UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO

Department of Pure and Applied Sciences

Ph.D. PROGRAMME IN

RESEARCH METHODS IN SCIENCE AND TECHNOLOGY

CYCLE XXXVIII

LOGICAL AND ALGEBRAIC INVESTIGATIONS INTO PROPERTIES OF CONCURRENT SYSTEMS

ACADEMIC DISCIPLINES: INFO-01/A, MATH-01/A, PHIL-02/A

Coordinator: Prof. Luca Lanci

Supervisor: Prof. Alessandro Aldini

Ph.D. student: Ludovico Fusco

ACADEMIC YEAR
2024/2025

Logical and Algebraic Investigations into Properties of Concurrent Systems

Ludovico Fusco

To my family

Abstract

This dissertation is intended to contribute to the logical and algebraic study of *behavioural properties* of concurrent systems from two complementary perspectives: a static one, focused on the compositional analysis of finite-trace properties in interleaving systems, and a dynamic one, exploring the modelling, specification, and verification of dynamic epistemic multi-agent systems. Accordingly, the work is organised into two parts.

In Part I, within the interleaving approach to concurrency, the thesis develops a proof-theoretic framework for the structural analysis of finite-trace properties, with particular emphasis on those defined via prefix-closure. The interaction of the prefix-closure operator and its residual (with respect to set-theoretic inclusion) with intersection, union, and language concatenation is investigated. The variety of *closure ℓ-monoids* is introduced as a minimal algebraic abstraction of finite-trace properties, amenable to description within an analytic proof system. As a proof-theoretic counterpart to these structures, the display-like sequent calculus LMC is introduced and shown to be sound and complete, to admit cut elimination, and to be decidable. Altogether, this approach supports an algebraic and proof-theoretic analysis of finite-trace property patterns of verification relevance, while providing a compositional treatment of safety, liveness, and related notions.

In Part II, a unified approach to the modelling and verification of knowledge-based dynamic multi-agent systems is developed. A new model of computation, called a *Kripke labelled transition system* (KLTS), is introduced, combining labelled transition systems with multi-agent Kripke frames to support uniform reasoning about system dynamics and the evolution of agents' knowledge. On this basis, the logic EHML is defined by combining Hennessy–Milner Logic with the normal multimodal logic $\mathsf{S5}_n$. A Hilbert-style proof system for EHML is provided; soundness and completeness with respect to the intended semantics, as well as decidability, are established. The logic is

shown to be associated with a bisimulation that characterises behavioural equivalence in KLTS-based models of the systems of interest. Finally, building on the KLTS semantics, the process-algebraic language ECCS is introduced to support modular specification and analysis of concurrent multi-agent systems, with running examples drawn from real-world multi-agent standards.

Overall, this thesis aims to advance the mathematical foundations of the formal analysis and verification of concurrent systems, fostering the interaction between logical/algebraic methods and computational modelling techniques.

Contents

Introduction **1**

I Structural analysis of finite-trace properties in concurrent systems **9**

1 Algebraic background **10**
1.1 Preliminaries on universal algebra 10
1.1.1 Fundamental constructions 11
1.1.2 Varieties, equations, and free algebras 13
1.2 On (pre)ordered monoids 16
1.2.1 Basic concepts 16
1.2.2 Divisibility in monoids 17
1.3 Unary residuation 20

2 Modelling framework for concurrent systems **23**
2.1 LTSs and their trace-based specification 23
2.2 Trace properties for LTSs 26
2.2.1 The mixed case 26
2.2.2 The finite case 28

3 Algebraic theory of finite-trace properties **31**
3.1 Prefix-closure 31
3.2 On the operator ◫ 34
3.3 Closure ℓ-monoids 37

4 The Gentzen-style system LMC **41**
4.1 Calculus design . . . 41
4.2 Algebraic completeness . . . 45
4.3 Cut elimination . . . 48
4.3.1 A (weak) mix rule for LMC . . . 49
4.3.2 Mix elimination for $\mathsf{LMC}_{\ltimes}$ . . . 50
4.4 Decidability . . . 56
4.4.1 T-normalisation . . . 57
4.4.2 Contraction control . . . 62
4.4.3 Concise proofs . . . 69
4.5 Related work . . . 71

II Behavioural analysis of dynamic epistemic multi-agent systems **72**

5 Epistemic Hennessy–Milner Logic **73**
5.1 Kripke labelled transition systems . . . 74
5.2 Syntax and semantics . . . 77
5.3 Axiomatisation, completeness, and decidability . . . 83
5.4 A Hennessy–Milner theorem for EHML . . . 90
5.5 Related work . . . 92

6 A process algebra with KLTS semantics **96**
6.1 Process terms of ECCS . . . 97
6.2 From process terms to ECCS-agents . . . 100
6.3 Modelling concurrent ECCS-agents . . . 102

Conclusion **112**

List of Figures

2.1 Request-response protocol. 24

3.1 Left divisibility in $\mathbf{End}_A$. 40

6.1 Portion of the KLTS underlying the pool of agents of the FIPA QIP. . . 109
6.2 Portion of the KLTS underlying the pool of agents of the FIPA CNIP. . 111

List of Tables

3.1 The structure of elements of $\Diamond(P \cdot Q)$. 32

4.1 The Gentzen-style system LMC. 43
4.2 Multiplicity maxima for elements of $\bigcup_{\Delta \prec \Gamma} T(\Delta, \sqcap)$ occurring in Γ. . . . 63

5.1 Hilbert-style proof system for EHML. 83

6.1 Semantic rules for sequential processes 99
6.2 Semantic rules for a pool of agents 104

Acknowledgements

First and foremost, I would like to express my heartfelt thanks to my supervisor, Alessandro Aldini, for guiding my research over these years with care, dedication, and expertise. Alessandro is a truly wonderful person: kind, curious, open-minded, and scientifically rigorous. Through his mentorship, I not only had the opportunity to deepen my knowledge of concurrency theory, system verification, and computational logic, but, more importantly, I learned to approach problems like a computer scientist, developing a mindset that complements my background in logic and philosophy. I feel incredibly fortunate to have worked with him, and I have benefited enormously from his thoughtful and perceptive comments on my research, as well as from his constant support and encouragement.

One of the most important experiences of my doctoral journey, both professionally and personally, was undoubtedly my visit to the Mathematical Institute at the University of Bern, where I spent most of the Spring Semester 2025 with George Metcalfe and his group. I cannot fully express how much I owe to George, one of the most brilliant logicians I have ever met, and truly one of the finest people I know. From the very beginning, he welcomed me with great kindness, immediately making me feel at home and involving me in many activities both within and beyond the university. Thanks to his insightful and constructive feedback, my time in Bern was very fruitful: there, I learned a great deal and was able to develop some key results in this thesis. I also have a special debt of gratitude towards George for kindly agreeing to review this dissertation and for providing valuable comments that have greatly improved the work.

Words also fail me when trying to thank Mario Piazza, an outstanding logician and philosopher whom I have had the privilege of knowing since my undergraduate years. Mario has played a crucial role in my intellectual development, constantly encouraging me to engage with deep and wide-ranging questions. At the Scuola Normale Superiore of Pisa, I attended, as an external student, several of his lectures, which had a lasting

impact on the way I understand logic and its connections with other disciplines. His conceptual depth and human qualities have been a constant source of inspiration to me. I am also sincerely grateful to him for carefully reviewing this dissertation. His profound comments have provided me with countless insights, not only for improving this work but also for shaping my future research.

The development of this thesis has also benefited greatly from the invaluable feedback of Francesco Paoli, who holds a very special place in my academic journey. Francesco is an extraordinarily versatile and insightful logician, as well as a refined universal algebraist. Much of what I know about algebraic logic I owe to him, and his inspiring guidance as one of the co-advisors of my Master's thesis helped me take my very first steps in research. Throughout these years, he was always willing to offer suggestions on my work, each resulting in genuine improvements. I am furthermore deeply grateful to Francesco for involving me in a fascinating side project in universal algebra which led to the publication of our paper *Enriched Płonka sums*, of which I am very proud.

There are many other people who, in one way or another, have contributed to improving this dissertation through their comments and suggestions. Among them, I would especially like to mention Nick Galatos and Peter Jipsen, whose insights were crucial to the development of this work. During and after my time in Bern, my friend Simon Santschi provided valuable feedback on several parts of this thesis, carefully discussing a number of proofs; his observations were always illuminating, and I am very grateful to him. I would also like to thank Augusto Antonio Basilico, a long-time friend, who read parts of this thesis with remarkable care, identifying mistakes that I would not otherwise have noticed.

If the University of Urbino is such a wonderful place to study logic, this is without any doubt largely due to Pierluigi Graziani, a person who deserves special thanks. From my very first days as a PhD student, Pierluigi offered me friendship, support, and invaluable advice, and was always available to help in times of difficulty, just as he was always ready to celebrate every milestone achieved. The consistent attention to the needs and well-being of doctoral students that I have found in Pierluigi is something truly exceptional. He taught me so much about the practical aspects of academic life, from the organisation of activities to the importance of doing one's best to contribute to the effective functioning of one's research community. I have tried to learn as much as I could from him.

I would also like to warmly thank Marco Bernardo and Claudio Antares Mezzina, two outstanding theoretical computer scientists, with whom I have had the pleasure of exchanging ideas and engaging in many fruitful discussions over the years, both within the Department of Pure and Applied Sciences and at the many workshops and conferences we attended together. I have always come away from our conversations enriched. Alongside Alessandro, Pierluigi, Marco, and Claudio, another person with whom I shared much of my time in the department, as well as many conference trips, is my friend Andrea Esposito, an irreplaceable companion in academic adventures—including an epic and seemingly endless train journey from Pesaro to Paris, which would have been far less bearable without him. I have had many occasions to appreciate his intelligence, wit, and generosity, and I am grateful to him for the many enjoyable conversations we have shared on the most diverse topics.

Beyond Urbino, many people have played an important role in my journey.

Among the members of the Bern Logic Group, in addition to Simon Santschi, I would like to thank Jim de Groot, Isabel Hortelano Martín, and Niels Vooijs. All four of them welcomed me into their office at Alpeneggstrasse 22 with exceptional warmth and kindness. They are wonderful people whom I am truly fortunate to call friends.

Special thanks also go to: Paolo Aglianò, for introducing me to his amazing research group at the University of Siena; my early mentors Luca Bellotti, Enrico Moriconi, and Giacomo Turbanti, without whose guidance I would not have come this far; Stefano Bonzio, for encouraging me to apply for the PhD in Urbino; Antonietta Cantini, for her unwavering support and wise advice; Agata Ciabattoni, for helping to make my visit to Bern possible; Davide Fazio, for our many stimulating exchanges; Andrea Sabatini, for all the good times we shared talking about logic and life; Nino Salibra, for his pleasant and engaging company during one of his visits to Urbino; Gavin St. John, for his constant encouragement and positivity; and Matteo Tesi, for his insightful comments on my work during some of our meetings.

Finally, I would like to thank my family: my father Roberto, my mother Lina, and my sister Agnese. Without you, I would not be who I am.

Urbino, March 2026

Introduction

Concurrent systems are ubiquitous, both in nature and in engineered infrastructures, and arise whenever one is confronted with collections of dynamic objects (agents) that perform tasks either synchronously or asynchronously while processing and exchanging information. Typical examples include distributed systems [46] (systems consisting of spatially distributed and interconnected components) and reactive systems [155] (systems capable of responding to environmental stimuli, for instance by adapting to adverse conditions). More generally, concurrent systems are those systems whose observable behaviour emerges from the interaction among their components. It is easy to see that the above categories encompass a wide range of concrete—and often very different—phenomena: computer networks, smart grids, beehives, ant colonies, flocks of birds, human social groups, and many other entities can be modelled within the paradigm of concurrency theory. Moreover, at least in principle, such systems can also be verified, that is, one can rigorously determine—by means of formal methods—whether or not they satisfy given properties.

We do not attempt here to outline a history (or a philosophy) of concurrency theory, but merely recall its main conceptual ingredients.

1. *Information.* The evolution of a concurrent system cannot be divorced from some notion of information, whose manipulation and transmission inherently determine the system's dynamic nature.

2. *Behaviour* and *interaction.* As an entity that can be precisely localised in space and time, a concurrent system is observed in terms of events corresponding to the actions it exhibits, that is, its visible behaviour. The latter is fully characterised by the individual behaviours of the agents or components from which it is built, insofar as these behaviours interact. The major conceptual innovation introduced by concurrency theory lies precisely in reformulating the notion of computation

in terms of behaviour and interaction [15, 124].

3. *Analysis* and *verification.* Designing computational architectures that exhibit concurrency requires the constant development of formal techniques to ensure that system executions align with the intended implementations [155]. There are several ways to model a concurrent system, ranging from Petri nets [137] to labelled transition systems [98] and event structures [133], depending on the perspective on concurrency one intends to adopt (see also [171]). Building on notions of behaviour arising from such models, one is naturally led to the development of mathematical methods and tools for system analysis and verification. A wide variety of approaches have been proposed to address these aspects, most of which are centred on a formal semantics for concurrent processes, according to the classical tripartition into axiomatic [47], denotational [130], and operational semantics [123].

This dissertation aims to contribute to the study of the formal characterisation of *behavioural properties* of concurrent systems from two complementary perspectives: a static one and a dynamic one. Accordingly, it is divided into two parts.

- *Part I. Structural analysis of finite-trace properties in concurrent systems.* In this part, which builds upon and extends [68], we analyse system properties as properties of finite behaviours, that is, executions exhibiting a finite number of actions. The perspective is static in that we study behavioural properties as mathematical objects in their own right, while abstracting patterns that arise from system dynamics. In particular, we investigate the structure of properties by combining tools from universal algebra and structural proof theory.

- *Part II. Behavioural analysis of dynamic epistemic multi-agent systems.* In this part, based on [8] (which, in turn, extends and builds on [5]), we study a particular class of concurrent systems in which the classical computational dynamics governing system evolution are integrated with an abstract notion of knowledge—namely, the information possessed by the system components. To this end, we develop a unified approach comprising a novel model of computation, an associated modal logic, a logic-induced bisimulation, and a specification language, together providing a coherent framework for reasoning about dynamic epistemic multi-agent systems.

We now provide a more detailed overview of the contents of Parts I and II.

Part I

In many algorithmic verification frameworks, notably model checking [16], computations are modelled as *execution traces* derived from high-level specifications such as automata or labelled transition systems (LTSs). This approach dates back to the very origins of concurrency theory as a distinct branch of computer science [113], as it is implicitly adopted in Dijkstra's solution to the mutual exclusion problem [55]. Models of computation that admit a trace-based specification may differ considerably from one another[1]. In each case, a trace is a finite or infinite sequence of objects defined in terms of the underlying model's states and/or actions, possibly equipped with additional data allowing to properly encode system executions. This naturally supports the representation of key system policies in terms of *trace properties* (i.e., sets of execution traces) and provides a foundation for their classification according to the well-established *safety-liveness dichotomy* [10, 112]. The strength of this methodology lies in the fact that traces, being essentially strings over alphabets, can be conveniently treated in language-theoretic terms. Modulo a labelling of states with propositions, trace properties thus become simply string properties expressing *patterns* of system behaviours.

In the formal methods literature, it is common to find system executions represented as infinite traces, since programs may not terminate. It follows that most classical definitions and results concerning behavioural properties and their verification are formulated with respect to infinite traces—a choice reflected in the semantics of most temporal logics for computer science [54]. However, applied research has shown compelling reasons to adopt *finite traces* in many different contexts, and a variety of formalisms has been introduced for specifying and verifying properties of finite computations (see, e.g., [21, 50, 134, 150]). In this setting, a finite-trace property (henceforth, *f-property*) is a formal language in the usual sense, i.e., a subset of (the universe of) a free monoid $\mathbf{F}_\Sigma = \langle \Sigma^*, \cdot, \varepsilon \rangle$. Most interestingly, the fundamental classes of safety and liveness f-properties admit an elegant characterisation in terms of prefix-closure [136, 149]. Indeed, $P \subseteq \Sigma^*$ models a safety f-property if it is *prefix-closed*—that is, if P consists of all strings that are prefixes of some string in P itself (equivalently, if P contains every

[1] For an overview of examples, see [41, §7.2].

prefix of each of its strings). By contrast, P expresses a liveness f-property when every string in Σ^* admits an extension belonging to P—that is, if the prefix-closure of P is Σ^*. It is well known that prefix-closure acts as a topological closure operator on $\wp(\Sigma^*)$, yielding a topology on Σ^* whose closed and dense sets model, respectively, safety and liveness f-properties. Depending on the operations defined on it, a set of the form $\wp(\Sigma^*)$ can serve as the universe of algebras belonging to several well-known classes, including the varieties of residuated lattices [73], residuated Boolean algebras [95], Kleene algebras [105], and action algebras [142]. These varieties provide the algebraic semantics for major *substructural logics* [73, 121, 135], which thus offer natural tools for reasoning about f-properties across different levels of granularity. In Part I, we are interested in identifying a suitable framework within *structural proof theory* for addressing questions of the following kind:

> Given $P, Q_1, \ldots, Q_n \subseteq \Sigma^*$, if P is constructed from $Q_1, \ldots, Q_n$ by applying certain operations (e.g. $\cap$, $\cup$, or language concatenation), what can be *proved* about the structure of the prefix-closure of P?

Our purpose is therefore to provide an analytic calculus enabling the structural analysis of many property patterns central to formal verification, while also affording a *compositional account* of safety, liveness, and related notions.

The design of such a proof system crucially depends on the choice of an underlying *algebraic model* for f-properties. To begin with, for a free monoid $\mathbf{F}_\Sigma = \langle \Sigma^*, \cdot, \varepsilon \rangle$, we endow $\wp(\Sigma^*)$ with the structure of a bounded ℓ-monoid, taking $\cap$, $\cup$, and language concatenation as fundamental operations. We then expand this algebra, denoted by $\wp(\mathbf{F}_\Sigma)$, by two additional operations:

- A (forward) diamond $\Diamond$ corresponding to prefix-closure;
- A (backward) box $\boxslash$ mapping a property $P \subseteq \Sigma^*$ to the property $\boxslash P$ such that $w \in \boxslash P$ iff every prefix of w belongs to P.

Our box operation is the *residual* [27, 121] of prefix-closure with respect to set-theoretic inclusion: that is, for $P, Q \subseteq \Sigma^*$, $\Diamond P \subseteq Q$ iff $P \subseteq \boxslash Q$. Moreover, as we shall prove, $\boxslash$ plays a key role in the mathematical foundation for the finite-trace counterpart of a well-established technique for specifying safety properties through the combination of future and past modalities in temporal logic [115].

We study the equational definability of this expanded structure, denoted by $\wp(\mathbf{F}_\Sigma)_+$, starting from the observation that, in a free monoid, the prefix order coincides with the *left divisibility* relation. We first show that divisibility preorders on monoids always satisfy the *Riesz Decomposition Property* (*RDP*) [12, 148]. This result allows us to characterise the general behaviour of inverse image operators (like our $\Diamond$) associated with such preorders. We then proceed with a more fine-grained examination of the algebraic properties of prefix-closure and provide a number of equations describing the interaction of $\Diamond$ and $\boxdot$ with the ℓ-monoid structure.

Next, we introduce the variety $\mathfrak{LMC}$ of *closure ℓ-monoids* as a minimal algebraic model for f-properties to be captured within an analytic proof system. Concretely, we work with division-free reducts of bounded distributive residuated lattices equipped with residuated pairs of unary modalities of the form forward diamond/backward box, where the diamond is a topological closure operator satisfying the inequation $\Diamond(x \cdot y) \leq \Diamond x \cdot \Diamond y$. The algebra $\wp(\mathbf{F}_\Sigma)_+$ turns out to be a closure ℓ-monoid; however, it does not generate $\mathfrak{LMC}$.

Finally, we introduce LMC, a sound and complete Gentzen-style calculus for closure ℓ-monoids. In designing it, given the expressiveness of our algebraic framework, we decided to follow the approach established by Belnap's *Display Logic* [22], where the Gentzen terms occurring in derivations are constructed by combining formulas through *structural operators* mirroring the behaviour of the logical connectives. Unary residuation and, more generally, relationships between connectives arising from the category-theoretic notion of *adjunction* [96] can be conveniently treated in display systems [37, 38, 79, 82, 169]. However, with a view to future work on the effective implementability of our research, we do not resort to the full framework of Display Logic and introduce only the minimal amount of structural machinery required to establish completeness. As a result, LMC is a single-conclusion system with structural operators appearing only on the left sides of sequents.

We design LMC building on the division-free fragment of the Distributive Full Lambek Calculus [72, 103, 146], which includes structural operators for monoid multiplication and meet. In particular, the structural meet enables the derivation of the distributive laws for the lattice operations—an idea independently pioneered by Dunn [56] and Minc [126] in their work on the positive relevant logic R^+. The syntax of structural terms also includes a structural diamond, whose rules—together with those for the modal operators—are taken from Moortgat's system $\mathsf{NL}(\Diamond)$ [127]. We show that

LMC enjoys cut elimination. This result is obtained as a corollary of a mix elimination theorem, established using a technique originally devised in the context of hypersequent calculi for fuzzy logics [120] and later employed in the cut elimination of the hypersequent calculus CSemFL [121] for commutative, semilinear pointed residuated lattices. Finally, we provide a decidability result for LMC by adapting Gentzen's classical proof for LK [75, 76].

Outline of Part I In Chapter 1, we provide the necessary background for our algebraic analysis of f-properties. In Chapter 2, we introduce our reference modelling framework for concurrent systems. In particular, we choose to work within the *interleaving paradigm*, starting from a notion of trace derived from the computational model of labelled transition systems (LTSs). Chapter 3 deals with the algebraic theory of f-properties: we begin by studying the concrete structure $\wp(\mathbf{F}_\Sigma)_+$ (Sections 3.1 and 3.2) and proceed to the variety of closure ℓ-monoids (Section 3.3). In Chapter 4 we present the Gentzen-style system LMC (Section 4.1), and establish completeness (Section 4.2), cut elimination (Section 4.3), and decidability (Section 4.4). We conclude by discussing related work (Section 4.5).

Part II

The study of knowledge-based interactions in dynamic multi-agent systems spans several fields, including logic, philosophy, and theoretical computer science. In particular, concurrency theory and modal logic underlie two closely related strands of research that intersect across several application domains, such as the formal verification of security properties of communication protocols (see [53] for a survey and, in particular, [17, 20, 35, 125]). The combination of approaches from these two areas has been extensively explored in the literature (see, e.g., [100, 101, 85, 52]), but is rarely conducted within a modelling setting supporting the specification and the simultaneous analysis of dynamic and epistemic properties of systems, for instance in a process-algebraic paradigm.

