalgebra: A Theory of Systems". *Theoretical Computer Science* 249 (2000). doi:10.1016/S0304-3975(00)00056-6 (cit. on p. 188).

[Sau+17] K. Saulnier, D. Saldaña, A. Prorok, G. J. Pappas, and V. Kumar. "Resilient Flocking for Mobile Robot Teams". *IEEE Robotics and Automation Letters* 2 (2017). doi:10.1109/LRA.2017.2655142 (cit. on p. 12).

[Sto00] A. Storjohann. "Algorithms for Matrix Canonical Forms". ETH Zurich, 2000. doi:10.3929/ETHZ-A-004141007 (cit. on p. 140).

[SY99] I. Suzuki and M. Yamashita. "Distributed Anonymous Mobile Robots: Formation of Geometric Patterns". *SIAM Journal on Computing* 28 (1999). doi:10.1137/S009753979628292X (cit. on pp. 2, 20).

[The05] J. Theys. "Joint Spectral Radius : Theory and Approximations". Université Catholique de Louvain, 2005 (cit. on p. 14).

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. 2005 (cit. on p. 10).

[Trn+75] V. Trnková, J. Adámek, V. Koubek, and J. Reiterman. "Free Algebras, Input Processes and Free Monads". *Commentationes Mathematicae Universitatis Carolinae* 016 (1975). URL: http://dml.mathdoc.fr/item/105628/ (cit. on pp. 146, 165).

[TLV24] L. Tseng, G. Liang, and N. H. Vaidya. "Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs". *Distributed Computing* 37 (2024). doi:10.1007/s00446-024-00468-2 (cit. on pp. 147, 184).

[vBB07] J. van Benthem and G. Bezhanishvili. "Modal Logics of Space". In: *Handbook of Spatial Logics*. 2007. doi:10.1007/978-1-4020-5587-4_5 (cit. on p. 66).

[vDit+21] H. van Ditmarsch, É. Goubault, M. Lazić, J. Ledent, and S. Rajsbaum. "A Dynamic Epistemic Logic Analysis of Equality Negation and Other Epistemic Covering Tasks". *Journal of Logical and Algebraic Methods in Programming* 121 (2021). doi:10.1016/j.jlamp.2021.100662 (cit. on p. 29).

[vDit+22] H. van Ditmarsch, É. Goubault, J. Ledent, and S. Rajsbaum. "Knowledge and Simplicial Complexes". In: *Philosophy of Computing*. 2022. doi:10.1007/978-3-030-75267-5_1 (cit. on p. 186).

[vDHK07] H. van Ditmarsch, W. van der Hoek, and B. P. Kooi. *Dynamic Epistemic Logic*. 2007 (cit. on pp. 29, 146).

[Zee62] E. C. Zeeman. "Dihomology: I. Relations Between Homology Theories". *Proceedings of the London Mathematical Society* s3-12 (1962). doi:10.1112/plms/s3-12.1.609 (cit. on p. 32).

# INDEX


Agreement, 4, 68, 95, 96, 98, 99
Always, 66, 73, 75, 76, 78, 94, 96, 169, 177
Automata composition, 48, 52, 53, 70

Causal closure, 107–111, 126, 131, 139, 141
Causal consistency, 4, 100–102, 104, 109–114, 118, 119, 121, 122, 124, 125, 129, 133, 140–142
Cell category, 4, 32, 33, 102, 111, 113, 114, 118, 120, 122, 125, 141, 143
Cellular sheaf, 32–35, 100–104, 111, 114, 118, 125, 131, 132, 137, 140, 141, 143, 191
Chromatic semi-simplicial set, 4, 25, 30, 31, 101, 114–122, 136, 143, 146–157, 160–164, 166, 167, 169, 170, 172, 174, 176–178, 181, 186–188, 190
Chromatic simplicial complex, 24, 25, 28, 29, 31, 114, 153, 167
Communication protocol, 14, 15, 17, 18, 23, 25, 26, 31, 49, 51, 65, 66, 68, 86, 99, 104, 105, 145, 146, 150, 151, 153–155, 157, 160–164, 166–170, 172, 175, 177, 179, 182, 186–188
Cset with decision values, 142, 178–180, 182–184, 187, 188