The goal of Part II is to develop a unifying framework for the specification and analysis of knowledge-based dynamic interacting systems, starting from a new model of computation that encompasses both LTSs and multi-agent Kripke frames. Our model of computation, which we call a *Kripke labelled transition system* (*KLTS*), implements the

natural idea of linking each state of an LTS to a multi-agent Kripke frame, thereby providing a uniform foundation for modelling and reasoning about both system dynamics and the evolution of agents' knowledge.

After presenting KLTSs, we introduce EHML (*Epistemic Hennessy–Milner Logic*), a formalism obtained by merging Hennessy–Milner Logic HML [90] with the normal multimodal logic $\mathsf{S5}_n$ [87]. The models of EHML are "epistemic" KLTSs, that is, KLTSs whose states are associated with frames for $\mathsf{S5}_n$. Although EHML was originally introduced in [5], we present here a syntactically refined version of it, together with a new Hilbert-style proof system, and prove soundness and completeness with respect to its intended semantics. Moreover, by leveraging classical decidability results [87] for K_n and $\mathsf{S5}_n$, we provide a decision procedure for provability in EHML. We also establish a Hennessy–Milner theorem for our logic, showing that it induces a bisimulation, which, for image-finite models, exactly matches modal equivalence between states. Finally, we situate our logic within the existing literature, paying particular attention to its relationship with frameworks for *combining logics* [34] and *dynamic epistemic logic* [165].

To emphasise the expressiveness and usability of the KLTS model, we define on top of it ECCS, a process-algebraic language for the specification and verification of real-world concurrent multi-agent systems. The language is organised in layers in order to facilitate the description of agent behaviour, network topology, and communication policies underlying knowledge transfer. The presentation is accompanied by running examples drawn from the FIPA standards for heterogeneous and interacting agents and agent-based systems [141].

Outline of Part II In Chapter 5, we begin by developing the theory of KLTSs (Section 5.1). We then introduce the syntax and semantics of EHML (Section 5.2) and present completeness and decidability results (Section 5.3). Subsequently, we prove our Hennessy–Milner theorem for EHML (Section 5.4) and conclude the chapter with a discussion of related work (Section 5.5).

In Chapter 6, we introduce the process calculus ECCS by first defining a basic CCS-inspired calculus for sequential processes with value passing (Section 6.1). We then define the notion of an ECCS-agent, whose behaviour can be specified through our process algebra (Section 6.2). Finally, we provide operational semantic rules for modelling concurrency and interactions among ECCS-agents (Section 6.3).

Note to the reader

Throughout this thesis, $\curlywedge$, $\curlyvee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, and $\exists$ denote metatheoretical conjunction, disjunction, material implication, material equivalence, universal quantification, and existential quantification, respectively. Our underlying set-theoretic framework is naïve set theory, under the assumption of the Axiom of Choice.

Part I

Structural analysis of finite-trace properties in concurrent systems

Chapter 1

Algebraic background

We begin by recalling some basic algebraic concepts and results that will be used in the subsequent chapters. For background and terminology not covered here, we refer the reader to standard sources in universal algebra [23, 30, 116], lattice theory [48, 81], residuation theory [27], and the general theory of ordered algebraic structures [65].

1.1 Preliminaries on universal algebra

Recall that, for $\mathscr{F}$ a set of operation symbols, a (*similarity*) *type* over $\mathscr{F}$ is a function $\nu\colon \mathscr{F} \to \mathbb{N}$ assigning to each $f \in \mathscr{F}$ an integer $\nu(f)$ called the *arity* of f. We follow the customary convention of identifying types $\nu\colon \mathscr{F} \to \mathbb{N}$ with sequences $\langle \nu(f) \rangle_{f \in \mathscr{F}}$, with arities listed in non-increasing order. As usual, algebras and their universes are denoted using boldface and italics, respectively. The symbol X will always denote a countable set of variables (ranged over by $x, y, z, \dots$). Fix a type $\nu\colon \mathscr{F} \to \mathbb{N}$. For $n \in \mathbb{N}$, let $\mathscr{F}_n$ be the set of all n-ary operation symbols in $\mathscr{F}$. Elements of $\mathscr{F}_0$ are called *constant symbols*. From now on, for notational convenience, the metavariable c will be used in place of f to denote constant symbols. We also define $\mathscr{F}_{\geq 1} := \bigcup_{n \geq 1} \mathscr{F}_n$.

An *algebra* of type ν (ν-*algebra*) is a pair $\mathbf{A} = \langle A, \{f^{\mathbf{A}}\}_{f \in \mathscr{F}} \rangle$, where A is a non-empty set, called the *universe* of $\mathbf{A}$, and for each $f \in \mathscr{F}$, $f^{\mathbf{A}}$ is a $\nu(f)$-ary operation on A. If $c \in \mathscr{F}_0$, then $c^{\mathbf{A}}$ is a distinguished element of A (called a *constant*). We denote by $\mathfrak{Alg}(\nu)$ the class of all ν-algebras. Let $\nu : \mathscr{F} \to \mathbb{N}$ and $\nu' : \mathscr{F}' \to \mathbb{N}$ be types such that $\mathscr{F} \subsetneq \mathscr{F}'$ and $\nu = \nu' \restriction \mathscr{F}$. If $\mathbf{A} \in \mathfrak{Alg}(\nu)$ and $\mathbf{A}' \in \mathfrak{Alg}(\nu')$ have the same universe, then $\mathbf{A}$ is called a *reduct* of $\mathbf{A}'$, and $\mathbf{A}'$ an *expansion* of $\mathbf{A}$.

Let $\mathbf{A}$ and $\mathbf{B}$ be ν-algebras. A *homomorphism* $h\colon \mathbf{A} \to \mathbf{B}$ of $\mathbf{A}$ into $\mathbf{B}$ is a function $h\colon A \to B$ such that for all $f \in \mathscr{F}_{\geq 1}$ and all $a_1, \ldots, a_n \in A$:

$$h(f^{\mathbf{A}}(a_1, \ldots, a_n)) = f^{\mathbf{B}}(h(a_1), \ldots, h(a_n)).$$

Moreover, if $\mathscr{F}_0 \neq \emptyset$, then $h(c^{\mathbf{A}}) = c^{\mathbf{B}}$, for all $c \in \mathscr{F}_0$. If h is surjective, we say that $\mathbf{B}$ is a *homomorphic image* of $\mathbf{A}$. We adopt the standard terminology whereby injective and bijective homomorphisms are referred to as *embeddings* and *isomorphisms*, respectively. We write $\mathbf{A} \cong \mathbf{B}$ if $\mathbf{A}$ and $\mathbf{B}$ are isomorphic.

The set $T_\nu(X)$ of *terms* of type ν over X is defined by the BNF:

$$t \ ::= \ x \mid c \mid f(t_1, \ldots, t_{\nu(f)})$$

where $x \in X$, $c \in \mathscr{F}_0$, $f \in \mathscr{F}_{\geq 1}$, and $t_1, \ldots, t_{\nu(f)} \in T_\nu(X)$.

A term $t \in T_\nu(X)$ is said to be *n-ary* if all the variables occurring in it are taken from $\{x_1, \ldots, x_n\} \subseteq X$. In this case we write $t(x_1, \ldots, x_n)$. Note that constant symbols qualify as n-ary terms, for all n such that $\{x_1, \ldots, x_n\} \subseteq X$.

Terms of type ν form the universe of a ν-algebra with symbols in $\mathscr{F}$ acting as term-forming operations. The *term algebra* of type ν over X is the ν-algebra $\mathbf{T}_\nu(X)$ with universe $T_\nu(X)$ and operations defined by:

$$c^{\mathbf{T}_\nu(X)} := c \qquad\qquad f^{\mathbf{T}_\nu(X)}(t_1, \ldots, t_{\nu(f)}) := f(t_1, \ldots, t_{\nu(f)})$$

for all $c \in \mathscr{F}_0$, $f \in \mathscr{F}_{\geq 1}$, and $t_1, \ldots, t_{\nu(f)} \in T_\nu(X)$.

Let $\mathbf{A}$ be a ν-algebra. For $t(x_1, \ldots, x_n) \in T_\nu(X)$, the *term operation* $t^{\mathbf{A}}$ associated with t is defined by the following recursive clauses:

1. If $t = x_i$, for $1 \leq i \leq n$, then $t^{\mathbf{A}}$ is the *projection map* $\langle a_1, \ldots, a_n \rangle \mapsto a_i$.
2. If t is a constant symbol c, then $t^{\mathbf{A}}$ is the *constant map* $\langle a_1, \ldots, a_n \rangle \mapsto c^{\mathbf{A}}$.
3. If $t = f(t_1(x_1, \ldots, x_n), \ldots, t_m(x_1, \ldots, x_n))$, for an m-ary $f \in \mathscr{F}_{\geq 1}$, then for all $a_1, \ldots, a_n \in A$, $t^{\mathbf{A}}(a_1, \ldots, a_n) := f^{\mathbf{A}}(t_1^{\mathbf{A}}(a_1, \ldots, a_n), \ldots, t_m^{\mathbf{A}}(a_1, \ldots, a_n))$.

1.1.1 Fundamental constructions

Definition 1.1.1. Let $\mathbf{A}, \mathbf{B} \in \mathfrak{Alg}(\nu)$. We say that $\mathbf{B}$ is a *subalgebra* of $\mathbf{A}$ (written $\mathbf{B} \leq \mathbf{A}$) if $B \subseteq A$ and $f^{\mathbf{B}} = f^{\mathbf{A}} \restriction B$ for every operation symbol $f \in \mathscr{F}$.

A *subuniverse* of an algebra $\mathbf{A}$ is a set $B \subseteq A$ such that $f^{\mathbf{A}}[B] \subseteq B$. We write $\mathrm{Sub}(\mathbf{A})$ for the set of subuniverses of $\mathbf{A}$. Clearly, $\mathbf{B} \leq \mathbf{A}$ only if $B \in \mathrm{Sub}(\mathbf{A})$.

Lemma 1.1.2. [30, Thm. 6.3] *Direct and inverse images of homomorphisms preserve subuniverses.*

Definition 1.1.3. For $G \subseteq A$, define the *subuniverse of* $\mathbf{A}$ *generated by* G as the set

$$\mathrm{Sg}^{\mathbf{A}}(G) := \bigcap \{B \in \mathrm{Sub}(\mathbf{A}) \mid G \subseteq B\}.$$

Whenever $\mathrm{Sg}^{\mathbf{A}}(G) = A$, we denote $\mathbf{A}$ by $\mathbf{A}(G)$ and say that G *generates* $\mathbf{A}$.

Lemma 1.1.4. [30, Thm. 6.2] *Let* $h_1, h_2 \colon \mathbf{A}(G) \to \mathbf{B}$ *be homomorphisms. If* $h_1(a) = h_2(a)$ *for every* $a \in G$*, then* $h_1 = h_2$*.*

Definition 1.1.5. An equivalence relation θ on the universe of a ν-algebra $\mathbf{A}$ is a *congruence* on $\mathbf{A}$ if, for every $f \in \mathscr{F}$ and all $a_1, \ldots, a_{\nu(f)}, b_1, \ldots, b_{\nu(f)} \in A$:

$$\text{for all } 1 \leq i \leq \nu(f),\ \langle a_i, b_i \rangle \in \theta \text{ implies } \langle f^{\mathbf{A}}(a_1, \ldots, a_{\nu(f)}), f^{\mathbf{A}}(b_1, \ldots, b_{\nu(f)}) \rangle \in \theta$$

We denote by $\mathrm{Con}(\mathbf{A})$ the set of congruences on $\mathbf{A}$.

Definition 1.1.6. For a ν-algebra $\mathbf{A}$ and $\theta \in \mathrm{Con}(\mathbf{A})$, the *quotient algebra* $\mathbf{A}/\theta$ has universe A/θ and operations defined by

$$c^{\mathbf{A}/\theta} := [c^{\mathbf{A}}]_\theta \qquad\qquad f^{\mathbf{A}/\theta}([a_1]_\theta, \ldots, [a_{\nu(f)}]_\theta) := [f^{\mathbf{A}}(a_1, \ldots, a_{\nu(f)})]_\theta$$

for all $c \in \mathscr{F}_0$, $f \in \mathscr{F}_{\geq 1}$, and $[a_1]_\theta, \ldots, [a_{\nu(f)}]_\theta \in A/\theta$.

Definition 1.1.7. Let $\mathbf{A}/\theta$ be a quotient of a ν-algebra $\mathbf{A}$. The map $[-]_\theta \colon \mathbf{A} \to \mathbf{A}/\theta$ defined by $a \mapsto [a]_\theta$ is called the *natural homomorphism* of $\mathbf{A}$ onto $\mathbf{A}/\theta$.

Proposition 1.1.8. [30, Thm. 6.10] *Let* $\mathbf{A} \in \mathfrak{Alg}(\nu)$ *and* $\theta \in \mathrm{Con}(\mathbf{A})$*. Then* $\mathbf{A}/\theta$ *is a homomorphic image of* $\mathbf{A}$ *under* $[-]_\theta$*.*

Definition 1.1.9. If $h \colon \mathbf{A} \to \mathbf{B}$ is a homomorphism, then

$$\ker h := \{\langle a, b \rangle \in A^2 \mid h(a) = h(b)\}$$

is a congruence on $\mathbf{A}$ called the *kernel* of h.

The following theorem shows that homomorphic images and quotient algebras are equivalent constructions.

Homomorphism Theorem. [30, Thm. 6.12] *Let* $h\colon \mathbf{A} \to \mathbf{B}$ *be a surjective homomorphism. Then there exists an isomorphism* $\tilde{h}\colon \mathbf{A}/\ker h \to \mathbf{B}$ *such that* $h = \tilde{h} \circ [-]_{\ker h}$.

In what follows, let $\{\mathbf{A}_k\}_{k\in K}$ be an arbitrary indexed family of ν-algebras.

Definition 1.1.10. The *direct product* of $\{\mathbf{A}_k\}_{k\in K}$ is the ν-algebra $\prod_{k\in K}\mathbf{A}_k$ with universe

$$\prod_{k\in K} A_k = \left\{ \mathbf{a}\colon K \to \bigcup_{k\in K} A_k \mid \forall k \in K\,(\mathbf{a}(k) \in A_k) \right\}$$

and operations defined as follows:

$$c^{\prod_{k\in K}\mathbf{A}_k}(k) := c^{\mathbf{A}_k} \qquad f^{\prod_{k\in K}\mathbf{A}_k}(\mathbf{a}_1,\dots,\mathbf{a}_{\nu(f)})(k) = f^{\mathbf{A}_k}(\mathbf{a}_1(k),\dots,\mathbf{a}_{\nu(f)}(k))$$

for all $c \in \mathscr{F}_0$, $f \in \mathscr{F}_{\geq 1}$, and $\mathbf{a}_1,\dots,\mathbf{a}_{\nu(f)} \in \prod_{k\in K} A_k$.

For $j \in K$, the *j-th projection map* of $\prod_{k\in K}\mathbf{A}_k$ is the surjective homomorphism $p_j\colon \prod_{k\in K}\mathbf{A}_k \to \mathbf{A}_j$ defined by $\mathbf{a} \mapsto \mathbf{a}(j)$.

Lemma 1.1.11. *Let* $\prod_{k\in K}\mathbf{A}_k$ *be a direct product of* ν*-algebras. For every* ν*-algebra* $\mathbf{A}$ *and homomorphisms* $\{h_k\colon \mathbf{A} \to \mathbf{A}_k\}_{k\in K}$*, there exists a unique homomorphism* $h_!\colon \mathbf{A} \to \prod_{k\in K}\mathbf{A}_k$ *such that* $h_j = p_j \circ h_!$*, for all* $j \in K$.

1.1.2 Varieties, equations, and free algebras

For $\mathfrak{K} \subseteq \mathfrak{Alg}(\nu)$, we denote by $\mathbb{H}(\mathfrak{K})$, $\mathbb{S}(\mathfrak{K})$, and $\mathbb{P}(\mathfrak{K})$ the classes of homomorphic images, subalgebras, and direct products of members of $\mathfrak{K}$, respectively. We assume that all classes under consideration are closed under isomorphisms.

Definition 1.1.12. A *variety* is a class $\mathfrak{V} \subseteq \mathfrak{Alg}(\nu)$ that is closed under $\mathbb{H}$, $\mathbb{S}$, and $\mathbb{P}$.

If $\mathfrak{V}$ is the least variety extending a given class $\mathfrak{X}$ of ν-algebras, we say that $\mathfrak{V}$ is *generated* by $\mathfrak{X}$ and write $\mathfrak{V} = \mathbb{V}(\mathfrak{X})$. If $\mathfrak{X} = \{\mathbf{A}\}$, then we write $\mathbb{V}(\mathbf{A})$ instead of $\mathbb{V}(\{\mathbf{A}\})$. As showed by Tarski [160], $\mathbf{A} \in \mathbb{V}(\mathfrak{X})$ iff $\mathbf{A}$ is a homomorphic image of a subalgebra of a direct product of members of $\mathfrak{X}$.

Theorem 1.1.13. [160, p. 164] $\mathbb{V}(\mathfrak{X}) = \mathbb{HSP}(\mathfrak{X})$.

Definition 1.1.14. An *equation* (of type ν) over X is a pair $\langle s, t\rangle \in T_\nu(X)^2$, denoted by $s \approx t$. If ν is a type for lattice-based structures, equations of the form $s \wedge t \approx s$ and $s \vee t \approx t$ are rewritten as $s \leq t$ and referred to as *inequations*. In the same context, each equation $s \approx t$ is split into inequations $s \leq t$ and $t \leq s$.

Definition 1.1.15. A ν-algebra $\mathbf{A}$ satisfies an equation $s \approx t$ (written $\mathbf{A} \models s \approx t$) iff $h(s) = h(t)$ for all $h\colon \mathbf{T}_\nu(X) \to \mathbf{A}$ (hence if $s^{\mathbf{A}} = t^{\mathbf{A}}$).

Satisfaction of sets of equations by algebras, and of equations or sets of equations by classes of algebras, is defined in the usual way.

Definition 1.1.16. The *equational theory* of a class $\mathfrak{K} \subseteq \mathfrak{Alg}(\nu)$ is the set

$$\mathrm{Th}_e(\mathfrak{K}) := \{s \approx t \in T_\nu(X)^2 \mid \mathfrak{K} \models s \approx t\}.$$

By the equational theory of a ν-algebra $\mathbf{A}$, we mean the equational theory of $\{\mathbf{A}\}$.

The following lemma shows that the equational theories of varieties coincide with those of their generators.

Lemma 1.1.17. [116, Lem. 4.128] *For any class $\mathfrak{K}$ of ν-algebras:*

$$\mathrm{Th}_e(\mathfrak{K}) = \mathrm{Th}_e(\mathbb{H}(\mathfrak{K})) = \mathrm{Th}_e(\mathbb{S}(\mathfrak{K})) = \mathrm{Th}_e(\mathbb{P}(\mathfrak{K})) = \mathrm{Th}_e(\mathbb{V}(\mathfrak{K})).$$

Definition 1.1.18. Let $\mathfrak{K}$ be a class of ν-algebras. The *equational consequence relation* $\models_{\mathfrak{K}}$ associated with $\mathfrak{K}$ is defined as follows: for every set $\Phi \cup \{s \approx t\}$ of equations, $\Phi \models_{\mathfrak{K}} s \approx t$ if and only if, for every $\mathbf{A} \in \mathfrak{K}$ and every homomorphism $h\colon \mathbf{T}_\nu(X) \to \mathbf{A}$, if $h(s') = h(t')$ for all $s' \approx t' \in \Phi$, then $h(s) = h(t)$.

Definition 1.1.19. An *equational class* of ν-algebras is the class of all models of some set Φ of equations.

We owe to Birkhoff [24] the following fundamental result.

Theorem 1.1.20. [24, Th. 10] *Let $\mathfrak{K} \subseteq \mathfrak{Alg}(\nu)$. Then $\mathfrak{K}$ is an equational class iff $\mathfrak{K}$ is a variety.*

Definition 1.1.21. A *quasi-equation* (of type ν) over X is a definite Horn clause $\bigwedge \Phi \Rightarrow s \approx t$, with $\Phi \cup \{s \approx t\}$ a *finite* set of equations.

Definition 1.1.22. A ν-algebra $\mathbf{A}$ satisfies a quasi-equation $\bigwedge\Phi \Rightarrow s \approx t$ (written $\mathbf{A} \models \bigwedge\Phi \Rightarrow s \approx t$) iff $\Phi \vDash_{\{\mathbf{A}\}} s \approx t$. This lifts naturally to classes of ν-algebras, so we have $\mathfrak{K} \models \bigwedge\Phi \Rightarrow s \approx t$ iff $\Phi \vDash_{\mathfrak{K}} s \approx t$.

Definition 1.1.23. Let $\mathfrak{K}$ be a class of ν-algebras. We say that a ν-algebra $\mathbf{A}(G)$ has the *universal mapping property* (UMP) over G for $\mathfrak{K}$ if, for every $\mathbf{B} \in \mathfrak{K}$, any map $\phi\colon G \to B$ extends uniquely to a homomorphism $h_\phi\colon \mathbf{A}(G) \to \mathbf{B}$. In this case, $\mathbf{A}(G)$ is said to be *free* for $\mathfrak{K}$ over G.

Freeness is preserved by downward class inclusion and variety generation.

Lemma 1.1.24. [116, Lem. 4.109] *For $\mathfrak{K}, \mathfrak{K}_1, \mathfrak{K}_2 \subseteq \mathfrak{Alg}(\nu)$, the following hold:*

1. *If $\mathbf{A}(G)$ is free for $\mathfrak{K}_2$ over G and $\mathfrak{K}_1 \subseteq \mathfrak{K}_2$, then $\mathbf{A}(G)$ is free for $\mathfrak{K}_1$ over G.*
2. *If $\mathbf{A}(G)$ is free for $\mathfrak{K}$ over G, then it is free for $\mathbb{V}(\mathfrak{K})$ over G.*

Definition 1.1.25. Let us define

$$\Theta(\mathfrak{K}, X) := \bigcap\{\theta \in \mathrm{Con}(\mathbf{T}_\nu(X)) \mid \mathbf{A}/\theta \in \mathbb{S}(\mathfrak{K})\}$$

Set $\overline{X} := \{[x]_{\Theta(\mathfrak{K},X)} \mid x \in X\}$. The quotient algebra $\mathbf{F}_{\mathfrak{K}}(\overline{X}) = \mathbf{T}_\nu(X)/\Theta(\mathfrak{K}, X)$, freely generated by $\overline{X}$, is called the ($\mathfrak{K}$-)*free algebra* over $\overline{X}$.

Since the construction of $\mathbf{F}_{\mathfrak{K}}(\overline{X})$ depends on (the cardinality of) X, when no ambiguity arises we denote this structure by $\mathbf{F}_X$.

Example 1.1.26. [116, pp. 239–240] Let Σ be a countable alphabet. The set $\Sigma^* := \bigcup_{n\in\mathbb{N}} \Sigma^n$ of all finite sequences (strings) over Σ forms a monoid under string concatenation, with the empty string ε as the identity element. It is straightforward to verify that this is the free $\mathfrak{M}$-algebra over Σ, where $\mathfrak{M}$ is the variety of monoids. Take Σ as a set of variables, and let h be the homomorphism of $\mathbf{T}_{\langle 2,0\rangle}(\Sigma)$ onto $\langle X^*, \cdot, \varepsilon\rangle$ sending each term $t \in T_{\langle 2,0\rangle}(\Sigma)$ to the string over Σ obtained by deleting all operation symbols in t, while preserving order and multiplicity of variable occurrences (e.g., $h((x \cdot (y \cdot 1)) \cdot x) = xyx$; clearly $h(1) = \varepsilon$). Since h is surjective, by the First Isomorphism Theorem, $\mathbf{T}_{\langle 2,0\rangle}(\Sigma)/\ker h \cong \langle \Sigma^*, \cdot, \varepsilon\rangle$. Finally, observe that if two terms s and t differ by even a single variable, then they are not equivalent modulo $\ker h$; hence $\ker h \subseteq \Theta(\mathfrak{M}, \Sigma)$, and consequently $\ker h = \Theta(\mathfrak{M}, \Sigma)$.

Lemma 1.1.27. [116, Cors. 4.119, 4.132] *A variety $\mathfrak{V}$ contains free algebras $\mathbf{F}_X$ for any $X \neq \emptyset$. Moreover, if X has infinitely many elements, then $\mathfrak{V} = \mathbb{V}(\mathbf{F}_X)$.*

Lemma 1.1.28. [116, Thm. 4.127] *For any variety $\mathfrak{V}$, free algebras $\mathbf{F}_X \in \mathfrak{V}$ have exactly the same equational theory as $\mathfrak{V}$. That is, for all $s, t \in T_\nu(X)$,*

$$\mathfrak{V} \models s \approx t \;\Leftrightarrow\; \langle s, t\rangle \in \Theta(\mathfrak{V}, X) \;\Leftrightarrow\; s^{\mathbf{F}_X} = t^{\mathbf{F}_X} \;\Leftrightarrow\; \mathbf{F}_X \models s \approx t.$$

1.2 On (pre)ordered monoids

1.2.1 Basic concepts

In what follows, for $\mathfrak{d} \in \{l, r\}$, we write $x \cdot_\mathfrak{d} y$ to denote monoid multiplication with argument positions determined by $\mathfrak{d}$. Specifically:

$$x \cdot_\mathfrak{d} y = \begin{cases} x \cdot y & \text{if } \mathfrak{d} = l, \\ y \cdot x & \text{if } \mathfrak{d} = r \end{cases}$$

Finally, we define $\bar{l} := r$ and $\bar{r} := l$.

Definition 1.2.1. Let $\preceq$ be a preorder on (the universe of) a monoid $\mathbf{M}$. We say that $\preceq$ is *compatible* with multiplication if, for all $a, b, c \in M$ and $\mathfrak{d} \in \{l, r\}$:

$$a \preceq b \;\Rightarrow\; c \cdot_\mathfrak{d} a \preceq c \cdot_\mathfrak{d} b \tag{COMP}$$

If (COMP) holds just for $\mathfrak{d} = l$ (resp. $\mathfrak{d} = r$), then we say that $\preceq$ is *l-compatible* (resp. *r-compatible*) with multiplication.

Definition 1.2.2. A *preordered* (resp. *partially ordered*) *monoid* is a pair $\langle \mathbf{M}, \preceq\rangle$ where $\mathbf{M}$ is a monoid and $\preceq$ is a preorder (resp. partial order) satisfying (COMP). Analogously, for a fixed $\mathfrak{d} \in \{l, r\}$, we call $\langle \mathbf{M}, \preceq\rangle$ a *$\mathfrak{d}$-preordered* (resp. *partially $\mathfrak{d}$-ordered*) *monoid* if $\preceq$ is a preorder (resp. a partial order) that is $\mathfrak{d}$-compatible with multiplication.

Definition 1.2.3. An *ℓ-monoid* (short for *lattice-ordered monoid*) is a partially ordered monoid $\langle \mathbf{M}, \preceq\rangle$ where $\preceq$ is a lattice order.[2]

[2]These structures were introduced by Birkhoff [25, §XIII] under the name of *lattice-ordered semigroups*.

We denote by $\mathfrak{LM}$ the class of all ℓ-monoids. This is in fact a variety, as an ℓ-monoid may be defined as an algebra $\mathbf{M} = \langle M, \cdot, \wedge, \vee, 1\rangle$ of type $\langle 2, 2, 2, 0\rangle$ such that $\langle M, \cdot, 1\rangle$ is a monoid, $\langle M, \wedge, \vee\rangle$ is a lattice, and multiplication distributes over join on both sides, i.e., for $\mathfrak{d} \in \{l, r\}$, the following equation holds:

$$x \cdot_{\mathfrak{d}} (y \vee z) \approx (x \cdot_{\mathfrak{d}} y) \vee (x \cdot_{\mathfrak{d}} z) \tag{DMJ}$$

One readily verifies that (DMJ) implies (COMP). Recall that the $\{\wedge\}$-free reducts of ℓ-monoids form the variety $\mathfrak{IGR}$ of *idempotent semirings*, which is axiomatised by equations for monoids, join-semilattices, and the distributivity equation (DMJ). An ℓ-monoid is *bounded* if so is its lattice reduct. In this case, we consider a type expansion with constants $\bot$ and $\top$ for the bounds, and assume the axioms $\bot \leq x$ and $x \leq \top$.

In what follows, we deal with ℓ-monoids whose underlying lattices are distributive. However, we avoid the term "distributive ℓ-monoid", which is typically reserved for inverse-free reducts of ℓ-groups [45].

1.2.2 Divisibility in monoids

In monoids and, more generally, in algebras with a monoid reduct, divisibility preorders provide a natural generalisation of the classical divisibility relation on integers.

Definition 1.2.4. Let $\mathbf{M} \in \mathfrak{M}$, and let $a, b \in M$. For $\mathfrak{d} \in \{l, r\}$, define:

$$a \mid_{\mathfrak{d}} b \Leftrightarrow \exists c \in M\ (b = a \cdot_{\mathfrak{d}} c)$$

Whenever $a \mid_l b$ (resp. $a \mid_r b$), we say that b is *left-divisible* (resp. *right-divisible*) by a and call a a *left divisor* (resp. *right divisor*) of b. We call $\mid_l$ and $\mid_r$ the *left* and *right divisibility* relations on $\mathbf{M}$, respectively[3].

The following lemma collects some well-known facts about divisibility in monoids.

Lemma 1.2.5. *For a monoid $\mathbf{M} = \langle M, \cdot, 1\rangle$, the structure $\langle M, \mid_{\mathfrak{d}}\rangle$ is a preordered set with least element 1. Moreover, for all $a, b, c \in M$, the following conditions hold:*

1. *If $a \mid_{\mathfrak{d}} 1$, then a admits a right inverse when $\mathfrak{d} = l$, and a left inverse when $\mathfrak{d} = r$;*
2. *If $a \mid_{\mathfrak{d}} b$, then $a \mid_{\mathfrak{d}} b \cdot_{\mathfrak{d}} c$;*

[3]Clearly, if $\mathbf{M}$ is commutative, then $\mid_l$ and $\mid_r$ coincide.

3. *If* $a \mid_{\mathfrak{d}} b$, *then* $c \cdot_{\mathfrak{d}} a \mid_{\mathfrak{d}} c \cdot_{\mathfrak{d}} b$ *(that is,* $\mid_{\mathfrak{d}}$ *is* $\mathfrak{d}$*-compatible with multiplication).*

4. *Let* $\langle \mathbf{N}, \preceq \rangle$ *be a partially ordered monoid with least element* $1^{\mathbf{N}}$*. For every homomorphism* $h : \mathbf{M} \to \mathbf{N}$*, if* $a \mid_{\mathfrak{d}} b$*, then* $h(a) \preceq h(b)$*.*

Proof. Since multiplication is unital, $\mid_{\mathfrak{d}}$ is reflexive and satisfies $1 \mid_{\mathfrak{d}} a$ for all $a \in M$. As for transitivity, let $a, b, c \in M$ with $a \mid_{\mathfrak{d}} b$ and $b \mid_{\mathfrak{d}} c$. Then there exist $d, e \in M$ such that $b = a \cdot_{\mathfrak{d}} d$ and $c = b \cdot_{\mathfrak{d}} e$. It follows that $c = (a \cdot_{\mathfrak{d}} d) \cdot_{\mathfrak{d}} e = a \cdot_{\mathfrak{d}} (d \cdot_{\mathfrak{d}} e)$, so setting $f = d \cdot_{\mathfrak{d}} e$ we have $c = a \cdot_{\mathfrak{d}} f$, and therefore $a \mid_{\mathfrak{d}} c$. Hence, $\mid_{\mathfrak{d}}$ is a preorder.

We now proceed to prove conditions (1)–(4).

1. If $a \mid_{\mathfrak{d}} 1$, then $a \cdot_{\mathfrak{d}} b = 1$ for some $b \in M$. It follows that b is a right (resp. left) inverse of a when $\mathfrak{d} = l$ (resp. $\mathfrak{d} = r$).

2. Suppose that $a \mid_{\mathfrak{d}} b$; that is, $b = a \cdot_{\mathfrak{d}} d$ for some $d \in M$. Then, for any $c \in M$, $b \cdot_{\mathfrak{d}} c = (a \cdot_{\mathfrak{d}} d) \cdot_{\mathfrak{d}} c = a \cdot_{\mathfrak{d}} (d \cdot_{\mathfrak{d}} c)$, hence $a \mid_{\mathfrak{d}} b \cdot_{\mathfrak{d}} c$.

3. If $c \cdot_{\mathfrak{d}} a \nmid_{\mathfrak{d}} c \cdot_{\mathfrak{d}} b$, then $c \cdot_{\mathfrak{d}} b \neq (c \cdot_{\mathfrak{d}} a) \cdot_{\mathfrak{d}} d = c \cdot_{\mathfrak{d}} (a \cdot_{\mathfrak{d}} d)$ for all $d \in M$. Hence $b \neq a \cdot_{\mathfrak{d}} d$ for all $d \in M$, so $a \nmid_{\mathfrak{d}} b$.

4. The claim follows immediately from the fact that $\preceq$ satisfies (COMP) and that $1^{\mathbf{N}} \preceq r$ for all $r \in N$. Together, these conditions yield $r \preceq r \cdot_{\mathfrak{d}} s$ for all $r, s \in N$. Hence, for $a, b, c \in M$, if $b = a \cdot_{\mathfrak{d}} c$, then $h(a) \preceq h(a) \cdot_{\mathfrak{d}} h(c) = h(b)$, as required. Finally, note that $b = 1^{\mathbf{M}}$ implies $h(a) = h(b) = h(c) = 1^{\mathbf{N}}$. □

Left and right divisibility can equivalently be expressed in terms of inclusion between principal ideals. For $a \in M$ and $\mathfrak{d} \in \{l, r\}$, let $M \cdot_{\mathfrak{d}} a := \{b \cdot_{\mathfrak{d}} a \mid b \in M\}$. If $\mathfrak{d} = l$ (resp. $\mathfrak{d} = r$), then $a \cdot_{\mathfrak{d}} M$ is called the *principal left* (resp. *right*) *ideal* generated by a.

Proposition 1.2.6. *In any monoid, for* $\mathfrak{d} \in \{l, r\}$, $a \mid_{\mathfrak{d}} b$ *iff* $M \cdot_{\overline{\mathfrak{d}}} b \subseteq M \cdot_{\overline{\mathfrak{d}}} a$.

We now recall a sufficient condition under which divisibility preorders are partial orders.

Definition 1.2.7. Consider the following quasi-equations of type $\langle 2, 0 \rangle$:

$$z \cdot_{\mathfrak{d}} x \approx z \cdot_{\mathfrak{d}} y \Rightarrow x \approx y \qquad (\text{where } \mathfrak{d} \in \{l, r\}) \tag{CANC}$$

$$x \cdot y \approx 1 \Rightarrow x \approx 1 \tag{CON$_1$}$$

$$x \cdot y \approx 1 \Rightarrow y \approx 1 \tag{CON$_2$}$$

A monoid is said to be *cancellative* [42] if it satisfies (CANC), and *conical* [170] if it satisfies both (CON_1) and (CON_2).

Observe that cancellativity allows to reduce equations in analogy with the simplifications permitted by inverses in groups[4]. Conicality, in turn, expresses the non-existence of non-trivial invertible elements in a monoid. In other words, a monoid is conical when the only element admitting both a right and a left inverse is the unit (which therefore coincides with its inverses). The following examples show that cancellativity and conicality are independent notions.

Example 1.2.8. Free monoids, as well as $\langle \mathbb{N}, +, 0 \rangle$, are both cancellative and conical. A monoid with an absorbing element cannot be cancellative, but it may still be conical (the most obvious example is $\langle \mathbb{N}, \cdot, 1 \rangle$). A well-known class of conical non-cancellative monoids is the variety of join-semilattices with zero (dually, meet-semilattices with unit). On the other hand, a noteworthy example of a cancellative but non-conical monoid is $\langle \{0,1\}, \oplus, 0 \rangle$, where $\oplus$ is the XOR operator. This monoid is cancellative, since $c \oplus a$ never coincides with $c \oplus b$ when $a \neq b$[5]. Moreover, $1 \oplus 1 = 0$ although $1 \neq 0$, so conicality does not hold. Finally, the monoid of transformations $f \colon A \to A$ on a set A is neither cancellative nor conical. Indeed, $h \circ f = h \circ g$ does not generally imply $f = g$, and the identity map id_A is not the unique invertible element, as every bijection is invertible too.

Proposition 1.2.9. *If a monoid is both cancellative and conical, then its divisibility preorders are antisymmetric.*

Proof. Let now $\mathbf{M}$ be a cancellative, conical monoid. For $\mathfrak{d} \in \{l, r\}$ and $a, b \in M$, suppose that $a \mid_{\mathfrak{d}} b$ and $b \mid_{\mathfrak{d}} a$. Then there exist $c, d \in M$ such that $b = a \cdot_{\mathfrak{d}} c$ and $a = b \cdot_{\mathfrak{d}} d$. Substituting $a \cdot_{\mathfrak{d}} c$ for b in the second expression gives $a = (a \cdot_{\mathfrak{d}} c) \cdot_{\mathfrak{d}} d = a \cdot_{\mathfrak{d}} (c \cdot_{\mathfrak{d}} d)$. As $a = a \cdot_{\mathfrak{d}} 1$, by cancellativity it follows that $c \cdot_{\mathfrak{d}} d = 1$. Since $\mathbf{M}$ is conical, this implies $c = d = 1$. Hence $a = b \cdot_{\mathfrak{d}} 1 = b$. □

[4] In fact, any group trivially satisfies (CANC). For this reason, cancellativity plays a key role in the study of the embeddability of semigroups into groups. In particular, it is a necessary and sufficient condition for embedding a commutative semigroup into a group, whereas in the non-commutative case it is only necessary. For details, see [42, §1.10] and [43, §12].

[5] Note that $\langle \{0,1\}, \oplus, 0 \rangle$ is the inverse-free reduct of the Abelian group $\langle \{0,1\}, \oplus, ', 0 \rangle$, where $'$ is defined by $x' \approx x$ (i.e., each element is its own inverse).

In the following chapters, we shall focus on divisibility in free monoids. In this setting, it is straightforward to observe that left (resp. right) divisibility corresponds to the *prefix* (resp. *suffix*) *relation* on strings. The following result is a corollary of the preceding discussion.

Proposition 1.2.10. *Let* $\mathbf{F}_\Sigma$ *be a free monoid. Then, for* $\mathfrak{d} \in \{l, r\}$*, the structure* $\langle \mathbf{F}_\Sigma, |_\mathfrak{d} \rangle$ *is a* $\mathfrak{d}$*-partially ordered monoid with least element* ε*.*

1.3 Unary residuation

Definition 1.3.1. Let $\mathbf{A} = \langle A, \preceq^{\mathbf{A}} \rangle$ and $\mathbf{B} = \langle B, \preceq^{\mathbf{B}} \rangle$ be partially ordered sets (posets). A map $f\colon A \to B$ is *residuated* if there exists a map $g\colon B \to A$ such that

$$f(a) \preceq^{\mathbf{B}} b \Leftrightarrow a \preceq^{\mathbf{A}} g(b), \tag{1RES}$$

for all $a \in A$ and all $b \in B$. When (1RES) holds, we say that g is the (*right*) *residual* of f, and we call $\langle f, g \rangle$ a *residuated pair*.

We refer to (1RES) as the *unary residuation law*. The following result provides a necessary and sufficient condition for (1RES).

Proposition 1.3.2. [73, Lem. 3.2] *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be posets, and let* $f\colon A \to B$ *and* $g\colon B \to A$ *be maps. Then* $\langle f, g \rangle$ *is a residuated pair if and only if:*

1. *both* f *and* g *are order-preserving;*
2. $\mathrm{id}_{\mathbf{A}} \preceq^{\mathbf{A}} g \circ f$ *and* $f \circ g \preceq^{\mathbf{B}} \mathrm{id}_{\mathbf{B}}$*, where (with a slight abuse of notation)* $\preceq^{\mathbf{A}}$ *and* $\preceq^{\mathbf{A}}$ *denote the pointwise orders on the sets of self-maps on* P *and* Q*, respectively.*

We now recall some basic facts about residuated pairs (for details, see [73, Lems. 3.1 and 3.3] and [121, Lem. 1.2.8]).

Proposition 1.3.3. *Let* $\mathbf{A}$*,* $\mathbf{B}$ *be posets, and let* $\langle f, g \rangle$ *be a residuated pair with* $f\colon A \to B$ *and* $g\colon B \to A$*. The following conditions hold:*

1. *for* $a \in A$*,* $b \in B$*,* $f(a) = \min\{b \mid a \preceq^{\mathbf{A}} g(b)\}$ *and* $g(b) = \max\{a \mid f(a) \preceq^{\mathbf{B}} b\}$*;*
2. $g \circ f$ *and* $f \circ g$ *are, respectively, a closure and an interior operator;*

3. $f \circ g \circ f = f$ *and* $g \circ f \circ g = g$.

4. *If* $\min \mathbf{A}$ *(resp.* $\max \mathbf{B}$*) exists, then* $\min \mathbf{B}$ *(resp.* $\max \mathbf{A}$*) exists as well, and* $f(\min \mathbf{A}) = \min \mathbf{B}$ *(resp.* $\max \mathbf{A} = g(\max \mathbf{B})$*).*

Unary residuation is naturally realised by certain pairs of operators defined from binary relations (see again, for example, [73, pp. 143–144] or [121, Ex. 1.2.2]). We recall this construction in the following example, introducing a notation that will be adopted in the following chapters.

Example 1.3.4. Let A and B be sets, and let $R \subseteq A \times B$. The direct image operator $\diamondsuit\!\!\!\!-_R$ and the inverse image operator $\diamondsuit_R$ associated with R are defined as follows:

$$\begin{array}{ll} \diamondsuit\!\!\!\!-_R \colon \wp(A) \to \wp(B) & \diamondsuit_R \colon \wp(B) \to \wp(A) \\ \quad C \mapsto \{b \in B \mid \exists c(cRb \curlywedge c \in C)\} & \quad D \mapsto \{a \in A \mid \exists d(aRd \curlywedge d \in D)\} \end{array}$$

where $C \subseteq A$ and $D \subseteq B$. Note that $\diamondsuit_R$ can equivalently be defined as the direct image operator associated with the converse relation R^{-1} (i.e. $\diamondsuit_R = \diamondsuit\!\!\!\!-_{R^{-1}}$). Both $\diamondsuit\!\!\!\!-_R$ and $\diamondsuit_R$ are residuated w.r.t. the inclusion order $\subseteq$, with residuals respectively defined by:

$$\begin{array}{ll} \Box_R \colon \wp(B) \to \wp(A) & \boxminus_R \colon \wp(A) \to \wp(B) \\ \quad D \mapsto \{a \in A \mid \forall d(aRd \Rightarrow d \in D)\} & \quad C \mapsto \{b \in B \mid \forall c(cRb \Rightarrow c \in C)\} \end{array}$$

In the preceding example, when $A = B$, the pair $\langle A, R\rangle$ may be regarded as a Kripke frame for some modal logic, and the operator definitions give rise to the semantic clauses for the usual connectives $\diamondsuit, \Box$ and their "backward" counterparts $\diamondsuit\!\!\!\!-, \boxminus$[6]. Clearly, if R is symmetric (as in the case of the modal logic $\mathsf{S5}$), then $\diamondsuit = \diamondsuit\!\!\!\!-$ and $\Box = \boxminus$.

In the present dissertation, we consider residuated pairs of the form $\langle \diamondsuit, \boxminus \rangle$ defined from partial orders.

Definition 1.3.5. Let $\mathbf{A}$ be a poset. A subset $D \subseteq A$ is a *downset* in $\mathbf{A}$ if $a \in D$ and $b \preceq a$ imply $b \in D$, for all $a, b \in A$. The *principal downset* generated by an element $a \in A$ is the set $\downarrow a := \{b \in A \mid b \preceq a\}$. For $B \subseteq A$, we define $\downarrow B := \bigcup_{b \in B} \downarrow b$. We call $\downarrow B$ the downset generated by B.

[6]For instance, if R is interpreted as a temporal precedence relation, the resulting structure yields a frame for Prior's Tense Logic [144], with $\diamondsuit, \Box, \diamondsuit\!\!\!\!-$, and $\boxminus$ corresponding, respectively, to the tense modalities F, G, P, and H.

It is immediate to to check that $\downarrow$ defines a closure operator on $\langle \wp(A), \subseteq \rangle$. Moreover, observe that $\bigcup_{b \in B} \downarrow b = \{a \in A \mid \exists b (a \preceq b \curlywedge b \in B)\}$. Hence $\downarrow B = \Diamond_{\preceq} B$. By (1RES), Prop. 1.3.3 (1), and the axioms for closure operators, it follows that $\boxdot_{\preceq}$ is an interior operator defined by $C \mapsto \bigcup \{B \mid \Diamond_{\preceq} B \subseteq C\}$, for any $C \subseteq A$.