Decision
  algebras, 146–148, 178–188
  function (algebras), 178, 181
  function (sheaf), 101, 104, 106, 108, 111, 115–118, 126, 137, 139, 141, 142, 144, 190
  sheaf, 4, 101–107, 109–111, 117–120, 122, 127, 129, 132–135, 140–142, 190
  space, 101, 102, 115–118, 124, 141, 143, 191
  terminating, 4, 100–111, 117, 125, 126, 129–131, 136, 139–142, 190
Determinism, 100, 103, 104, 106, 108–111, 117, 125, 126, 130, 131, 141
Distributed task, 17, 18, 24, 25, 66, 95, 101, 103, 105, 106, 108, 111, 114, 121, 131, 136, 137, 167, 177
Dynamical system, 7–9, 12, 13, 37, 42, 43
  controlled, 7–9, 11, 43

hybrid, 3, 7, 9, 13, 14, 38, 43, 46, 48, 56–58, 60–63
switched, 13, 14, 41, 46, 47, 58

Environment, 7, 10, 37, 39, 48–56, 59, 60, 65, 68, 70–72, 82, 98
Epistemic, 3, 14, 39, 42, 44, 46, 48, 49, 56, 57, 59, 60, 63–65, 73, 85, 93, 98, 173
Eventually, 27, 66, 73–75, 83, 84, 89, 93, 94, 96, 169, 177
Execution, 3, 4, 48, 55, 56, 60–63, 65–67, 69, 72, 74, 76, 77, 79–82, 94–96, 104–109, 111, 120, 141–143, 145, 148, 155, 172, 190, 191
Execution cut, 4, 101, 105–111, 113, 126, 130, 131, 136, 139, 141–143
Exploration, 4, 21, 66, 68, 72, 79–87, 89, 90, 92, 93, 98, 185, 190

Free algebra, 146, 147, 164–169, 171, 172, 174, 176, 182–188, 190

Gathering, 4, 12, 13, 21, 22, 66, 68, 73, 79, 93–99, 184–186, 190
Global, 3, 38–42, 45, 46, 52, 54–56, 58, 60, 63, 103, 111, 114, 125, 133, 140, 177, 190

I/O automaton, 17, 39, 48, 49, 51–54, 168, 179
Inevitably, 66, 74, 75, 80, 94, 96
Input projection, 115

Knowledge, 3, 18, 27, 29, 51, 65–67, 69, 70, 73, 76–78, 80, 81, 83–85, 88, 90, 91, 93, 97–100, 143, 145, 148, 168, 169, 172–174, 178, 190–192
common, 27, 169, 172, 174, 177
distributed, 27, 29, 66, 73, 78–80, 88, 99, 169, 172, 174, 185
mutual, 74

Local, 3, 25, 38–42, 45–47, 52, 55, 56, 58, 59, 63, 70, 73, 87, 100, 103, 110, 111, 114, 119, 125, 132, 133, 137, 140, 141, 149, 153, 177, 190
Look-Compute-Move, 2, 20–22, 26, 36, 38, 48, 51, 179, 181

Mission space, 3, 4, 66–68, 72, 73, 75, 76, 79–89, 93, 94, 96–99, 185, 190, 191
Multi-robot system
computational, 3, 4, 37, 38, 48, 53–58, 60–63, 65–70, 79, 80, 82, 85, 86, 89, 94, 98, 99, 102, 105, 106, 118, 124, 125, 130, 131, 136, 139, 141, 142, 145, 163, 179, 191
dynamical, 6, 37, 40, 41, 46–48, 57, 61, 85, 99, 163, 179