Chapter 2

Modelling framework for concurrent systems

2.1 LTSs and their trace-based specification

The mathematical representation of concurrency can be given within a variety of modelling paradigms [154, 171]. We place ourselves within the framework of interleaving concurrency and take labelled transition systems (LTSs) [98] as our reference model of computation. This is a natural choice, as LTSs provide the operational semantics for process calculi like CCS [122] or CSP [93]; however, the following discussion can be readily adapted to automata, Kripke structures, and similar models.

Definition 2.1.1. A *labelled transition system* (*LTS*) is a triple $\mathcal{T} = \langle S, Act, T \rangle$ where:

- S is a nonempty set of *states*;
- Act is a nonempty set of *actions*;
- $T \subseteq S \times Act \times S$ is a *transition relation*.

A *rooted LTS* is a quadruple $\mathcal{T} = \langle S, Act, T, s_0 \rangle$ where $\langle S, Act, T \rangle$ is an LTS and $s_0 \in S$ is a distinguished state called the *initial state*.

All LTSs considered in this dissertation are assumed to be rooted, so we henceforth write "LTS" for "rooted LTS".

Definition 2.1.2. Given an LTS $\mathcal{T} = \langle S, Act, T, s_0 \rangle$, we refer to elements of T as *transitions* and, for $s, s' \in S$ and $\mathtt{a} \in Act$, we write $s \xrightarrow{\mathtt{a}} s'$ if $\langle s, \mathtt{a}, s' \rangle \in T$. A *path* in $\mathcal{T}$ is a (possibly infinite) sequence of transitions such that whenever $s_i \xrightarrow{\mathtt{a}_i} s_j$ and $s_k \xrightarrow{\mathtt{a}_j} s_h$ occur consecutively, we have $s_j = s_k$. We denote paths using the streamlined notation $\cdots s_i \xrightarrow{\mathtt{a}_i} s_j \xrightarrow{\mathtt{a}_j} s_k \cdots$. A path is *rooted* if its first transition starts from s_0.

Running example. We consider an LTS, which we call $\mathcal{E}$, modelling a simplified request-response protocol (the system architecture is displayed in Figure 2.1). At the initial state, the action `conn` establishes a connection between client and server. The client then transmits a request via `snd`. At this stage, the system evolves non-deterministically depending on the server's response: if the request is correctly received, the server sends an acknowledgement and the requested data (action `ack`, from s_2 to s_4); otherwise, the server signals a transmission problem via a negative acknowledgment (action `nack`, from s_2 to s_3) and the protocol resets (action `end`, from s_3 to s_0). In s_4, the client may either terminate the protocol by executing `end` and returning to s_0 or initiate another request (action `req`, from s_4 to s_1).

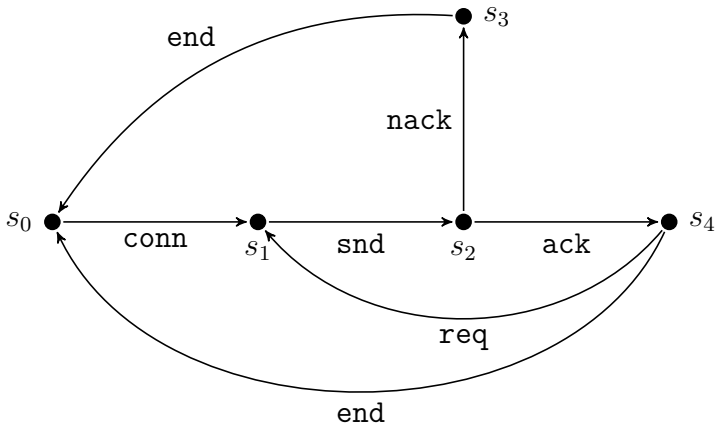


Figure 2.1: Request-response protocol.

Paths in LTSs provide an immediate representation of the behaviour of the systems being modelled. However, to formalise (temporal) properties of concurrent processes and to study their structure, we require an additional abstraction, which is provided by the notion of an *execution trace*. As in the following chapters we do not address any specific class of concurrent systems, for the general definitions in the present subsection

we adopt the perspective of "mixed" executions, following [149, 136]. In particular, by slightly adapting the modelling framework proposed in [41, §7.2.2], we represent traces as finite or infinite sequences of state/action pairs, and classify them as valid or invalid according to a formal notion of compatibility with the transition relation.

Definition 2.1.3. For $\mathcal{T} = \langle S, Act, T, s_0 \rangle$ an LTS, let $\Sigma_{\mathcal{T}} := S \times Act$. A *$\mathcal{T}$-trace* is any sequence of elements from $\Sigma_{\mathcal{T}}$, possibly empty or infinite.

For $\langle s, \mathtt{a} \rangle \in \Sigma_{\mathcal{T}}$, we define projections $proj_1 \colon \langle s, \mathtt{a} \rangle \mapsto s$ and $proj_2 \colon \langle s, \mathtt{a} \rangle \mapsto \mathtt{a}$. Treating $\Sigma_{\mathcal{T}}$ as an alphabet, finite $\mathcal{T}$-traces correspond to words in $\Sigma_{\mathcal{T}}^*$, and infinite ones to ω-words in $\Sigma_{\mathcal{T}}^\omega$. Whenever it is necessary to distinguish finite from infinite traces, we use $w, u, v, \ldots$ (possibly with annotations) to denote elements of $\Sigma_{\mathcal{T}}^*$, and $\alpha, \beta, \gamma, \ldots$ to denote elements of $\Sigma_{\mathcal{T}}^\omega$. For mixed traces, we adopt the same notation as in the finite case. If w is a finite trace, we denote by $|w|$ its length. Let $\Sigma_{\mathcal{T}}^\infty := \Sigma_{\mathcal{T}}^* \cup \Sigma_{\mathcal{T}}^\omega$. We say that $v \in \Sigma_{\mathcal{T}}^*$ is a *prefix* of $w \in \Sigma_{\mathcal{T}}^\infty$ (written $v \sqsubseteq w$) if there exists $u \in \Sigma_{\mathcal{T}}^\infty$ such that $w = vu$. Finally, we define $pref(w) := \{v \in \Sigma_{\mathcal{T}}^* \mid v \sqsubseteq w\}$. For $w \in \Sigma_{\mathcal{T}}^\infty$ and $i \in \mathbb{N}$, $w[i]$ and $w[\triangleright i]$ respectively denote the i-th pair occurring in w and the prefix $w[0], \ldots, w[i]$.

Definition 2.1.4. Let $\mathcal{T}$ be an LTS. A finite $\mathcal{T}$-trace $w \in \Sigma_{\mathcal{T}}^*$ is *valid* iff:

$$\text{for all } i < |w| - 1,\ proj_1(w[i]) \xrightarrow{proj_2(w[i])} proj_1(w[i+1]) \tag{VAL$_*$}$$

Similarly, an infinite $\mathcal{T}$-trace $\alpha \in \Sigma_{\mathcal{T}}^\omega$ is *valid* if and only if:

$$\text{for all } i \in \mathbb{N},\ proj_1(\alpha[i]) \xrightarrow{proj_2(\alpha[i])} proj_1(\alpha[i+1]) \tag{VAL$_\omega$}$$

A $\mathcal{T}$-trace $w \in \Sigma_{\mathcal{T}}^\infty$ is *invalid* if it is not valid.

Definitions 2.1.3 and 2.1.4 show that the notion of a trace captures and extends that of a path, as each valid trace corresponds to a unique path. From the perspective of formal verification, the distinction between valid and invalid traces may appear redundant, since only traces corresponding to paths are relevant in that context. However, as we will see, it becomes useful when one aims to provide an algebraic characterisation of properties of finite traces as subsets of a free monoid. This in no way affects verification: determining whether a valid trace satisfies a property reduces to checking whether it belongs to a set of strings with certain features, regardless of whether the set contains only valid traces or also includes invalid ones.

Running example. Below are examples of valid $\mathcal{E}$-traces with their associated paths.

Class	Example $\mathcal{E}$-trace	Path
$\Sigma_{\mathcal{E}}^*$	$\langle s_0, \texttt{conn}\rangle\langle s_1, \texttt{snd}\rangle\langle s_2, \texttt{ack}\rangle\langle s_4, \texttt{end}\rangle$	$s_0 \xrightarrow{\texttt{conn}} s_1 \xrightarrow{\texttt{snd}} s_2 \xrightarrow{\texttt{ack}} s_4$
	$(\langle s_1, \texttt{snd}\rangle\langle s_2, \texttt{nack}\rangle\langle s_3, \texttt{end}\rangle\langle s_0, \texttt{conn}\rangle)^3$	$\pi \xrightarrow{\texttt{conn}} \pi \xrightarrow{\texttt{conn}} \pi$, where $\pi = s_1 \xrightarrow{\texttt{snd}} s_2 \xrightarrow{\texttt{nack}} s_3 \xrightarrow{\texttt{end}} s_0$
$\Sigma_{\mathcal{E}}^\omega$	$(\langle s_0, \texttt{conn}\rangle\langle s_1, \texttt{snd}\rangle\langle s_2, \texttt{nack}\rangle\langle s_3, \texttt{end}\rangle)^\omega$	$s_0 \xrightarrow{\texttt{conn}} s_1 \xrightarrow{\texttt{snd}} s_2 \xrightarrow{\texttt{nack}} s_3 \xrightarrow{\texttt{end}} s_0 \cdots$
	$(\langle s_1, \texttt{snd}\rangle\langle s_2, \texttt{ack}\rangle\langle s_4, \texttt{req}\rangle)^\omega$	$s_1 \xrightarrow{\texttt{snd}} s_2 \xrightarrow{\texttt{ack}} s_4 \xrightarrow{\texttt{req}} s_1 \cdots$

The following are instead examples of invalid $\mathcal{E}$-traces:

any element of $\Sigma_{\mathcal{E}}$ $\qquad$ $\langle s_1, \texttt{snd}\rangle(\langle s_2, \texttt{ack}\rangle\langle s_2, \texttt{nack}\rangle)^*$ $\qquad$ $\langle s_0, \texttt{conn}\rangle^\omega$

2.2 Trace properties for LTSs

We now discuss trace properties for LTSs, beginning with the general case of mixed executions and then turning to finite-trace models.

2.2.1 The mixed case

We now discuss trace properties for LTSs, beginning with the general case of mixed executions and then turning to finite-trace models.

Definition 2.2.1. Let $\mathcal{T}$ be an LTS. A *trace property* for $\mathcal{T}$ is any set $P \subseteq \Sigma_{\mathcal{T}}^\infty$. A $\mathcal{T}$-trace w *satisfies* a property P if $w \in P$.

Remark 2.2.2. Note that an LTS $\mathcal{T}$ may be identified with a special trace property, namely the set $V_{\mathcal{T}}$ of its valid traces. In our setting, let $FV_{\mathcal{T}} := \{w \in \Sigma_{\mathcal{T}}^* \mid w \text{ satisfies } (\text{VAL}_*)\}$ and $IV_{\mathcal{T}} := \{w \in \Sigma_{\mathcal{T}}^\omega \mid w \text{ satisfies } (\text{VAL}_\omega)\}$. Then $V_{\mathcal{T}} := FV_{\mathcal{T}} \cup IV_{\mathcal{T}}$.

Definition 2.2.3. For $\mathcal{T}$ an LST, let $P \subseteq \Sigma_{\mathcal{T}}^\infty$. Consider the following conditions:

$$\forall w \in \Sigma_{\mathcal{T}}^\infty \ (w \in P \Leftrightarrow \mathit{pref}(w) \subseteq P) \tag{SP$_\infty$}$$

$$\forall w \in \Sigma_{\mathcal{T}}^* \ \exists w' \in P \ (w \in \mathit{pref}(w')) \tag{LP$_\infty$}$$

Then P is a *safety property* [149] if it satisfies (SP$_\infty$) and is a *liveness property* [136] if it satisfies (LP$_\infty$).

Remark 2.2.4. $\emptyset$ is a safety property, while $\Sigma_{\mathcal{T}}^*$ and $\Sigma_{\mathcal{T}}^\infty$ are both safety and liveness properties.

The computational interpretation of conditions (SP_∞) and (LP_∞) is well known [10]. A safety property S is satisfied by a trace w if S is preserved in each prefix of w. If w is valid, this amounts to saying that no event violating S occurs during the computation; if it did, it would be observable in some finite subcomputation—possibly in the last transition of the path associated with w, when w is finite. In turn, an execution satisfies a liveness property L if some event witnessing L eventually occurs during the computation—possibly in the limit, in the case of infinite behaviours. Formally, any finite trace $w \in \Sigma_{\mathcal{T}}^*$ admits an extension, finite or infinite, to a trace $w' \in \Sigma_{\mathcal{T}}^\infty$ that satisfies L.

Remark 2.2.5. A classical theorem by Alpern and Schneider [10, Thm. 1] states that in infinite-trace systems, every property is the intersection of a safety and a liveness property. This is established via a topological argument, as in that context safety and liveness correspond, respectively, to closure and density in the Plotkin topology of observable properties[7]. Pasqua and Mastroeni [136, Prop. 3] generalised this result to mixed traces, showing that (SP_∞) and (LP_∞) define the closed and dense sets in the topology on $\Sigma_{\mathcal{T}}^\infty$ induced by the closure operator $P \mapsto \lim(\bigcup_{w \in P} \mathit{pref}(w))$, where lim is the Eilenberg-limit operator [60].

Running example. Consider the following system policies for $\mathcal{E}$:

1. The client never receives a message from the server before having sent a request.

2. Whenever the client forwards a request to the server, it receives a response.

Following the terminology of Manna and Pnueli [115], policies (1) and (2) are instances of the *causal dependence* and *responsiveness* property patterns, respectively.

Policy (1) prescribes that neither state s_3 nor state s_4 can be reached before the system has entered state s_2. In other words, $\mathcal{E}$ cannot enable actions `ack` or `nack` without having previously enabled action `snd`. Abstractly, this policy corresponds to the set P_1 of $w \in \Sigma_{\mathcal{E}}^\infty$ such that, for every substring $w[i-1]w[i]$ with $\mathit{proj}_2(w[i-1]) \in \{\texttt{ack}, \texttt{nack}\}$, there exists $j < i$ with $\mathit{proj}_2(w[j-1]) = \texttt{snd}$. Observe that, for any $w \in P_1$ and $v \sqsubseteq w$, the prefix v also satisfies P_1. If v contains a pair of the form $\langle s, \texttt{ack} \rangle$

[7] For further details, see Smyth [157].

or $\langle s, \texttt{nack}\rangle$, it must be preceded by some pair $\langle s', \texttt{snd}\rangle$. If v contains no $\langle s, \texttt{ack}\rangle$ or $\langle s, \texttt{nack}\rangle$ pair, then the property holds vacuously. Conversely, let $w \in \Sigma_{\mathcal{T}}^{\infty}$ be such that $\mathit{pref}(w) \subseteq P_1$. If w is finite, then trivially $w \in P_1$; if w is infinite, suppose for a contradiction that $w \notin P_1$. Then there exists $i \in \mathbb{N}$ such that $w[\triangleright i] \notin P_1$, but this is impossible since all prefixes of w are in P_1. Therefore, P_1 is a safety property and can be formalised both in classical LTL [139] and in its finite-trace variant LTL_f [50] by means of the following formula [156] (where U is the *until* operator):

$$\neg(\neg \mathit{request}\ U\ (\mathit{response} \wedge \neg \mathit{request}))$$

Policy (2) states that once the system reaches s_2, it is guaranteed to subsequently reach either state s_3 or state s_4. Therefore, enabling `snd` implies the eventual execution of `ack` or `nack`. We can thus define the abstract property corresponding to this policy as the set P_2 of $\mathcal{E}$-traces $w \in \Sigma_{\mathcal{E}}^{\infty}$ such that, for every substring $w[i-1]w[i]$ with $\mathit{proj}_2(w[i-1]) = \texttt{snd}$, there exists $j > i$ with $\mathit{proj}_2(w[j-1]) \in \{\texttt{ack}, \texttt{nack}\}$. P_2 is a liveness property: any $v \in \Sigma_{\mathcal{E}}^{*}$ is indeed a prefix of some $w \in P_2$. The formalisation of P_2 in LTL or LTL_f is given by the following formula (where G and F are the *globally* and *eventually* operators, respectively):

$$G(\mathit{request} \rightarrow F(\mathit{response}))$$

Observe that all valid $\mathcal{E}$-traces satisfy both P_1 and P_2, that is, $V_{\mathcal{E}} \subseteq P_1$ and $V_{\mathcal{E}} \subseteq P_2$.

2.2.2 The finite case

We proceed to examine f-properties. In this case, for an LTS $\mathcal{T}$, our domain of reference is simply $\wp(\Sigma_{\mathcal{T}}^{*})$, while $\mathcal{T}$ itself may be specified as the f-property $FV_{\mathcal{T}}$ defined in Remark 2.2.2. As shown in Proposition 1.2.10, the prefix relation $\sqsubseteq$ is the left divisibility order on $\Sigma_{\mathcal{T}}^{*}$; hence, for $w \in \Sigma_{\mathcal{T}}^{*}$, $\mathit{pref}(w)$ is the principal downset $\downarrow w$ in $\langle \Sigma_{\mathcal{T}}^{*}, \sqsubseteq\rangle$. The inverse image operator $\Diamond = \Diamond_{\sqsubseteq}$ on $\wp(\Sigma_{\mathcal{T}}^{*})$ defined by

$$\Diamond P := \bigcup_{w \in P} \mathit{pref}(w) = \{v \in \Sigma_{\mathcal{T}}^{*} \mid \exists w (v \sqsubseteq w \curlywedge w \in P)\}$$

is called the *prefix-closure operator*. Since $\sqsubseteq$ is a partial order, $\Diamond$ is a topological closure operator on $\langle \wp(\Sigma_{\mathcal{T}}^{*}), \subseteq\rangle$, i.e. for $P, Q \subseteq \Sigma_{\mathcal{T}}^{*}$, it satisfies the Kuratowski axioms:

(D_1) $P \subseteq \Diamond P$;

(D_2) $\Diamond\Diamond P = \Diamond P$;

(D_3) $\Diamond(P \cup Q) = \Diamond P \cup \Diamond Q$;

(D_4) $\Diamond \emptyset = \emptyset$

Roşu [149] observed that the notion of safety for f-properties coincides precisely with being a fixpoint of $\Diamond$, while Pasqua and Mastroeni [136] showed that restricting (LP_∞) to finite traces characterises liveness f-properties as those whose image under $\Diamond$ is the entire $\Sigma^*_{\mathcal{T}}$. We can thus restate Definition 2.2.3 as follows:

Definition 2.2.6. For $\mathcal{T}$ an LST, let $P \subseteq \Sigma^*_{\mathcal{T}}$. Consider the following conditions:

$$\Diamond P = P \tag{SP_*}$$

$$\Diamond P = \Sigma^*_{\mathcal{T}} \tag{LP_*}$$

Then P is a *safety f-property* [149] if it satisfies (SP_*) and is a *liveness f-property* [136] if it satisfies (LP_*).

One easily checks that restricting (SP_∞) to $\Sigma^*_{\mathcal{T}}$ yields a condition equivalent to (SP_*).

Proposition 2.2.7. *For all $P \subseteq \Sigma^*_{\mathcal{T}}$, $\Diamond P = P$ iff $P = \{w \in \Sigma^*_{\mathcal{T}} \mid pref(w) \subseteq P\}$.*

Proof. Suppose that P is a fixpoint of $\Diamond$. Since $\Diamond P = \bigcup_{w \in P} pref(w)$, it follows that $pref(v) \subseteq \Diamond P = P$ for all $v \in P$. This establishes the left-to-right implication. For the converse, assume *ad absurdum* that $P \neq \Diamond P$ and that $P = \{w \in \Sigma^*_{\mathcal{T}} \mid pref(w) \subseteq P\}$. Since $\Diamond$ is a closure operator, we then have $P \subsetneq \Diamond P$, hence there exists $v \in \Diamond P$ such that $v \notin P$. But then, for some $u \in P$, we must have $v \in pref(u)$, and as $pref(u) \subseteq P$, it follows that $v \in P$—a contradiction. □

Running example. The properties P_1 and P_2 introduced in Subsection 2.2.1 for mixed traces can be restricted to the finite case by defining $P_{1,f} := P_1 \cap \Sigma^*_{\mathcal{E}}$ and $P_{2,f} := P_2 \cap \Sigma^*_{\mathcal{E}}$. In this setting, the temporal logic specification naturally reduces to LTL_f alone.

Remark 2.2.8. Pasqua and Mastroeni [136, §3.2] define liveness f-properties as follows:

$$L \subseteq \Sigma^*_{\mathcal{T}} \text{ is liveness} \quad \Leftrightarrow \quad \forall w \in \Sigma^*_{\mathcal{T}}\, \exists w' \in L\ (w \sqsubseteq w').$$

Observe that the existentially quantified subformula is precisely the requirement for w to belong to $\Diamond L$. Hence we can rewrite:

$$L \subseteq \Sigma^*_{\mathcal{T}} \text{ is liveness} \quad \Leftrightarrow \quad \forall w \in \Sigma^*_{\mathcal{T}} (w \in \Diamond L).$$

Therefore L is a liveness f-property if and only if $\Sigma^*_{\mathcal{T}} \subseteq \Diamond L$, and thus $\Sigma^*_{\mathcal{T}} = \Diamond L$.

The following result is an immediate consequence of the above discussion.

Proposition 2.2.9. [136, Prop. 1] *Safety and liveness f-properties are, respectively, the closed and dense sets in the topology over* $\Sigma_{\mathcal{T}}^*$ *induced by* $\Diamond$.

Chapter 3

Algebraic theory of finite-trace properties

In this chapter, we develop an algebraic framework for representing f-properties, with a focus on those specified via the prefix-closure operator $\Diamond$ and its residual $\boxslash = \boxslash_{\sqsubseteq}$. As noted in the Introduction, for a free monoid $\mathbf{F}_\Sigma = \langle \Sigma^*, \cdot, \varepsilon \rangle$, several algebras can be defined on $\wp(\Sigma^*)$. In what follows, we adopt a "minimal" approach, taking as our base structure the bounded ℓ-monoid $\wp(\mathbf{F}_\Sigma) = \langle \wp(\Sigma^*), \cap, \cup, \cdot, \{\varepsilon\}, \emptyset, \Sigma^* \rangle$, where multiplication in the monoid reduct $\langle \wp(\Sigma^*), \cdot, \{\varepsilon\} \rangle$ is given by the standard language concatenation defined as $P \cdot Q := \{vw \mid v \in P \curlywedge w \in Q\}$ for $P, Q \subseteq \Sigma^*$.

Let $\wp(\mathbf{F}_\Sigma)_+$ be the algebra obtained by expanding $\wp(\mathbf{F}_\Sigma)$ with the residuated pair $\langle \Diamond, \boxslash \rangle$. The algebraic model we are going to introduce, which we refer to as a *closure ℓ-monoid*, arises as an abstraction of $\wp(\mathbf{F}_\Sigma)_+$ (see Lemma 3.3.5). The reasons for this are twofold. First, to the best of our knowledge, there is no complete axiomatisation of $\wp(\mathbf{F}_\Sigma)$, nor is it known whether such an axiomatisation could be finitary. Second, as we shall see, the proof-theoretic framework adopted in Chapter 4 is too restrictive to permit the derivation of (the sequent counterparts of) some conditions holding in $\wp(\mathbf{F}_\Sigma)_+$.

3.1 Prefix-closure

We begin by examining how $\Diamond$ interacts with the operations of $\wp(\mathbf{F}_\Sigma)$. In the case of lattice operations, $\Diamond$ behaves as a standard $\mathsf{S4}$ diamond. Hence, using conditions

(D_1)–(D_4) in Subsection 2.2.2, one readily shows, for instance, that

$$P \subseteq Q \Rightarrow \Diamond P \subseteq \Diamond Q \quad \Diamond(P \cap Q) \subseteq \Diamond P \cap \Diamond Q \quad \Diamond(P \cap Q) = \Diamond(\Diamond P \cap \Diamond Q)$$

for all $P, Q \subseteq \Sigma^*$. As for constants, we already know that $\Diamond$ preserves $\emptyset$. Of course, $\Diamond\Sigma^* = \Sigma^*$, and since $\varepsilon = \min\langle \Sigma^*, \sqsubseteq \rangle$, we also have $\Diamond\{\varepsilon\} = \{\varepsilon\}$. What is less immediate, however, is how $\Diamond$ behaves with respect to language concatenation. First, observe that for $P, Q \subseteq \Sigma^*$, a word w belongs to $\Diamond(P \cdot Q)$ if there exists $uv \in P \cdot Q$ (with $u \in P$ and $v \in Q$) such that exactly one of the cases in the following table applies.

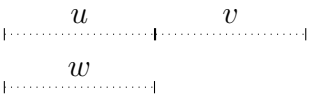
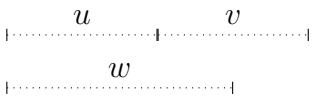
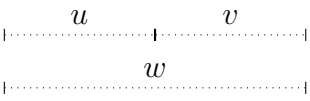
Case 1. $w \sqsubseteq u$, $w \neq u$	Case 2. $w = u$
	
Case 3. $w = uv'$, $v' \sqsubseteq v$, $v' \neq v$, $v' \neq \varepsilon$	Case 4. $w = uv$
	

Table 3.1: The structure of elements of $\Diamond(P \cdot Q)$.

From the cases in Table 3.1, it follows immediately that the following property holds in $\langle \mathbf{F}_\Sigma, \sqsubseteq \rangle$, for all $u, v, w \in \Sigma^*$:

$$w \sqsubseteq uv \text{ implies } w = u'v', \text{ with } u' \sqsubseteq u \text{ and } v' \sqsubseteq v \qquad (\text{RDP}_*)$$

Readers having some familiarity with the theory of ℓ-groups will have recognised in (RDP_*) a particular instance of the well-known *Riesz Decomposition Property* (*RDP*)[8]. It is a standard fact that the RDP is satisfied by some structures with weaker requirements than ℓ-groups (e.g. Riesz groups [66]). We show that any monoid endowed with a divisibility preorder has the RDP. In the following lemma, for a monoid $\mathbf{M}$, $\mathfrak{d} \in \{l, r\}$, and elements $a_1, \ldots, a_n \in M$, we denote by $[a_1 \cdots a_n]_\mathfrak{d}$ the product $a_1 \cdots a_n$ if $\mathfrak{d} = l$, and $a_n \cdots a_1$ if $\mathfrak{d} = r$.

[8]Originally introduced in the context of functional analysis [148], specifically in connection with vector lattices, the RDP is defined for ℓ-groups as follows [12, p. 3]. Let $\langle \mathbf{G}, \preceq \rangle$ be an (additive) ℓ-group. Define $G_+ := \{a \in G \mid 0 \prec a\}$ (this set is called the *positive cone* of $\langle \mathbf{G}, \preceq \rangle$). Let $a_1, \ldots, a_n \in G_+$. Then $\langle \mathbf{G}, \preceq \rangle$ has the RDP if $0 \preceq b \preceq \sum_{1 \leq i \leq n} a_i$ implies $b = \sum_{1 \leq i \leq n} b_i$ with $0 \preceq b_i \preceq a_i$, for all $i \in \{1, \ldots, n\}$.

Lemma 3.1.1. *Let* $\mathbf{M}$ *be a monoid. Then, for* $\mathfrak{d} \in \{l, r\}$, $\langle \mathbf{M}, |_\mathfrak{d} \rangle$ *has the RDP.*

Proof. We prove by induction on $n \in \mathbb{Z}_+$ that, for all $a_1, \ldots, a_n, b \in M$:

$$b \mid_\mathfrak{d} [a_1 \cdots a_n]_\mathfrak{d} \text{ implies } b = [b_1 \cdots b_n]_\mathfrak{d}, \text{ with } b_i \mid_\mathfrak{d} a_i \text{ for } i \in \{1, \ldots, n\} \tag{RDP}$$

The base case ($n = 1$) is immediate. Suppose now that (RDP) holds for $n = j$ ($j > 1$); that is, whenever $b \mid_\mathfrak{d} [a_1 \cdots a_j]_\mathfrak{d}$, we have $b = [b_1 \cdots b_j]_\mathfrak{d}$ with $b_i \mid_\mathfrak{d} a_i$ for all $i \leq j$. Let $b \mid_\mathfrak{d} [a_1 \cdots a_j]_\mathfrak{d}$, and take $a_{j+1} \in M$. By Lem. 1.2.5(2), we have $b \mid_\mathfrak{d} [a_1 \cdots a_j a_{j+1}]_\mathfrak{d}$ and $b_j \mid_\mathfrak{d} a_j \cdot_\mathfrak{d} a_{j+1}$. By the induction hypothesis, we thus have $b_j = b'_j \cdot_\mathfrak{d} b_{j+1}$ with $b'_j \mid_\mathfrak{d} a_j$ and $b_{j+1} \mid_\mathfrak{d} a_{j+1}$. Hence $b = [b_1 \cdots b_{j-1} b'_j b_{j+1}]_\mathfrak{d}$, and (RDP) holds for $n = j + 1$. □

Corollary 3.1.2. $\langle \mathbf{F}_\Sigma, \sqsubseteq \rangle$ *has the RDP.*

From the above considerations, we obtain three fundamental laws linking $\cdot$ and $\Diamond$.

Proposition 3.1.3. *For all* $P, Q \subseteq \Sigma^*$, *the following conditions hold:*

$$\Diamond(P \cdot Q) \subseteq \Diamond P \cdot \Diamond Q \tag{K}$$

$$\Diamond(P \cdot Q) = \Diamond P \cup (P \cdot \Diamond Q) \tag{C}$$

$$P \cap Q \subseteq \Diamond(P \cdot Q) \tag{I}$$

Proof. It is straightforward to check that (K) is equivalent to (RDP$_*$). We further observe that $\langle \mathbf{F}_\Sigma, \sqsubseteq \rangle$ may be viewed as an expanded ternary frame in which string concatenation and the prefix order serve, respectively, as accessibility relations for language concatenation and prefix closure on $\wp(\Sigma^*)$[9]. In such a setting, (RDP$_*$) is precisely the first-order frame condition corresponding to (K).

For the left-to-right inclusion of (C), suppose that $w \in \Diamond(P \cdot Q)$. Then $w \sqsubseteq uv$ for some $u \in P$ and $v \in Q$. Let us reconsider Table 3.1. In Case 1, we have $w = u'$, for u' a proper prefix of u, and $v = \varepsilon$. Case 2 is analogous, except that $w = u' = u$. In either situation, we observe that $w \in \Diamond P$. In Case 3, we have $w = uv'$ for v' a proper, non-empty prefix of v. Since $u \in P$ and $v' \in \Diamond Q$, it follows that $w \in P \cdot \Diamond Q$. Finally, in Case 4, where $w = uv$, since $u \in P$ and $v \in Q \subseteq \Diamond Q$, we again obtain $w \in P \cdot \Diamond Q$. Conversely, suppose that $w \in \Diamond P \cup (P \cdot \Diamond Q)$. Therefore, w is either

[9] Abstracting away from the specific structure under consideration, this is nothing but the frame semantics employed for modal Lambek calculi within the framework of categorial type logics [13, 14, 109, 127].

a (possibly improper) prefix of some string in P, or it is obtained by concatenating a string in P with a (possibly improper) prefix of some string in Q. Observe that these conditions are precisely those expressed in Table 3.1; hence $w \in \Diamond(P \cdot Q)$.

Finally, suppose that $w \in P \cap Q$. Then $w^2 \in P \cdot Q$, and since $w \sqsubseteq w^2$, it follows that $w \in \Diamond(P \cdot Q)$. This establishes (I). □

Remark 3.1.4. One easily verifies that, for all $w, u, v \in \Sigma^*$, the first-order frame conditions corresponding to (C) and (I) are, respectively,

$$w \sqsubseteq uv \Leftrightarrow (w \sqsubseteq u \curlyvee \exists v' \, (v' \sqsubseteq v \curlywedge w \sqsubseteq uv')) \quad \text{and} \quad w \sqsubseteq w^2.$$

Observe that (K) and (I) hold for every operator $\Diamond_{|_\flat}$ defined from a divisibility preorder.

3.2 On the operator $\boxslash$

We now briefly discuss some properties of the residual $\boxslash$ of $\Diamond$, showing how it naturally arises in the trace-based specification of concurrent systems and why it provides a useful addition to our algebraic model for f-properties.

We first review some basic algebraic laws involving $\boxslash$ (complete proofs will be given in the next subsection, together with proofs of further equations). We have seen that, in the $\{\cap, \cup, \Diamond, \emptyset, \Sigma^*\}$-reduct of $\wp(\mathbf{F}_\Sigma)_+$, prefix closure is essentially an $\mathsf{S4}$ diamond. When examining the interaction between $\boxslash$ and $\Diamond$, however, the residuation law leads us to a setting that is instead reminiscent of $\mathsf{S5}$, as well as of tense logics with both future and past modalities. In particular, by Prop. 1.3.2(2), we have

$$P \subseteq \boxslash\Diamond P \quad \text{and} \quad \Diamond\boxslash P \subseteq P$$

for all $P \subseteq \Sigma^*$. Since the prefix relation is a partial order, by residuation, $\boxslash$ is a topological interior operator on $\langle \wp(\Sigma^*), \subseteq \rangle$, that is, it satisfies the *duals* of the four Kuratowski axioms, namely

(B_1) $\boxslash P \subseteq P$;

(B_2) $\boxslash\boxslash P = \boxslash P$;

(B_3) $\boxslash(P \cap Q) = \boxslash P \cap \boxslash Q$;

(B_4) $\boxslash\Sigma^* = \Sigma^*$.

for all $P, Q \subseteq \Sigma^*$. Analogously to the case of $\Diamond$, in the $\{\cap, \cup, \boxdot\}$-reduct of $\wp(\mathbf{F}_\Sigma)_+$, $\boxdot$ is an S4 box. Indeed, from (B_1)–(B_4), one derives, for example:

$$\boxdot P \cup \boxdot Q \subseteq \boxdot(P \cup Q) \quad \text{and} \quad \boxdot(P \cup Q) = \boxdot(\boxdot P \cup \boxdot Q)$$

Moreover, by (B_1), the empty set is a fixed point of $\boxdot$. By Prop. 1.3.3(2), we also know that the compositions $\boxdot\Diamond$ and $\Diamond\boxdot$ are, respectively, a closure and an interior operator. These modalities in fact coincide with $\Diamond$ and $\boxdot$ themselves: applying (1RES) to $\Diamond\Diamond P = \Diamond P$ and $\boxdot\boxdot P = \boxdot P$ yields $\Diamond P = \boxdot\Diamond P$ and $\boxdot P = \Diamond\boxdot P$, respectively. As for the interaction with language concatenation, one can show, for example, a condition dual to (K): since $\Diamond(\boxdot P \cdot \boxdot Q) \subseteq \Diamond\boxdot P \cdot \Diamond\boxdot Q$, $\Diamond\boxdot P \subseteq P$, and $\Diamond\boxdot Q \subseteq Q$, we obtain $\Diamond(\boxdot P \cdot \boxdot Q) \subseteq P \cdot Q$; hence, by (1RES), $\boxdot P \cdot \boxdot Q \subseteq \boxdot(P \cdot Q)$.

Most importantly, residuation, together with the closure and interior properties of our operators, implies that the fixed points of $\Diamond$ coincide with those of $\boxdot$. This not only entails that $\boxdot$ preserves $\{\varepsilon\}$, but also allows for an alternative characterisation of safety f-properties. This is what we are about to discuss.

Manna and Pnueli [115] showed that, in the context of infinite-trace systems, safety properties can be specified by appropriately combining past and future operators in temporal logic. In what follows, we denote by LTLB [62] the standard extension of classical LTL (with *next* and *until*) with the past connectives *previous* and *since*. Two LTLB-formulas φ, ψ are *initially equivalent* if $\alpha, 0 \Vdash \varphi$ iff $\alpha, 0 \Vdash \psi$, for all $\alpha \in \Sigma_{\mathcal{T}}^\omega$. It is well known that LTL and LTLB are equally expressive with respect to initial equivalence [71].

For an LTS $\mathcal{T}$, let $\mathtt{A}\colon \wp(\Sigma_{\mathcal{T}}^*) \to \wp(\Sigma_{\mathcal{T}}^\omega)$ be defined by $Q \mapsto \{\alpha \in \Sigma_{\mathcal{T}}^\omega \mid \mathit{pref}(\alpha) \subseteq Q\}$. This operator is used in [115] to define safety for infinite traces: a set $P \subseteq \Sigma_{\mathcal{T}}^\omega$ is a safety property if and only if $P = \mathtt{A}(Q)$ for some $Q \subseteq \Sigma_{\mathcal{T}}^*$. Let now φ be an LTLB-formula. We define $\mathrm{Sat}(\varphi) := \{\alpha \in \Sigma_{\mathcal{T}}^\omega \mid \alpha \Vdash \varphi\}$, that is, the property corresponding to φ.

We say that φ is a *past formula* if it is either an atomic proposition or if it contains only past operators. A finite trace $w \in \Sigma_{\mathcal{T}}^*$ *end-satisfies* a past formula φ (written $w \Vdash_{\mathrm{e}} \varphi$) if there exists $\alpha \in \Sigma_{\mathcal{T}}^\omega$ such that $w \sqsubseteq \alpha$ and $\alpha, |w| - 1 \Vdash \varphi$. In other words, $w \Vdash_{\mathrm{e}} \varphi$ if there exists an infinite extension α of w in which the satisfaction of φ depends on its truth at the last position of w, and therefore, since φ is a "backward looking" formula, on the whole w. We write $\mathrm{eSat}(\varphi) := \{w \in \Sigma_{\mathcal{T}}^* \mid w \Vdash_{\mathrm{e}} \varphi\}$ for the f-property given by all finite traces end-satisfying φ. An f-property $Q \subseteq \Sigma_{\mathcal{T}}^*$ is *expressible* in LTLB if $Q = \mathrm{eSat}(\varphi)$, for some past formula φ. Observe that safety properties $\mathtt{A}(\mathrm{eSat}(\varphi))$

coincide with the properties $\mathrm{Sat}(G\varphi)$, where G is the *globally* operator [115, p. 392]. Indeed, we have:

$$\begin{aligned}\alpha \in \mathrm{Sat}(G\varphi) &\Leftrightarrow \alpha \Vdash G\varphi \\ &\Leftrightarrow w \Vdash_{\mathrm{e}} \varphi \text{ for all } w \in \mathit{pref}(\alpha) \\ &\Leftrightarrow \mathit{pref}(\alpha) \subseteq \mathrm{eSat}(\varphi) \\ &\Leftrightarrow \alpha \in \mathtt{A}(\mathrm{eSat}(\varphi)).\end{aligned}$$

Formulas of the form $G\varphi$ are called *safety formulas*. It is therefore clear that every safety formula specifies a safety property expressible in LTL by means of a formula ψ that is initially equivalent to $G\varphi$.

Running example. Let us consider the safety property P_1 introduced in the previous section. We have seen that, for infinite traces, it can be specified by the LTL-formula $\neg(\neg \mathit{request}\ U\ (\mathit{response} \wedge \neg \mathit{request}))$, which is initially equivalent to the LTLB-formula $G(\mathit{response} \rightarrow P(\mathit{request}))$, where P is the *once* operator (see [156]).

We show how the above outlined framework readily adapts to the case of finite traces. Let PLTL_f be the *pure-past* counterpart of LTL_f. It is well-known that these two logics are expressively equivalent [49]. For a PLTL_f-formula φ, we denote by $\varphi_\triangleright$ an LTL_f-formula that is equivalent to φ. Then, in LTL_f, we have:

$$\begin{aligned}w \in \mathrm{Sat}(G\varphi_\triangleright) &\Leftrightarrow v \Vdash \varphi \text{ for all } v \in \mathit{pref}(w) \\ &\Leftrightarrow \mathit{pref}(w) \subseteq \mathrm{Sat}(\varphi_\triangleright) \\ &\Leftrightarrow w \in \boxdot\mathrm{Sat}(\varphi_\triangleright).\end{aligned}$$

Clearly, $G\varphi_\triangleright$ specifies a safety property. Indeed, $\mathrm{Sat}(G\varphi_\triangleright) = \boxdot\mathrm{Sat}(\varphi_\triangleright) = \Diamond\boxdot\mathrm{Sat}(\varphi_\triangleright)$. This allows us to provide an alternative, residuation-theoretic proof of Prop. 2.2.7:

$$\begin{aligned}P \subseteq \Sigma^*_{\mathcal{T}} \text{ is a safety property} &\Leftrightarrow P = \Diamond P \\ &\Leftrightarrow P = \boxdot P \\ &\Leftrightarrow P = \{w \in \Sigma^*_{\mathcal{T}} \mid \forall v(v \sqsubseteq w \Rightarrow v \in P)\} \\ &\Leftrightarrow P = \{w \in \Sigma^*_{\mathcal{T}} \mid \mathit{pref}(w) \subseteq P\}\end{aligned}$$

We conclude that the residuation relation between $\Diamond$ and $\boxdot$ underlies the finite-trace variant of the Manna-Pnueli method for specifying safety properties by means of past

formulas. To the best of our knowledge, this connection between residuation and the formal specification of properties has not been explicitly recognised in the model-checking and temporal-logic communities, and it deserves further investigation.

3.3 Closure ℓ-monoids

In this section, we introduce and study the variety of closure ℓ-monoids, conceived as an algebraic abstraction—amenable to a characterisation in structural proof-theoretic terms—of f-properties specified by means of $\Diamond$ and $\boxdot$.

Definition 3.3.1. A *closure ℓ-monoid* is an algebra $\mathbf{A} = \langle A, \cdot, \wedge, \vee, \Diamond, \boxdot, 1, \bot, \top\rangle$ such that $\langle A, \wedge, \vee, \bot, \top\rangle$ is a bounded distributive lattice, $\langle A, \cdot, \wedge, \vee, 1\rangle$ is an ℓ-monoid, $\bot$ is an absorbing element for monoid multiplication, and $\Diamond, \boxdot$ are unary operations defined by the following equations:

1. $x \leq \Diamond x$
2. $\Diamond\Diamond x \approx \Diamond x$
3. $\Diamond(x \vee y) \approx \Diamond x \vee \Diamond y$
4. $\boxdot(x \wedge y) \approx \boxdot x \wedge \boxdot y$
5. $\Diamond(x \cdot y) \leq \Diamond x \cdot \Diamond y$
6. $\Diamond\boxdot x \leq x$
7. $x \leq \boxdot\Diamond x$

We denote by $\mathfrak{LMC}$ the variety of closure ℓ-monoids.

Example 3.3.2. The algebra $\wp(\mathbf{F}_\Sigma)_+$ is a closure ℓ-monoid. In general, let $\langle \mathbf{M}, \preceq\rangle$ be a monoid endowed with a preorder satisfying the RDP. Then

$$\wp(\mathbf{M})_+ = \langle \wp(M), \cdot, \cap, \cup, \Diamond_\preceq, \boxdot_\preceq, \{1\}, \emptyset, M\rangle$$

is a closure ℓ-monoid. It follows that, for $\mathfrak{d} \in \{l, r\}$, any algebra $\wp(\mathbf{M})_+$ arising from $\langle \mathbf{M}, |_\mathfrak{d}\rangle$ is in $\mathfrak{LMC}$.

Recall that an interior operator ι on an ℓ-monoid $\mathbf{M}$ is a *co-nucleus* [73, §3.14.16] if the following conditions are satisfied, for all $a, b \in M$:

$$\iota(\iota(a) \cdot \iota(b)) = \iota(a) \cdot \iota(b) \tag{CN$_1$}$$
$$\iota(a) \cdot \iota(1) = \iota(1) \cdot \iota(a) = \iota(a) \tag{CN$_2$}$$

If ι satisfies just (CN$_1$), then it is called a *weak co-nucleus* [31]. Note that, in $\wp(\mathbf{F}_\Sigma)_+$, $\boxdot$ satisfies both (CN$_1$) and (CN$_2$).

Lemma 3.3.3. *In any closure ℓ-monoid, $\langle \Diamond, \boxdot \rangle$ is a residuated pair where $\Diamond$ is a topological closure operator and $\boxdot$ is a weak co-nucleus*[10]*. Moreover, the two operators preserve the bounds.*

Proof. Let $\mathbf{A} \in \mathfrak{LMC}$. By axioms (3) and (4), both $\Diamond$ and $\boxdot$ are order-preserving. Together with (6) and (7), this implies that $\langle \Diamond, \boxdot \rangle$ is a residuated pair (Prop. 1.3.2). Note that $\Diamond\bot = \bot$ by Prop. 1.3.3(4). It then follows from axioms (1), (2), and (3) that $\Diamond$ is a topological closure operator. Moreover, since $\top \leq \Diamond\top$ by (1), $\Diamond$ is bound-preserving.