Next, 169, 172

Ontic, 3, 14, 39, 42, 44, 46, 49, 57, 63, 64, 80–83, 93, 94, 98

Perfect recall, 4, 51, 67, 70, 76–78, 98, 99, 106, 115, 191

Physical realizability, 61, 63, 98, 99, 179, 191
Positive epistemic formula, 76, 147, 173–175, 187
Proposition
  atomic, 26, 27, 29, 73, 75, 79, 93, 94, 98
  dynamic, 73, 79, 93
  initial position, 93, 96, 97, 186
  iteration, 168–171, 173, 177, 184–186
  position, 93, 94, 96
  space, 3, 66, 67, 72–74, 76, 78–84, 86, 93, 98, 100, 143, 148, 168, 169, 185, 187
  space-time, 86–88, 90, 92
  static, 3, 73, 75–77, 83, 92, 93, 97
Protocol complex functor, 146, 147, 153, 154, 161, 163, 166–171, 174–176, 182, 183, 187, 190
  with decisions, 179, 180, 182, 184–186, 191
Protocol functor, 4, 145–147, 151–158, 160–162, 164, 166, 168, 170, 178–181, 186, 187
  with decisions, 5, 148, 179, 181, 182, 184, 187

Robot
  computational, 18, 39, 48–56, 59, 60, 65–67, 70–73, 76–81, 83–91, 93, 94, 96–99, 103, 179, 185, 190, 191
  dynamical, 6, 40, 42–47, 57, 59, 85, 149, 191
Robot task, 20–23, 26, 93, 95, 187

Scheduler, 16, 41, 46, 47, 52, 54, 56, 70, 71, 76
Semantics
  free algebra, 147, 169, 171, 173–175, 184, 185, 187
  operational, 4, 55, 59
  space, 74, 76–79, 82–85, 90–92, 148, 163
Simplex, 23, 24, 29–31, 115, 117–120, 126, 132, 133, 137, 139, 143, 150, 153, 154, 156, 157, 162, 169, 171, 172, 174, 178, 186
Solution set, 47, 48, 61, 63, 163
Surveillance, 4, 66, 68, 72, 79, 86–93, 98, 190, 191
System abstraction, 3, 57–63, 148, 189, 191
System frame, 3, 4, 65–71, 74–76, 79, 82, 84, 91, 95, 96, 98–100, 102–105, 107–112, 114, 115, 130, 131, 136, 139, 141, 145, 146, 148, 163, 168, 170, 175, 186, 190
System run, 48, 56, 60, 61, 63, 69, 145, 163, 168
System slice, 4, 100–102, 104, 110–114, 117, 118, 120–127, 129–133, 137, 139–143, 190

Task sheaf, 4, 35, 100, 102, 104, 109, 114, 115, 118–121, 123–137, 139–

143, 145, 148, 163, 164, 188, 190, 191
agent, 101, 102, 121–124, 140, 141
restriction map, 4, 118–121, 125, 127, 131–134, 138, 139, 141
section, 101–103, 125–131, 133, 135–142, 144, 190
stalk, 118–120, 125–129, 131–135, 137, 142
Task solvability, 18, 101, 103–105, 143, 144
free algebras, 167
simplicial complex, 25, 147, 187
system frame, 100–106, 108, 109, 111, 114, 124, 125, 130, 131, 136, 137, 139–144, 190
Task specification, 17, 25, 102–105, 109, 167, 185
cset, 4, 100, 102, 103, 115–120, 123–126, 130, 136, 137, 139, 141, 142, 144
simplicial complex, 25, 114, 115
Terminal execution cut, 105, 106, 108, 109, 111, 112, 130, 131, 136, 140, 142
Time path, 3, 13, 16, 38, 40–42, 46, 47, 54–58, 63, 76, 190
Trajectory, 47, 48, 59–63, 148
Transition function, 49, 51–53, 55, 56, 59, 61, 69, 70