We now prove that $\boxdot$ is a weak co-nucleus. Let $a, b \in A$. First, observe that from (1) and (6) we obtain $\boxdot a \leq \Diamond\boxdot a \leq a$, and therefore, by transitivity, $\boxdot a \leq a$. Hence $\boxdot$ is contractive, and in particular $\boxdot\boxdot a \leq \boxdot a$. Next, since (6) and (2) yield $\Diamond\Diamond\boxdot a \leq a$, repeated application of (1RES) gives $\boxdot a \leq \boxdot\boxdot a$. Thus $\boxdot$ is idempotent. Since it is also meet-preserving, we conclude that $\boxdot$ is an interior operator. It is in fact a topological, bound-preserving one, as $\boxdot\top = \top$ by Prop. 1.3.3(4) and $\boxdot\bot \leq \bot$ by contractiveness. It remains to verify that $\boxdot a \cdot \boxdot b = \boxdot(\boxdot a \cdot \boxdot b)$. By (5), $\Diamond(\boxdot a \cdot \boxdot b) \leq \Diamond\boxdot a \cdot \Diamond\boxdot b$. Note that $\Diamond\boxdot a = \boxdot a$: the right-to-left inequality follows from (1), while the converse is obtained from $\boxdot a \leq \boxdot\boxdot a$ using (1RES). Hence $\Diamond(\boxdot a \cdot \boxdot b) \leq \Diamond\boxdot a \cdot \Diamond\boxdot b$ can be rewritten as $\Diamond(\boxdot a \cdot \boxdot b) \leq \boxdot a \cdot \boxdot b$. Applying (1RES), we conclude that $\boxdot a \cdot \boxdot b \leq \boxdot(\boxdot a \cdot \boxdot b)$. Since the converse inequality holds by contractiveness, we obtain $\boxdot(\boxdot a \cdot \boxdot b) = \boxdot a \cdot \boxdot b$. □

In the proof of the preceding lemma we derived some important laws from the axioms in Definition 3.3.1. We report them below.

(i) $\boxdot x \leq x$;

(ii) $\boxdot\boxdot x \approx \boxdot x$;

(iii) $\boxdot(\boxdot x \cdot \boxdot y) \approx \boxdot x \cdot \boxdot y$;

(iv) $\Diamond\boxdot x \approx \boxdot x$

(v) $\Diamond\bot \approx \bot$;

(vi) $\Diamond\top \approx \top$;

(vii) $\boxdot\bot \approx \bot$;

(viii) $\boxdot\top \approx \top$;

In the next result we extend the above list with further noteworthy equations.

Lemma 3.3.4. *The following equations hold in any closure ℓ-monoid:*

[10] We thank Simon Santschi for pointing out the weak co-nuclearity of $\boxdot$.

(ix) $\Diamond(x \wedge y) \leq \Diamond x \wedge \Diamond y$;

(x) $\Box x \vee \Box y \leq \Box(x \vee y)$;

(xi) $\Box\Diamond x \approx \Diamond x$;

(xii) $\Diamond(\Diamond x \cdot \Diamond y) \approx \Diamond x \cdot \Diamond y$;

(xiii) $\Diamond(\Diamond x \wedge \Diamond y) \approx \Diamond x \wedge \Diamond y$;

(xiv) $\Box(\Box x \vee \Box y) \approx \Box x \vee \Box y$.

(xv) $\Box x \cdot \Box y \leq \Box(x \cdot y)$.

Moreover, $x \leq \Diamond y \vDash_{\mathfrak{LMC}} \Diamond x \leq \Diamond y$ *and* $\Diamond x \leq \Diamond y \vDash_{\mathfrak{LMC}} x \leq \Diamond y$.

Proof. Again, let $\mathbf{A} \in \mathfrak{LMC}$ and let $a, b \in A$.

(ix) Since $\Diamond$ is order-preserving, we have $\Diamond(a \wedge b) \leq \Diamond a$ and $\Diamond(a \wedge b) \leq \Diamond b$. The desired inequality then follows by lattice-theoretic properties.

(x) Dual argument to (ix).

(xi) By (i), we have $\Box\Diamond a \leq \Diamond a$. Moreover, by (2), $\Diamond\Diamond a \leq \Diamond a$, and hence by (1RES) we conclude that $\Diamond a \leq \Box\Diamond a$.

(xii) By (1), we have $\Diamond a \cdot \Diamond b \leq \Diamond(\Diamond a \cdot \Diamond b)$. As for the converse, (5) gives $\Diamond(\Diamond a \cdot \Diamond b) \leq \Diamond\Diamond a \cdot \Diamond\Diamond b$, and an application of (2) then yields the desired inequality.

(xiii) Analogous argument to (xii), using (ix) in place of (5).

(xiv) By (i), we have $\Box(\Box a \vee \Box b) \leq \Box a \vee \Box b$. Now, since $\Box$ is order-preserving, $\Box\Box a \leq \Box(\Box a \vee \Box b)$ and $\Box\Box b \leq \Box(\Box a \vee \Box b)$. It follows, by (ii) and lattice-theoretic properties, that the right-to-left inequality holds.

(xv) By (5), (6), and order preservation of $\cdot$, we have $\Diamond(\Box a \cdot \Box b) \leq \Diamond\Box a \cdot \Diamond\Box b \leq a \cdot b$. Hence, the desired inequality follows by transitivity and (1RES).

Finally, suppose that $a \leq \Diamond b$. Then $\Diamond a \leq \Diamond\Diamond b$, and hence, by (2), $\Diamond a \leq \Diamond b$. Conversely, if $\Diamond a \leq \Diamond b$, then, by (1) and transitivity, it follows that $a \leq \Diamond b$. □

Lemma 3.3.5. $\mathfrak{LMC} \neq \mathbb{V}(\wp(\mathbf{F}_\Sigma)_+)$.

Proof. We aim to show that there exists an equation satisfied by $\wp(\mathbf{F}_\Sigma)_+$ but not by $\mathfrak{LMC}$. Our candidate is $\Diamond(x \cdot y) \leq \Diamond x \vee (x \cdot \Diamond y)$, which abstracts the left-to-right direction of (C). Let $\mathbf{End}_A = \langle \mathrm{End}_A, \circ, \mathrm{id}_A \rangle$ be the monoid of self-maps on a set A.

Consider the closure ℓ-monoid $\wp(\mathbf{End}_A)_+ = \langle \wp(\mathrm{End}_A), \circ, \cap, \cup, \Diamond, \boxdot, \{\mathrm{id}_A\}, \emptyset, \mathrm{End}_A \rangle$, where $\Diamond = \Diamond_{|_l}$, $\boxdot = \boxdot_{|_l}$, and for all $F, G \subseteq \mathrm{End}_A$, $F \circ G := \{f \circ g \mid f \in F,\ g \in G\}$.

We begin by observing that, for $f, g \in \mathrm{End}_A$, $f \mid_l g$ iff there exists $f' \in \mathrm{End}_A$ such that $g = f \circ f'$, i.e., the diagram in Figure 3.1 is commutative.

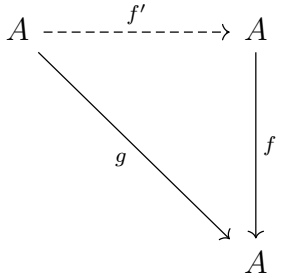


Figure 3.1: Left divisibility in $\mathbf{End}_A$.

We now establish that $\wp(\mathbf{End}_{\mathbb{Z}})_+ \not\models \Diamond(x \cdot y) \leq \Diamond x \vee (x \cdot \Diamond y)$. If the opposite were true, then, in particular $\Diamond(\{f \circ g\}) \subseteq \Diamond\{f\} \cup (\{f\} \circ \Diamond\{g\})$, for all $f, g \in \mathrm{End}_{\mathbb{Z}}$[11]. We show that, for a suitable choice of $f, g \in \mathrm{End}_{\mathbb{Z}}$, there exists $k \in \mathrm{End}_{\mathbb{Z}}$ such that $k \mid_l f \circ g$ and both of the following conditions hold:

1. $k \nmid_l f$;
2. for all $m \in \mathrm{End}_{\mathbb{Z}}$ such that $m \mid_l g$, it holds that $k \neq f \circ m$.

Let $f, g \in \mathrm{End}_{\mathbb{Z}}$ be defined as follows, for $n \in \mathbb{Z}$:

$$f(n) := \begin{cases} 1 & \text{if } n \leq 0, \\ 2 & \text{if } n > 0 \end{cases} \qquad\qquad g(n) := -1$$

Then $f \circ g$ is a constant map, as $f(g(n)) = 1$ for all $n \in \mathbb{Z}$. We now seek a left divisor k of $f \circ g$ (with respect to $\circ$) such that there is no $h \in \mathrm{End}_{\mathbb{Z}}$ satisfying $k \circ h = f$. Define

$$k(n) := \begin{cases} 0 & \text{if } n \leq 0, \\ 1 & \text{if } n > 0 \end{cases}$$

and let $k' : n \mapsto 2$. Clearly, for all $n \in \mathbb{Z}$, $k(k'(n)) = 1 = f(g(n))$ and therefore $k \mid_l f \circ g$. The functions f and k have, respectively, ranges $\{1, 2\}$ and $\{0, 1\}$, hence $k \circ h \neq f$ for all $h \in \mathrm{End}_{\mathbb{Z}}$. This establishes (1). The proof of (2) is immediate: as f takes only the values 1 and 2, it is clear that $k \neq f \circ m$ for all $m \in \mathrm{End}_{\mathbb{Z}}$, and hence (2) holds. □

[11] Note that $\Diamond(\{f\} \circ \{g\}) = \Diamond(\{f \circ g\})$.

Chapter 4

The Gentzen-style system LMC

In this chapter, we introduce the Gentzen-style calculus LMC for closure ℓ-monoids.

4.1 Calculus design

Definition 4.1.1. Let ν_1 be the similarity type $\langle 2,2,2,1,1,0,0,0\rangle$. The set Fm of LMC-*formulas* is the set of ν_1-terms generated over a denumerable set X of variables according to the BNF:

$$\varphi \ ::= \ x \mid 1 \mid \bot \mid \top \mid \psi_1 \cdot \psi_2 \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \Diamond\psi \mid \boxplus\psi$$

where $x \in X$ and $\psi, \psi_1, \psi_2 \in Fm$. We define $\mathbf{Fm} := \mathbf{T}_{\nu_1}(X)$.

We now introduce a measure for the complexity of formulas. Let $\mathrm{cp}\colon Fm \to \mathbb{N}$ be the map defined as follows:

$$\begin{array}{ll} \mathrm{cp}(\varphi) = 0 & \text{if } \varphi \in X \cup \{1, \bot, \top\} \\ \mathrm{cp}(\varphi \divideontimes \psi) = \mathrm{cp}(\varphi) + \mathrm{cp}(\psi) + 1 & \text{for } \divideontimes \in \{\cdot, \wedge, \vee\} \\ \mathrm{cp}(\bigcirc\varphi) = \mathrm{cp}(\varphi) + 1 & \text{for } \bigcirc \in \{\Diamond, \boxplus\} \end{array}$$

The structural terms of LMC are generated inductively from formulas by means of two binary structural operators $\circ$ and $\sqcap$, a unary structural operator $\langle - \rangle$, and a structural constant ϵ. In particular, $\langle \circ, \epsilon \rangle$ and $\sqcap$ correspond, respectively, to monoid- and meet-semilattice-like term grouping, while $\langle - \rangle$ is intended to express $\Diamond$-like term modalisation.

Definition 4.1.2. Let ν_2 be the similarity type $\langle 2, 2, 1, 0\rangle$. The set *Struct* of *structural terms* for LMC is the set of ν_2-terms generated over Fm according to the BNF:

$$\Gamma \ ::= \ \varphi \mid \epsilon \mid \Delta_1 \circ \Delta_2 \mid \Delta_1 \sqcap \Delta_2 \mid \langle\Delta\rangle$$

where $\varphi \in Fm$, $\Delta, \Delta_1, \Delta_2 \in Struct$. We define $\mathbf{Struct} := \mathbf{T}_{\nu_2}(Fm)$.

The complexity of structural terms is measured by the map $\mathrm{cp}_s : Struct \to \mathbb{N}$ defined by:

$$\begin{array}{ll} \mathrm{cp}_s(\Gamma) = 0 & \text{if } \Gamma \in Fm \cup \{\epsilon\} \\ \mathrm{cp}_s(\Gamma \divideontimes \Delta) = \mathrm{cp}_s(\Gamma) + \mathrm{cp}_s(\Delta) + 1 & \text{for } \divideontimes \in \{\circ, \sqcap\} \\ \mathrm{cp}_s(\langle\Gamma\rangle) = \mathrm{cp}_s(\Gamma) + 1 & \end{array}$$

Remark 4.1.3. It is important to observe that the syntax of structural terms is not two-layered, in the sense that the construction of a formula does not constitute a special case of constructing a structural term. From an algebraic perspective, each $\Gamma \in Struct$ is just a polynomial symbol generated over Fm, and therefore LMC-formulas (of any complexity) should be regarded as *atomic structural terms*. This means that, e.g., x is a subformula (but *not* a structural subterm) of $x \wedge y$.

Definition 4.1.4. An LMC-*sequent* is a pair $\langle\Gamma, \varphi\rangle$, denoted $\Gamma \succ \varphi$, where $\varphi \in Fm$ and $\Gamma \in Struct$.

Axioms and inference rules of LMC are provided in Table 4.1 where:

- For $\Gamma, \Delta \in Struct$, we write $\Gamma\{\Delta\}$ to indicate a distinguished occurrence of Δ as a structural subterm of Γ. We also stipulate that, in any structural term containing multiple occurrences of expressions of the form $\Gamma\{\Delta\}$, each instance refers to the same specific occurrence of Δ within Γ.
- A rule of the form $\dfrac{\Gamma \succ \varphi}{\Delta \succ \psi}$ is shorthand for the two rules $\dfrac{\Gamma \succ \varphi}{\Delta \succ \psi}$ and $\dfrac{\Delta \succ \psi}{\Gamma \succ \varphi}$

In the subsequent discussion, we will often need to easily switch between the proof-theoretic language of LMC and the algebraic language of closure ℓ-monoids, and *vice versa*. By the *formula translation* of a structural term we mean a map ${}^\natural\colon Struct \to Fm$ recursively defined as follows:

$$\begin{array}{ll} \Gamma^\natural = \Gamma \text{ (if } \Gamma \in Fm) & \\ (\Gamma \circ \Delta)^\natural = \Gamma^\natural \cdot \Delta^\natural & (\Gamma \sqcap \Delta)^\natural = \Gamma^\natural \wedge \Delta^\natural \\ \langle\Gamma\rangle^\natural = \Diamond\Gamma^\natural & \epsilon^\natural = 1 \end{array}$$

Axioms and structural rules

$$\frac{}{\varphi \succ \varphi}\ \mathsf{init} \qquad \frac{\Delta \succ \varphi \qquad \Gamma\{\varphi\} \succ \psi}{\Gamma\{\Delta\} \succ \psi}\ \mathsf{cut} \qquad \frac{\Gamma\{((\Delta_1 \circ \Delta_2) \circ \Delta_3)\} \succ \varphi}{\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\} \succ \varphi}\ \circ\mathsf{A}$$

$$\frac{\Gamma\{\Delta\} \succ \varphi}{\Gamma\{\Delta \circ \epsilon\} \succ \varphi}\ \circ\epsilon \qquad \frac{\Gamma\{\Delta\} \succ \varphi}{\Gamma\{\epsilon \circ \Delta\} \succ \varphi}\ \epsilon\circ \qquad \frac{\Gamma\{((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)\} \succ \varphi}{\Gamma\{(\Delta_1 \sqcap (\Delta_2 \sqcap \Delta_3))\} \succ \varphi}\ \sqcap\mathsf{A}$$

$$\frac{\Gamma\{\Delta_1\} \succ \varphi}{\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi}\ \sqcap\mathsf{W}_1 \qquad \frac{\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi}{\Gamma\{\Delta_2 \sqcap \Delta_1\} \succ \varphi}\ \sqcap\mathsf{E} \qquad \frac{\Gamma\{\Delta \sqcap \Delta\} \succ \varphi}{\Gamma\{\Delta\} \succ \varphi}\ \sqcap\mathsf{C}$$

$$\frac{\Gamma\{\langle\Delta_1\rangle \circ \langle\Delta_2\rangle\} \succ \varphi}{\Gamma\{\langle\Delta_1 \circ \Delta_2\rangle\} \succ \varphi}\ \mathsf{K} \qquad \frac{\Gamma\{\langle\Delta\rangle\} \succ \varphi}{\Gamma\{\Delta\} \succ \varphi}\ \mathsf{T} \qquad \frac{\Gamma\{\langle\Delta\rangle\} \succ \varphi}{\Gamma\{\langle\langle\Delta\rangle\rangle\} \succ \varphi}\ \mathsf{4}$$

Logical rules

$$\frac{\Gamma\{\varphi \circ \psi\} \succ \chi}{\Gamma\{\varphi \cdot \psi\} \succ \chi}\ \cdot\mathsf{L} \qquad \frac{\Gamma_1 \succ \varphi \qquad \Gamma_2 \succ \psi}{\Gamma_1 \circ \Gamma_2 \succ \varphi \cdot \psi}\ \cdot\mathsf{R} \qquad \frac{\Gamma \succ \varphi}{\Gamma \succ \varphi \cdot 1}\ \cdot 1 \qquad \frac{\Gamma \succ \varphi}{\Gamma \succ 1 \cdot \varphi}\ 1\cdot$$

$$\frac{\Gamma\{\varphi \sqcap \psi\} \succ \chi}{\Gamma\{\varphi \wedge \psi\} \succ \chi}\ \wedge\mathsf{L} \qquad \frac{\Gamma \succ \varphi \qquad \Gamma \succ \psi}{\Gamma \succ \varphi \wedge \psi}\ \wedge\mathsf{R}$$

$$\frac{\Gamma\{\varphi\} \succ \chi \qquad \Gamma\{\psi\} \succ \chi}{\Gamma\{\varphi \vee \psi\} \succ \chi}\ \vee\mathsf{L} \qquad \frac{\Gamma \succ \varphi}{\Gamma \succ \varphi \vee \psi}\ \vee\mathsf{R}_1 \qquad \frac{\Gamma \succ \psi}{\Gamma \succ \varphi \vee \psi}\ \vee\mathsf{R}_2$$

$$\frac{\Gamma\{\langle\varphi\rangle\} \succ \psi}{\Gamma\{\Diamond\varphi\} \succ \psi}\ \Diamond\mathsf{L} \qquad \frac{\Gamma \succ \varphi}{\langle\Gamma\rangle \succ \Diamond\varphi}\ \Diamond\mathsf{R} \qquad \frac{\Gamma\{\varphi\} \succ \psi}{\Gamma\{\langle\Box\varphi\rangle\} \succ \psi}\ \Box\mathsf{L} \qquad \frac{\langle\Gamma\rangle \succ \varphi}{\Gamma \succ \Box\varphi}\ \Box\mathsf{R}$$

$$\frac{\Gamma\{\epsilon\} \succ \varphi}{\Gamma\{1\} \succ \varphi}\ 1\mathsf{L} \qquad \frac{}{\epsilon \succ 1}\ 1\mathsf{R} \qquad \frac{}{\Gamma\{\bot\} \succ \varphi}\ \bot\mathsf{L} \qquad \frac{}{\Gamma \succ \top}\ \top\mathsf{R}$$

Table 4.1: The Gentzen-style system LMC.

The reader can easily check that $^\natural$ is a surjective homomorphism of **Struct** onto the $\{\vee, \Box, \bot\}$-free reduct of **Fm**. Formula translations of structural terms enable us to give an *(in)equational translation* for LMC-sequents, i.e. a map $^\sharp\colon \mathit{Struct} \times \mathit{Fm} \to \mathit{Fm}^2$ defined by $(\Gamma \succ \varphi)^\sharp = \Gamma^\natural \leq \varphi$. Now we want to go the other way round and introduce

a *sequent translation* for equations of type ν. Since every equation $\varphi \approx \psi$ may be seen as a pair of inequations $\{\varphi \leq \psi, \psi \leq \varphi\}$, we define a function

$$^\flat \colon Fm^2 \uplus Fm^2 \to (Struct \times Fm) \cup \wp(Struct \times Fm)$$

such that: $(\varphi \leq \psi)^\flat = \varphi \succ \psi$, for every inequation $\varphi \leq \psi$; $(\varphi \approx \psi)^\flat = \{\varphi \succ \psi, \psi \succ \varphi\}$, for every equation $\varphi \approx \psi$.

Proposition 4.1.5. *The following inference rules are derivable in* LMC.

$$\frac{\Gamma\{\Delta_2\} \succ \varphi}{\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi}\,\sqcap\mathsf{W}_2 \qquad \frac{\Gamma\{\varphi_1\} \succ \chi}{\Gamma\{\varphi_1 \wedge \varphi_2\} \succ \chi}\,\wedge\mathsf{L}_1 \qquad \frac{\Gamma\{\varphi_2\} \succ \chi}{\Gamma\{\varphi_1 \wedge \varphi_2\} \succ \chi}\,\wedge\mathsf{L}_2$$

$$\frac{\varphi_1 \succ \psi_1 \qquad \varphi_2 \succ \psi_2}{\varphi_1 \cdot \varphi_2 \succ \psi_1 \cdot \psi_2}\,\cdot\mathsf{iso} \qquad \frac{\varphi_1 \succ \psi_1 \qquad \varphi_2 \succ \psi_2}{\varphi_1 \wedge \varphi_2 \succ \psi_1 \wedge \psi_2}\,\wedge\mathsf{iso} \qquad \frac{\varphi_1 \succ \psi_1 \qquad \varphi_2 \succ \psi_2}{\varphi_1 \vee \varphi_2 \succ \psi_1 \vee \psi_2}\,\vee\mathsf{iso}$$

Proof. Left to the reader. □

In what follows, for r a rule of LMC, $(\mathsf{r})\!\downarrow_n$ denotes n successive applications of r.

Lemma 4.1.6. *The sequent translations of all equations in* Def. 3.3.1 *are derivable in* LMC.

Proof. We first give derivations for equations (1), (2), (5), (6), and (7).

$$\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle x\rangle \succ \Diamond x}\,\Diamond\mathsf{R}}{x \succ \Diamond x}\,\mathsf{T} \qquad \dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle \Box x\rangle \succ x}\,\Box\mathsf{L}}{\Diamond\Box x \succ x}\,\Diamond\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle x\rangle \succ \Diamond x}\,\Diamond\mathsf{R}}{x \succ \Box\Diamond x}\,\Box\mathsf{R}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle x\rangle \succ \Diamond x}\,\Diamond\mathsf{R}}{\langle\langle x\rangle\rangle \succ \Diamond x}\,4}{\Diamond\Diamond x \succ \Diamond x}\,(\Diamond\mathsf{L})\!\downarrow_2 \qquad \dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle x\rangle \succ \Diamond x}\,\Diamond\mathsf{R} \qquad \dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{\langle y\rangle \succ \Diamond y}\,\Diamond\mathsf{R}}{\langle x\rangle \circ \langle y\rangle \succ \Diamond x \cdot \Diamond y}\,\cdot\mathsf{R}}{\langle x \circ y\rangle \succ \Diamond x \cdot \Diamond y}\,\mathsf{K}}{\langle x \cdot y\rangle \succ \Diamond x \cdot \Diamond y}\,\cdot\mathsf{L}}{\Diamond(x \cdot y) \succ \Diamond x \cdot \Diamond y}\,\Diamond\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle x\rangle \succ \Diamond x}\,\Diamond\mathsf{R}}{x \succ \Diamond x}\,\mathsf{T}}{\langle x\rangle \succ \Diamond\Diamond x}\,\Diamond\mathsf{R}}{\Diamond x \succ \Diamond\Diamond x}\,\Diamond\mathsf{L}$$

Note that, since $(\Diamond\Diamond x \approx \Diamond x)^\flat = \{\Diamond\Diamond x \succ \Diamond x, \Diamond x \succ \Diamond\Diamond x\}$ the equation (2) corresponds to two derivations (on the bottom right and on the bottom left). The same applies to (3) and (4). By way of example, we provide the derivations of $\Diamond(x \vee y) \succ \Diamond x \vee \Diamond y$ and $\Box x \wedge \Box y \succ \Box(x \wedge y)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle x\rangle \succ \Diamond x}\,\Diamond\mathsf{R}}{\langle x\rangle \succ \Diamond x \vee \Diamond y}\,\vee\mathsf{R}_1 \qquad \dfrac{\dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{\langle y\rangle \succ \Diamond y}\,\Diamond\mathsf{R}}{\langle y\rangle \succ \Diamond x \vee \Diamond y}\,\vee\mathsf{R}_2}{\langle x \vee y\rangle \succ \Diamond x \vee \Diamond y}\,\vee\mathsf{L}}{\Diamond(x \vee y) \succ \Diamond x \vee \Diamond y}\,\Diamond\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle \Box x\rangle \succ x}\,\Box\mathsf{L}}{\langle \Box x \wedge \Box y\rangle \succ x}\,\wedge\mathsf{L}_1 \qquad \dfrac{\dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{\langle \Box y\rangle \succ y}\,\Box\mathsf{L}}{\langle \Box x \wedge \Box y\rangle \succ y}\,\wedge\mathsf{L}_2}{\langle \Box x \wedge \Box y\rangle \succ x \wedge y}\,\wedge\mathsf{R}}{\Box x \wedge \Box y \succ \Box(x \wedge y)}\,\Box\mathsf{R}$$

The converse cases are left to the reader. □

Example 4.1.7. It is not difficult to exhibit derivations in LMC for the equations proved in Lemmas 3.3.3 and 3.3.4. For example, here are proofs for the weak co-nuclearity of $\Box$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle\Box x\rangle \succ x}\,\Box\mathsf{L}}{\langle\langle\Box x\rangle\rangle \succ x}\,4}{\langle\Box x\rangle \succ \Box x}\,\Box\mathsf{R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{\langle\Box y\rangle \succ y}\,\Box\mathsf{L}}{\langle\langle\Box y\rangle\rangle \succ y}\,4}{\langle\Box y\rangle \succ \Box y}\,\Box\mathsf{R}}{\langle\Box x\rangle \circ \langle\Box y\rangle \succ \Box x \cdot \Box y}\,\cdot\mathsf{R}}{\langle\Box x \circ \Box y\rangle \succ \Box x \cdot \Box y}\,\mathsf{K}}{\langle\Box x \cdot \Box y\rangle \succ \Box x \cdot \Box y}\,\cdot\mathsf{L}}{\langle\langle\Box(\Box x \cdot \Box y)\rangle\rangle \succ \Box x \cdot \Box y}\,\Box\mathsf{L}}{\Box(\Box x \cdot \Box y) \succ \Box x \cdot \Box y}\,(\mathsf{T})\!\downarrow_2$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{\langle\Box x\rangle \succ x}\,\Box\mathsf{L}}{\langle\langle\Box x\rangle\rangle \succ x}\,4}{\langle\Box x\rangle \succ \Box x}\,\Box\mathsf{R} \qquad \dfrac{\dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{\langle\Box y\rangle \succ y}\,\Box\mathsf{L}}{\langle\langle\Box y\rangle\rangle \succ y}\,4}{\langle\Box y\rangle \succ \Box y}\,\Box\mathsf{R}}{\langle\Box x\rangle \circ \langle\Box y\rangle \succ \Box x \cdot \Box y}\,\cdot\mathsf{R}}{\langle\Box x \circ \Box y\rangle \succ \Box x \cdot \Box y}\,\mathsf{K}}{\Box x \circ \Box y \succ \Box(\Box x \cdot \Box y)}\,\Box\mathsf{R}}{\Box x \cdot \Box y \succ \Box(\Box x \cdot \Box y)}\,\cdot\mathsf{L}$$

4.2 Algebraic completeness

We are now going to show that LMC is sound and complete with respect to the variety of closure ℓ-monoids. First, we check the soundness of individual inference rules.

Lemma 4.2.1. *Let* $\dfrac{s_1, \ldots, s_n}{s}$ *be a rule of* LMC. *Then* $s_1^\sharp, \ldots, s_n^\sharp \vDash_{\mathfrak{LMC}} s^\sharp$.

Proof. The proof is trivial for almost all the rules displayed in Table 4.1. The only rules for which some caution is required are cut, $\sqcap\mathsf{W}_1$, K, $\vee\mathsf{L}$, and $\Box\mathsf{L}$. The reason is that, in these cases, the subterms appearing in the left-hand sides of sequents are permitted to occur at an arbitrary depth, yet the rule's structure alone does not guarantee soundness (unlike, for instance, with $\circ\,\epsilon$, $\sqcap\mathsf{A}$, or $\Diamond\mathsf{L}$).

- Let us start by considering the rules cut, $\sqcap\mathsf{W}_1$, K, and $\Box\mathsf{L}$. We first note that all fundamental operations of closure ℓ-monoids are order-preserving. This implies that the interpretation of formula translations for structural terms results in term operations that are inherently order-preserving. Take now $\Gamma\{-\}$ to be a structural term with an empty argument place. For $\Delta \in \mathit{Struct}$ and $\varphi, \psi \in \mathit{Fm}$, suppose that $\mathfrak{LMC}$ satisfies $\Gamma\{\varphi\}^\natural \leq \psi$ and $\Delta^\natural \leq \varphi$. But then $\mathfrak{LMC} \models \Gamma\{\Delta\}^\natural \leq \Gamma\{\varphi\}^\natural$ whence, by transitivity, $\mathfrak{LMC} \models \Gamma\{\Delta\}^\natural \leq \psi$. This establishes the soundness of cut. Analogous arguments apply to the remaining cases.

- We now turn to $\vee\mathsf{L}$. Here, it is easy to see that order preservation is of no help in checking soundness for this rule. However, by examining the syntax of structural

terms, we can observe that the counterpart of each structural operator is a join-distributive connective. Therefore, to prove that $\vee\mathsf{L}$ is sound, it is enough to show that the equation

$$E := \Gamma\{\varphi\}^\natural \vee \Gamma\{\psi\}^\natural \approx \Gamma\{\varphi \vee \psi\}^\natural$$

is satisfied by $\mathfrak{LMC}$. We proceed by induction on $\mathrm{cp}_s(\Gamma)$, recalling that the formulas, understood as structural terms, have complexity 0. In the base case our claim clearly holds. Suppose now that $\mathfrak{LMC} \models E$ for all the instances of $\vee\mathsf{L}$ with structural terms of complexity $\leq n$. For $\xi \in \{\varphi, \psi, \varphi \vee \psi\}$, let $\Gamma\{\xi\} := \Delta \circ \Gamma'\{\xi\}$, where $\mathrm{cp}_s(\Gamma') = n$. Since

$$\Delta^\natural \cdot (\Gamma'\{\varphi\}^\natural \vee \Gamma'\{\psi\}^\natural) \approx (\Delta^\natural \cdot \Gamma'\{\varphi\}^\natural) \vee (\Delta^\natural \cdot \Gamma'\{\psi\}^\natural)$$

is an instance of a defining equation of $\mathfrak{LMC}$ and, by induction hypothesis, $\mathfrak{LMC} \models \Gamma'\{\varphi\}^\natural \vee \Gamma'\{\psi\}^\natural \approx \Gamma'\{\varphi \vee \psi\}^\natural$, we immediately obtain that

$$\mathfrak{LMC} \models (\Delta^\natural \cdot \Gamma'\{\varphi\}^\natural) \vee (\Delta^\natural \cdot \Gamma'\{\psi\}^\natural) \approx \Delta^\natural \cdot \Gamma'\{\varphi \vee \psi\}^\natural.$$

We leave it to the reader to check the cases in which $\Gamma\{\xi\}$ is defined as $\Gamma'\{\xi\} \circ \Delta$, $\Delta \sqcap \Gamma'\{\xi\}$, $\Gamma'\{\xi\} \sqcap \Delta$, and $\langle\Gamma'\{\xi\}\rangle$. □

Corollary 4.2.2. *Any derivable rule of* LMC *is sound.*

Theorem 4.2.3. *Let* $\Gamma \succ \varphi$ *be an* LMC*-sequent. Then* $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$ *iff* $\mathfrak{LMC} \models (\Gamma \succ \varphi)^\sharp$.

Proof. Drawing upon Lemma 4.2.1, soundness is obtained via a straightforward induction on the height of derivations in LMC. We establish completeness via the Lindenbaum-Tarski method. Let $\Theta(\mathsf{LMC}) \subseteq Fm^2$ be defined by

$$\Theta(\mathsf{LMC}) := \left\{\langle\varphi, \psi\rangle \in Fm^2 \mid \vdash_{\mathsf{LMC}} \varphi \succ \psi \text{ and } \vdash_{\mathsf{LMC}} \psi \succ \varphi\right\}.$$

It is easy to see that $\Theta(\mathsf{LMC})$ is an equivalence relation on Fm: it is symmetric by design, while reflexivity and transitivity follow from the rules init and cut, respectively. We now show that $\Theta(\mathsf{LMC})$ is a congruence on $\mathbf{Fm}$. For $\varphi_1, \varphi_2, \psi_1, \psi_2 \in Fm$, suppose that $\langle\varphi_1, \psi_1\rangle, \langle\varphi_2, \psi_2\rangle \in \Theta(\mathsf{LMC})$. By using the rules $\cdot\mathsf{iso}$, $\wedge\mathsf{iso}$, and $\vee\mathsf{iso}$ from Proposition 4.1.5, we can derive in one step the sequents:

$$\begin{array}{ll} \varphi_1 \cdot \varphi_2 \succ \psi_1 \cdot \psi_2 & \psi_1 \cdot \psi_2 \succ \varphi_1 \cdot \varphi_2 \\ \varphi_1 \wedge \varphi_2 \succ \psi_1 \wedge \psi_2 & \psi_1 \wedge \psi_2 \succ \varphi_1 \wedge \varphi_2 \\ \varphi_1 \vee \varphi_2 \succ \psi_1 \vee \psi_2 & \psi_1 \vee \psi_2 \succ \varphi_1 \vee \varphi_2 \end{array}$$

Therefore $\langle \varphi_1 \divideontimes \varphi_2, \psi_1 \divideontimes \psi_2 \rangle \in \Theta(\mathsf{LMC})$, for $\divideontimes \in \{\cdot, \wedge, \vee\}$. Suppose now that $\langle \varphi, \psi \rangle \in \Theta(\mathsf{LMC})$. Then $\langle \Diamond\varphi, \Diamond\psi \rangle, \langle \Box\varphi, \Box\psi \rangle \in \Theta(\mathsf{LMC})$, as shown by the following proofs:

$$\dfrac{\dfrac{\dfrac{d}{\varphi \succ \psi}}{\langle\varphi\rangle \succ \Diamond\psi}\Diamond\mathsf{R}}{\Diamond\varphi \succ \Diamond\psi}\Diamond\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{d'}{\psi \succ \varphi}}{\langle\psi\rangle \succ \Diamond\varphi}\Diamond\mathsf{R}}{\Diamond\psi \succ \Diamond\varphi}\Diamond\mathsf{L} \qquad \dfrac{\dfrac{\dfrac{d}{\varphi \succ \psi}}{\langle\Box\varphi\rangle \succ \psi}\Box\mathsf{L}}{\Box\varphi \succ \Box\psi}\Box\mathsf{R} \qquad \dfrac{\dfrac{\dfrac{d'}{\psi \succ \varphi}}{\langle\Box\psi\rangle \succ \varphi}\Box\mathsf{L}}{\Box\psi \succ \Box\varphi}\Box\mathsf{R}$$

We conclude that $\Theta(\mathsf{LMC}) \in \mathrm{Con}(\mathbf{Fm})$. We proceed to show that the quotient algebra $\mathbf{Fm}/\Theta(\mathsf{LMC})$ is a closure ℓ-monoid. Let $\leq_{\mathsf{LMC}}$ be the binary relation on $Fm/\Theta(\mathsf{LMC})$ defined by $[\varphi] \leq_{\mathsf{LMC}} [\psi]$ iff $\vdash_{\mathsf{LMC}} \varphi \succ \psi$, for $\varphi, \psi \in Fm$. Clearly, $\leq_{\mathsf{LMC}}$ is a partial order (reflexifity and transitivity again follow from the rules init and cut; antisymmetry holds by design). Most importantly, $\leq_{\mathsf{LMC}}$ is a lattice order. Indeed it is immediate to check that $[\varphi] \leq_{\mathsf{LMC}} [\psi]$ is equivalent to $[\varphi] \wedge [\psi] = [\varphi]$. Therefore $\langle Fm/\Theta(\mathsf{LMC}), \wedge, \vee \rangle$ is a lattice. By constructing suitable derivations, it can be readily verified that, for all $\varphi, \psi, \chi \in Fm$:

$$\begin{aligned}
&([\varphi] \cdot [\psi]) \cdot [\chi] = [\varphi] \cdot ([\psi] \cdot [\chi]) \\
&[\varphi] \cdot [1] = [1] \cdot [\varphi] = [\varphi] \\
&[\varphi] \cdot [\bot] = [\bot] \cdot [\varphi] = [\bot] \\
&[\bot] \leq_{\mathsf{LMC}} [\varphi] \text{ and } [\varphi] \leq_{\mathsf{LMC}} [\top] \\
&[\varphi] \leq_{\mathsf{LMC}} [\psi] \Rightarrow [\varphi] \cdot [\chi] \leq_{\mathsf{LMC}} [\psi] \cdot [\chi] \\
&[\varphi] \leq_{\mathsf{LMC}} [\psi] \Rightarrow [\chi] \cdot [\varphi] \leq_{\mathsf{LMC}} [\chi] \cdot [\psi]
\end{aligned}$$

Hence $\langle Fm/\Theta(\mathsf{LMC}), \wedge, \vee, \cdot, [1], [\bot], [\top] \rangle$ is a bounded ℓ-monoid with $[\bot]$ an absorbing element for multiplication. In order to prove distributive laws for lattice operations, the role of $\sqcap$ is crucial. We just construct derivations for $x \wedge (y \vee z) \succ (x \wedge y) \vee (x \wedge z)$ and $(x \vee y) \wedge (x \vee z) \succ x \vee (y \wedge z)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{x \sqcap y \succ x}\sqcap\mathsf{W}_1 \quad \dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{x \sqcap y \succ y}\sqcap\mathsf{W}_2}{x \sqcap y \succ x \wedge y}\wedge\mathsf{R}}{x \sqcap y \succ (x \wedge y) \vee (x \wedge z)}\vee\mathsf{R}_1 \quad \dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{x \sqcap z \succ x}\sqcap\mathsf{W}_1 \quad \dfrac{\dfrac{}{z \succ z}\,\mathsf{init}}{x \sqcap z \succ z}\sqcap\mathsf{W}_2}{x \sqcap z \succ x \wedge z}\wedge\mathsf{R}}{x \sqcap y \succ (x \wedge y) \vee (x \wedge z)}\vee\mathsf{R}_2}{x \sqcap (y \vee z) \succ (x \wedge y) \vee (x \wedge z)}\vee\mathsf{L}}{x \wedge (y \vee z) \succ (x \wedge y) \vee (x \wedge z)}\wedge\mathsf{L}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{(x \vee y) \sqcap x \succ x}\,\sqcap\mathsf{W}_2}{(x \vee y) \sqcap x \succ x \vee (y \wedge z)}\,\vee\mathsf{R}_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x \succ x}\,\mathsf{init}}{x \sqcap z \succ x}\,\sqcap\mathsf{W}_1}{x \sqcap z \succ x \vee (y \wedge z)}\,\vee\mathsf{R}_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{y \succ y}\,\mathsf{init}}{y \sqcap z \succ y}\,\sqcap\mathsf{W}_1 \qquad \dfrac{\dfrac{}{z \succ z}\,\mathsf{init}}{y \sqcap z \succ z}\,\sqcap\mathsf{W}_2}{y \sqcap z \succ y \wedge z}\,\wedge\mathsf{R}}{y \sqcap z \succ x \vee (y \wedge z)}\,\vee\mathsf{R}_2}{(x \vee y) \sqcap z \succ x \vee (y \wedge z)}\,\vee\mathsf{L}}{(x \vee y) \sqcap (x \vee z) \succ x \vee (y \wedge z)}\,\vee\mathsf{L}}{(x \vee y) \wedge (x \vee z) \succ x \vee (y \wedge z)}\,\wedge\mathsf{L}$$

Finally, by using Lemma 4.1.6, we immediately see that operations $\Diamond[\varphi] := [\Diamond\varphi]$ and $\boxplus[\varphi] := [\boxplus\varphi]$ satisfy equations (1)-(7) in Def. 3.3.1. Therefore $\mathbf{Fm}/\Theta(\mathsf{LMC}) \in \mathfrak{LMC}$. Let now $h\colon \mathbf{Fm} \to \mathbf{Fm}/\Theta(\mathsf{LMC})$ be the natural homomorphism defined by $\varphi \mapsto [\varphi]$ and suppose that, for $\Gamma \succ \varphi$ an LMC-sequent, $\mathfrak{LMC} \models (\Gamma \succ \varphi)^\sharp$. Therefore $\mathfrak{LMC} \models \Gamma^\natural \leq \varphi$ and, in particular, $h(\Gamma^\natural) \leq_{\mathsf{LMC}} h(\varphi)$ in $\mathbf{Fm}/\Theta(\mathsf{LMC})$. But then, by the definition of $\leq_{\mathsf{LMC}}$, we have $\vdash_{\mathsf{LMC}} \Gamma^\natural \succ \varphi$. Since obviously $\vdash_{\mathsf{LMC}} \Gamma \succ \Gamma^\natural$, by applying cut we get $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$. □

Remark 4.2.4. The sequent translation of the equation $\Diamond(x \cdot y) \approx \Diamond x \vee (x \cdot \Diamond y)$ is not derivable in LMC. In particular, by Lemma 3.3.5, $\mathfrak{LMC} \not\models \Diamond(x \cdot y) \leq \Diamond x \vee (x \cdot \Diamond y)$. Therefore, by Theorem 4.2.3, $\nvdash_{\mathsf{LMC}} \Diamond(x \cdot y) \succ \Diamond x \vee (x \cdot \Diamond y)$.

4.3 Cut elimination

This section is devoted to proving a cut elimination result for LMC. As the reader will have observed, in derivations of the form

$$\dfrac{\dfrac{d_1}{\Delta \succ \varphi} \qquad \dfrac{\dfrac{d_2}{\Gamma\{\Pi\{\varphi\} \sqcap \Pi\{\varphi\}\} \succ \psi}}{\Gamma\{\Pi\{\varphi\}\} \succ \psi}\,\sqcap\mathsf{C}}{\Gamma\{\Pi\{\Delta\}\} \succ \psi}\,\mathsf{cut}$$

permuting the $\sqcap$C with cut does not result in a decrease in the height of d_2. A direct proof of cut elimination for LMC is therefore precluded. This comes as no surprise, since analogous situations arise (with respect to contraction rules for different connectives) in many well-known logical calculi. We follow the standard approach of introducing a mix rule—which subsumes cut as a special case—to simplify the structure of derivations by eliminating, in a single move, derivation steps that obstruct induction in cut elimination arguments. We then establish a mix elimination theorem for LMC (without cut and with the mix rule), from which cut elimination is obtained as a direct corollary.

4.3.1 A (weak) mix rule for LMC

The form of a mix rule depends on the intrinsic features of the calculus for which it is designed. In our framework, two central aspects must be addressed. First, structural terms of LMC are not sets or multisets, but rather have a *tree structure*. Second, as in the case of the substructural logic $\mathsf{FL}_{\mathsf{gc}}$ [158], the rule $\sqcap\mathsf{C}$ operates on structural terms, hence it is a *global* contraction rule.

For $n \geq 0$, let us write $\Gamma\{\!|\Delta|\!\}_n$ to denote a structural term Γ where we distinguish a family of n occurrences of Δ. Note that each of these occurrences corresponds to a node in the syntactic tree associated with Γ. Clearly, $\Gamma\{\!|\Delta|\!\}_1$ is just an alternative notation for $\Gamma\{\Delta\}$. The notations $\Gamma\{-\}$ and $\Gamma\{\!|-|\!\}_n$ can also be combined: we thus write $\Gamma\{\Pi\}\{\!|\Delta|\!\}_n$ to denote a structural term Γ in which both a single occurrence of Π and n occurrences of Δ are distinguished. In the case where Π contains an occurrence of Δ, we write $\Gamma\{\Pi\{\underline{\Delta}\}\}\{\!|\Delta|\!\}_n$ if the selected Δ in Π is included in $\{\!|\Delta|\!\}_n$, and $\Gamma\{\Pi\{\Delta\}\}\{\!|\Delta|\!\}_n$ otherwise. In this setting, assuming the choice of all occurrences of Δ to be fixed, it is clear that structural terms of the form $\Gamma\{\Pi\{\underline{\Delta}\}\}\{\!|\Delta|\!\}_n$ and $\Gamma\{\Pi\{\Delta\}\}\{\!|\Delta|\!\}_{n-1}$ coincide.

We are now ready to define a mix rule for LMC.