Varietor, 147, 165, 166, 182, 187

Zeroth cohomology, 4, 101, 103, 104, 132, 134–137, 139, 142, 143, 190, 191

ECOLE
DOCTORALE

**Titre :** La structure de l'information en robotique distribuée

**Mots clés :** robotique distribuée, calcul distribué, systèmes hybrides

**Résumé :** La robotique distribuée se situe à l'intersection de la robotique mobile et du calcul distribué, deux domaines étudiant la coordination sous incertitude par des approches complémentaires. Des efforts récents ont été menés pour les rapprocher, en adaptant des procédures algorithmiques au contrôle de systèmes dynamiques et en ajoutant la dimension spaciele à des modèles computationnels. Cette thèse développe les fondements mathématiques de cette intersection, organisés autour de la structure de l'information disponible aux robots. La première contribution établit que des systèmes dynamiques hybrides et des modèles computationnels discrets peuvent encoder la même information épistémique, justifiant des abstractions discrètes pour raisonner sur des robots à dynamique continue. Trois approches complémentaires sont développées à partir de ce modèle commun. La connaissance au cours d'une exécution est modélisée comme un cadre, où les états globaux sont reliés par causalité ou indiscernabilité. Des propositions spatiales dérivées de l'espace de mission fondent une logique temporelle-épistémique formalisant les contraintes de tâche. Des conditions suffisantes pour l'exploration, la surveillance et le rassemblement sont dérivées, séparant propriétés comportementales et de communication. Des faisceaux cellulaires reformulent la résolvabilité comme un problème d'extension locale-globale. Un faisceau de tâche assigne des décisions localement valides aux cellules d'une catégorie d'exécution, avec des restrictions imposant la cohérence par robot. La résolvabilité correspond à l'existence de sections globales, et la cohomologie de degré zéro de sa linéarisation fournit une procédure semi-calculable pour construire des solutions. Les foncteurs de protocole modélisent la communication comme des endofoncteurs préservant la structure sur les ensembles semi-simpliciaux chromatiques. Leurs algèbres libres accumulent toutes les configurations atteignables par itération non bornée, récupérant le processus génératif que les approches précédentes avaient abstrait. Une logique temporelle-épistémique sur ces algèbres exhibe la persistance de la connaissance comme propriété structurelle. Ensemble, ces approches relient les perspectives épistémique, algébrique et topologique sur la coordination.

**Title :** The structure of information in distributed robotics

**Keywords :** distributed robotics, distributed computing, hybrid systems

**Abstract :** Distributed robotics lies at the intersection of mobile robotics and distributed computing. The two fields study multi-agent coordination under behavioral uncertainty from complementary approaches. Recent efforts have been made to bridge them, adapting algorithmic procedures to the control of dynamical systems and adding mobility to purely computational models. This thesis develops the mathematical foundations for this intersection, organized around the structure of information available to coordinating robots. The first contribution establishes that hybrid dynamical systems and discrete computational models can encode the same epistemic information, justifying computational abstractions for reasoning about robots with continuous dynamics. This provides a common execution model from which three complementary frameworks are developed. The knowledge throughout an execution is modeled as a system frame, where global states are connected by causality or indistinguishability relations. Spatial propositions derived from the mission space topology ground a temporal-epistemic logic that formalizes task requirements. Sufficient epistemic conditions for exploration, surveillance and gathering are derived, separating behavioral from communication properties. Then, cellular sheaves defined from the same execution structure reformulate task solvability as a local-to-global extension problem. A task sheaf assigns locally valid decisions to cells of a category derived from the execution structure, with restriction maps enforcing robot-wise consistency. Task solvability corresponds to the existence of global sections, and the zeroth cohomology of its linearization provides a semi-computable procedure for constructing solutions. Finally, protocol functors model communication as structure-preserving endofunctors on chromatic semi-simplicial sets. Their free algebras accumulate all reachable configurations through unbounded iteration, recovering the generative process that previous frameworks abstracted away. A temporal-epistemic logic on these algebras exhibits knowledge persistence as a structural property rather than an external axiom. Together, these frameworks connect epistemic, algebraic, and topological perspectives on coordination.

**Institut Polytechnique de Paris**
91120 Palaiseau, France