Definition 4.3.1. Let us denote by mix the inference rule

$$\frac{\Delta \succ \varphi \qquad \Gamma\{\!|\varphi|\!\}_n \succ \psi}{\Gamma\{\!|\Delta|\!\}_n \succ \psi}$$

where $\Gamma\{\!|\Delta|\!\}_n$ is the structural term obtained by replacing, in $\Gamma\{\!|\varphi|\!\}_n$, each of the n occurrences of φ with an occurrence of Δ. In an instance of mix, φ is called the mix-formula. We define $\mathsf{LMC}_{\ltimes}$ as the system $(\mathsf{LMC} - \mathsf{cut}) + \mathsf{mix}$.

Note that mix is close in spirit to Kiriyama and Ono's *weak-mix* rule for (the propositional fragment of) $\mathsf{FL}_{\mathsf{ec}}$ [99], in that it permits the simultaneous substitution of some, but not necessarily all, occurrences of the mix-formula.

Remark 4.3.2. Every instance of cut is nothing but an instance of mix where just one occurrence of the mix-formula has been singled out. Conversely, every instance of mix over n occurrences of the mix-formula can be replaced by a sequence of n instances of cut.

4.3.2 Mix elimination for $\mathsf{LMC}_{\ltimes}$

We establish a mix elimination theorem for $\mathsf{LMC}_{\ltimes}$ by adapting the method of Metcalfe *et al.* [120], developed for cut elimination in hypersequent systems for fuzzy logics and subsequently applied to the hypersequent calculus CSemFL [121]. In what follows, for d a derivation in $\mathsf{LMC}_{\ltimes}$, we denote by $h(d)$ its height.

Definition 4.3.3. The *rank* of an instance of mix

$$\dfrac{\dfrac{d_1}{\Delta \succ \varphi} \qquad \dfrac{d_2}{\Gamma\{\!\{\varphi\}\!\}_n \succ \psi}}{\Gamma\{\!\{\Delta\}\!\}_n \succ \psi}\,\mathsf{mix}$$

is the ordered triple $\langle \mathrm{cp}(\varphi), p(d_1), h(d_1) + h(d_2)\rangle$, where $p(d_1) = 0$ if d_1 ends with an application of a rule that (read backwards) decomposes φ, and $p(d_1) = 1$ otherwise. We regard ranks as elements of the partially ordered set $\langle \mathbb{N}^3, \leq_{\mathrm{lex}}\rangle$, where $\leq_{\mathrm{lex}}$ is the lexicographic order defined from three copies of $\mathbb{N}$.

We write $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma \succ \varphi$ to denote that an $\mathsf{LMC}_{\ltimes}$-sequent $\Gamma \succ \varphi$ admits a mix-free proof.

Theorem 4.3.4. *The rule* mix *can be eliminated from* $\mathsf{LMC}_{\ltimes}$.

Proof. Our goal is to show that any derivation in $\mathsf{LMC}_{\ltimes}$ making use of mix can be transformed into a mix-free one. To this end, it suffices to consider uppermost instances of mix in derivations. We prove by induction on the mix rank that, if d_1 is a mix-free derivation of $\Delta \succ \varphi$ and d_2 is a mix-free derivation of $\Gamma\{\!\{\varphi\}\!\}_n \succ \psi$, then it is possible to construct a mix-free proof of $\Gamma\{\!\{\Delta\}\!\}_n \succ \psi$.

As for the base case, suppose that $h(d_1) + h(d_2) = 2$. We consider the following alternatives:

1. If d_1 and d_2 are instances of init, then $\varphi = \psi$ and our claim trivially holds.

2. If d_1 is an instance of $\bot\mathsf{L}$ and d_2 is an instance of $\top\mathsf{R}$, then we have the following transformation:

$$\dfrac{\dfrac{}{\Delta\{\bot\} \succ \varphi}\,\bot\mathsf{L} \qquad \dfrac{}{\Gamma\{\!\{\varphi\}\!\}_n \succ \top}\,\top\mathsf{R}}{\Gamma\{\!\{\Delta\{\bot\}\}\!\}_n \succ \top}\,\mathsf{mix} \quad \dashrightarrow \quad \dfrac{}{\Gamma\{\!\{\Delta\{\bot\}\}\!\}_n \succ \top}\,\bot\mathsf{L}/\top\mathsf{R}$$

3. If d_1 is $1\mathsf{R}$ and d_2 is the initial sequent $1 \succ 1$, then by applying mix we would obtain $\epsilon \succ 1$ and so the whole derivation reduces to $1\mathsf{R}$. The same reasoning applies if: d_1 is the initial sequent $\bot \succ \bot$ and d_2 is any instance of $\bot\mathsf{L}$; d_1 (d_2) is the initial sequent $\top \succ \top$ and the endsequent of d_2 (d_1) is of the form $\Gamma\{\top\} \succ \top$.

For the induction step, we distinguish two possible classes of cases:

(i) Either no new occurrence of the mix-formula is introduced on the right side of the endsequent of d_1 or no new occurrence of the mix-formula is introduced on the left side of the endsequent of d_2.

(ii) A new occurrence of the mix-formula is introduced both on the right side of the endsequent of d_1 and on the left side of the endsequent of d_2.

Let us begin with the cases falling under (i). Suppose that d_1 and d_2 end with applications of unary rules that, respectively, do not introduce any new occurrences of the mix-formula on the right side of the endsequent of d_1 and the left side of the endsequent of d_2. In this case, we have the following transformation:

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{\dfrac{d_2}{\Gamma' \succ \psi'}}{\Gamma\{\!\{\varphi\}\!\}_n \succ \psi}\,\mathsf{r}_2}{\Gamma\{\!\{\Delta\}\!\}_n \succ \psi}\,\mathsf{mix} \quad \dashrightarrow \quad \dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{d_2'}{\Gamma'\{\!\{\varphi\}\!\}_m \succ \psi'}}{\dfrac{\Gamma'\{\!\{\Delta\}\!\}_m \succ \psi'}{\Gamma\{\!\{\Delta\}\!\}_n \succ \psi}\,\mathsf{r}_2}\,\mathsf{mix} \tag{4.1}$$

Observe that, although r_2 cannot introduce any new occurrence of φ, it may still remove one (this is the case where $\mathsf{r}_2 = \sqcap\mathsf{C}$). Accordingly, in the right derivation pattern we require m to be either n or $n+1$. As $h(d_2') < h(d_2)$, the induction hypothesis (IH) applies, enabling us to conclude that $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma\{\!\{\Delta\}\!\}_n \succ \psi$.

If instead r_2 introduces a new occurrence of φ within Γ, then two cases must be considered. If r_2 is $\sqcap\mathsf{W}_1$, then mix can be smoothly permuted with $\sqcap\mathsf{W}_1$ as illustrated in the following transformation schema:

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{\dfrac{d_2}{\Gamma\{\Pi\} \succ \psi}}{\Gamma\{\Pi \sqcap \underline{\varphi}\}\{\!\{\varphi\}\!\}_n \succ \psi}\,\sqcap\mathsf{W}_1}{\Gamma\{\Pi \sqcap \underline{\Delta}\}\{\!\{\Delta\}\!\}_n \succ \psi}\,\mathsf{mix} \quad \dashrightarrow \quad \dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{d_2'}{\Gamma\{\Pi\}\{\!\{\varphi\}\!\}_{n-1} \succ \psi}}{\dfrac{\Gamma\{\Pi\}\{\!\{\Delta\}\!\}_{n-1} \succ \psi}{\Gamma\{\Pi \sqcap \Delta\}\{\!\{\Delta\}\!\}_{n-1} \succ \psi}\,\sqcap\mathsf{W}_1}\,\mathsf{mix} \tag{4.2}$$

Since $\Gamma\{\Pi \sqcap \Delta\}\{\!\{\Delta\}\!\}_{n-1} = \Gamma\{\Pi \sqcap \underline{\Delta}\}\{\!\{\Delta\}\!\}_n$ and $h(d_2') < h(d_2)$, by the IH we have $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma\{\Pi \sqcap \underline{\Delta}\}\{\!\{\Delta\}\!\}_n \succ \psi$.

If r_2 is of $\cdot\mathsf{L}$, $\wedge\mathsf{L}$, $\Diamond\mathsf{L}$, or $\boxplus\mathsf{L}$, then we start from derivations of the form:

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{\dfrac{d_2}{\Gamma\{\Pi'\} \succ \psi}}{\Gamma\{\Pi\{\underline{\varphi}\}\}\{\!\{\varphi\}\!\}_n \succ \psi}\,\mathsf{r}_2}{\Gamma\{\Pi\{\underline{\Delta}\}\}\{\!\{\Delta\}\!\}_n \succ \psi}\,\mathsf{mix} \tag{4.3}$$

First, we consider the penultimate step of r_2 and replace the possible $n-1$ occurrences of φ that appear in $\Gamma\{\Pi'\}$:

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{d_2'}{\Gamma\{\Pi'\}\{\!\{\varphi\}\!\}_{n-1} \succ \psi}}{\Gamma\{\Pi'\}\{\!\{\Delta\}\!\}_{n-1} \succ \psi}\,\mathsf{mix} \tag{4.4}$$

Observe that, as $h(d_2') < h(d_2)$, $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma\{\Pi'\}\{\!\{\Delta\}\!\}_{n-1} \succ \psi$ by the IH. We now move upwards along d_1 until the rule decomposing the mix-formula φ is reached, and, depending on the structure of Π', we derive $\Gamma\{\Pi\{\underline{\Delta}\}\}\{\!\{\Delta\}\!\}_n \succ \psi$ by using (one or more times) lower ranked applications of mix, each one possibly followed by applications of other rules. To clarify this point, let us consider the case where $\mathsf{r}_2 = \cdot\mathsf{L}$, $\Pi' := \varphi_1 \circ \varphi_2$, $\varphi := \varphi_1 \cdot \varphi_2$, and d_1 has the following structure:

$$\dfrac{\dfrac{\dfrac{\dfrac{d_{11}}{\Delta_1\{\xi_1\} \succ \varphi_1} \quad \dfrac{d_{12}}{\Delta_2 \succ \varphi_2}}{\Delta_1\{\xi_1\} \circ \Delta_2 \succ \varphi_1 \cdot \varphi_2}\cdot\mathsf{R} \quad \dfrac{\dfrac{d_{13}}{\Delta_1\{\xi_2\} \succ \varphi_1} \quad \dfrac{d_{14}}{\Delta_2 \succ \varphi_2}}{\Delta_1\{\xi_2\} \circ \Delta_2 \succ \varphi_1 \cdot \varphi_2}\cdot\mathsf{R}}{\Delta_1\{\xi_1 \vee \xi_2\} \circ \Delta_2 \succ \varphi_1 \cdot \varphi_2}\vee\mathsf{L}}{\Delta_1\{\xi_1 \vee \xi_2\} \circ \Delta_2' \succ \varphi_1 \cdot \varphi_2}\,\mathsf{r}_1 \tag{4.5}$$

Let us define $\Gamma' := \Gamma\{\varphi_1 \circ \varphi_2\}\{\!\{\Delta_1\{\xi_1 \vee \xi_2\} \circ \Delta_2'\}\!\}_{n-1}$. We can now apply mix and replace φ_1 by $\Delta_1\{\xi_1\}$:

$$\dfrac{\dfrac{d_{11}}{\Delta_1\{\xi_1\} \succ \varphi_1} \quad \dfrac{(4.4)}{\Gamma'\{\underline{\varphi_1} \circ \varphi_2\}\{\!\{\varphi\}\!\}_1 \succ \psi}}{\Gamma'\{\underline{\Delta_1\{\xi_1\}} \circ \varphi_2\}\{\!\{\Delta_1\{\xi_1\}\}\!\}_1 \succ \psi}\,\mathsf{mix} \tag{4.6}$$

Since $\mathrm{cp}(\varphi_1) < \mathrm{cp}(\varphi_1 \cdot \varphi_2)$, we have that $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma'\{\underline{\Delta_1\{\xi_1\}} \circ \varphi_2\}\{\!\{\Delta_1\{\xi_1\}\}\!\}_1 \succ \psi$. Upon defining $\Gamma'' := \Gamma'\{\underline{\Delta_1\{\xi_1\}} \circ \varphi_2\}\{\!\{\Delta_1\{\xi_1\}\}\!\}_1$, we apply mix to the premisses $\Delta_2 \succ \varphi_2$ and $\Gamma''\{\Delta_1\{\xi_1\} \circ \underline{\varphi_2}\}\{\!\{\varphi_2\}\!\}_1 \succ \psi$ and derive $\Gamma''\{\Delta_1\{\xi_1\} \circ \underline{\Delta_2}\}\{\!\{\Delta_2\}\!\}_1 \succ \psi$, which admits a mix-free proof by the IH. By the same procedure, using the derivations d_{13} and d_{14}, it is possible to obtain $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma''\{\Delta_1\{\xi_2\} \circ \underline{\Delta_2}\}\{\!\{\Delta_2\}\!\}_1 \succ \psi$. Finally, by applying $\vee\mathsf{L}$ and then r_1 to these last two sequents, it is not difficult to see that one obtains a proof of the conclusion of (4.3)—for $\mathsf{r}_2 = \cdot\mathsf{L}$, $\Pi' := \varphi_1 \circ \varphi_2$, $\varphi := \varphi_1 \cdot \varphi_2$, and d_1 as in (4.5)—where only lower ranked instances of mix have been applied. The proof for $\mathsf{r}_1/\wedge\mathsf{L}$ is analogous. The cases $\mathsf{r}_1/\Diamond\mathsf{L}$, $\mathsf{r}_1/\Box\mathsf{L}$, and $\mathsf{r}_1/1\mathsf{L}$ are simpler and are left to the reader.

If r_2 is either $\cdot\mathsf{R}$, $\wedge\mathsf{R}$, or $\vee\mathsf{L}$, then it suffices to permute mix with r_1 as illustrated in

the transformation schema below:

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta' \succ \varphi}}{\Delta \succ \varphi}\,\mathsf{r}_1 \qquad \dfrac{\dfrac{d_{21}}{\Gamma' \succ \psi'} \qquad \dfrac{d_{22}}{\Gamma'' \succ \psi''}}{\Gamma\{\!\{\varphi\}\!\}_n \succ \psi}\,\mathsf{r}_2}{\Gamma\{\!\{\Delta\}\!\}_n \succ \psi}\,\mathsf{mix} \quad \dashrightarrow \quad \dfrac{\dfrac{\dfrac{d'_1}{\Delta' \succ \varphi} \qquad \dfrac{\dfrac{d_{21}}{\Gamma' \succ \psi'} \qquad \dfrac{d_{22}}{\Gamma'' \succ \psi''}}{\Gamma\{\!\{\varphi\}\!\}_n \succ \psi}\,\mathsf{r}_2}{\Gamma\{\!\{\Delta'\}\!\}_n \succ \psi}\,\mathsf{mix}}{\Gamma\{\!\{\Delta\}\!\}_n \succ \psi}\,(\mathsf{r}_1)\!\downarrow_n \tag{4.7}$$

Suppose now that $\Delta \succ \varphi$ is obtained via an application of $\vee\mathsf{L}$. This is the only binary rule that does not decompose the mix-formula in the right side of the endsequent of d_1. We start with a derivation

$$\dfrac{\dfrac{\dfrac{d_{11}}{\Delta\{\xi_1\} \succ \varphi} \qquad \dfrac{d_{12}}{\Delta\{\xi_2\} \succ \varphi}}{\Delta\{\xi_1 \vee \xi_2\} \succ \varphi}\,\vee\mathsf{L} \qquad \dfrac{\dfrac{d_2}{\Gamma\{\Pi'\} \succ \psi}}{\Gamma\{\Pi\{\underline{\varphi}\}\}\{\!\{\varphi\}\!\}_n \succ \psi}\,\mathsf{r}_2}{\Gamma\{\Pi\{\underline{\Delta\{\xi_1 \vee \xi_2\}}\}\}\{\!\{\Delta\{\xi_1 \vee \xi_2\}\!\}_n \succ \psi}\,\mathsf{mix} \tag{4.8}$$

We first proceed as in (4.4) and replace the $n-1$ selected occurrences of φ appearing in $\Gamma\{\Pi'\}$. Then we apply r_2 and introduce φ. By the IH, we conclude that $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\kappa} \Gamma\{\Pi\{\varphi\}\}\{\!\{\Delta\{\xi_1 \vee \xi_2\}\!\}_{n-1} \succ \psi$. We now trace back along d_{11} and d_{12} until we reach the two instances of the rule r_φ decomposing φ. Consider, for instance, the derivation schema (4.9). For convenience, and without loss of generality, we assume that r_φ is unary, and that no applications of $\vee\mathsf{L}$ occur in the intermediate derivation steps between $\Delta_{11} \succ \varphi$ and $\Delta\{\xi_1\} \succ \varphi$, or between $\Delta_{12} \succ \varphi$ and $\Delta\{\xi_2\} \succ \varphi$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{d_{11}}{\Delta'_{11} \succ \varphi'_{11}}\,\mathsf{r}_\varphi}{\Delta_{11} \succ \varphi}}{\begin{array}{c}\text{some}\\ \text{derivation}\\ \text{steps}\end{array}}}{\Delta\{\xi_1\} \succ \varphi} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{d_{12}}{\Delta'_{12} \succ \varphi'_{12}}\,\mathsf{r}_\varphi}{\Delta_{12} \succ \varphi}}{\begin{array}{c}\text{some}\\ \text{derivation}\\ \text{steps}\end{array}}}{\Delta\{\xi_2\} \succ \varphi}}{\Delta\{\xi_1 \vee \xi_2\} \succ \varphi}\,\vee\mathsf{L} \qquad \dfrac{d'_2}{\Gamma\{\Pi'\}\{\!\{\varphi\}\!\}_{n-1} \succ \psi}\,\mathsf{mix}}{\Gamma\{\Pi'\}\{\!\{\Delta\{\xi_1 \vee \xi_2\}\!\}_{n-1} \succ \psi}\,\mathsf{r}_2}{\Gamma\{\Pi\{\varphi\}\}\{\!\{\Delta\{\xi_1 \vee \xi_2\}\!\}_{n-1} \succ \psi} \tag{4.9}$$

Let us call d'_{11} and d'_{12} the subderivations of d_{11} and d_{12} that end with the sequents $\Delta_{11} \succ \varphi$ and $\Delta_{12} \succ \varphi$, respectively. Setting $\Gamma' := \Gamma\{\Pi\{\varphi\}\}\{\!\{\Delta\{\xi_1 \vee \xi_2\}\!\}_{n-1}$, we construct

the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{d'_{11}}{\dfrac{\Delta'_{11} \succ \varphi'_{11}}{\Delta_{11} \succ \varphi}\,\mathsf{r}_{11}} \quad \dfrac{(4.9)}{\Gamma'\{\Pi\{\underline{\varphi}\}\}\{\!|\varphi|\!\}_1 \succ \psi}}{\Gamma'\{\Pi\{\underline{\Delta_{11}}\}\}\{\!|\Delta_{11}|\!\}_1 \succ \psi}\,\mathsf{mix}}{\begin{array}{c}\text{some}\\ \text{derivation}\\ \text{steps}\end{array}}}{\Gamma'\{\Pi\{\Delta\{\xi_1\}\}\} \succ \psi} \quad \dfrac{\dfrac{\dfrac{\dfrac{d'_{12}}{\dfrac{\Delta'_{12} \succ \varphi'_{12}}{\Delta_{12} \succ \varphi}\,\mathsf{r}_{12}} \quad \dfrac{(4.9)}{\Gamma'\{\Pi\{\underline{\varphi}\}\}\{\!|\varphi|\!\}_1 \succ \psi}}{\Gamma'\{\Pi\{\underline{\Delta_{12}}\}\}\{\!|\Delta_{12}|\!\}_1 \succ \psi}\,\mathsf{mix}}{\begin{array}{c}\text{some}\\ \text{derivation}\\ \text{steps}\end{array}}}{\Gamma'\{\Pi\{\Delta\{\xi_2\}\}\} \succ \psi}}{\Gamma'\{\Pi\{\Delta\{\xi_1 \vee \xi_2\}\}\} \succ \psi}\,\vee\mathsf{L}$$

Note that $p(d'_{11}) = p(d'_{12}) = 0$, whereas in (4.8) we have $p(d_1) = 1$; hence, the IH applies yielding $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma'\{\Pi\{\underline{\Delta_{11}}\}\}\{\!|\Delta_{11}|\!\}_1 \succ \psi$ and $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma'\{\Pi\{\underline{\Delta_{12}}\}\}\{\!|\Delta_{12}|\!\}_1 \succ \psi$. Therefore $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_{\ltimes}} \Gamma'\{\Pi\{\Delta\{\xi_1 \vee \xi_2\}\}\} \succ \psi$. It is immediate to check that the latter sequent is precisely the conclusion of (4.8). We leave to the reader the cases in which $\mathsf{r}_1 = \vee\mathsf{L}$ and r_2 is a binary rule: the proofs for the $\vee\mathsf{L}/\wedge\mathsf{R}$ and $\vee\mathsf{L}/\cdot\,\mathsf{R}$ combinations proceed similarly to (4.7), the only difference being that the IH must be applied twice. As for the $\vee\mathsf{L}/\vee\mathsf{L}$ case, one must first replace the $n-1$ occurrences of the mix-formula on the left sides of the two premisses of $\vee\mathsf{L}$ in the last step of d_2; the proof then proceeds as above.

If a new occurrence of the mix-formula is introduced on the right side of the endsequent of d_1, and nothing happens on the left side of the endsequent of d_2, the proof is straightforward as it suffices to permute mix with (possibly iterated applications) of r_1.

As for (ii), we restrict our attention to cases involving logical rules. Instances where d_2 ends with an application of $\sqcap\mathsf{W}_1$ are treated as in (4.2).

Let us consider the following instance of mix:

$$\dfrac{\dfrac{\dfrac{d_{11}}{\Delta_1 \succ \varphi} \quad \dfrac{d_{12}}{\Delta_2 \succ \psi}}{\Delta_1 \circ \Delta_2 \succ \varphi \cdot \psi}\cdot\mathsf{R} \quad \dfrac{\dfrac{d_2}{\Gamma\{\varphi \circ \psi\} \succ \chi}}{\Gamma\{\underline{\varphi \cdot \psi}\}\{\!|\varphi \cdot \psi|\!\}_n \succ \chi}\cdot\mathsf{L}}{\Gamma\{\underline{\Delta_1 \circ \Delta_2}\}\{\!|\Delta_1 \circ \Delta_2|\!\}_n \succ \chi}\,\mathsf{mix} \tag{4.10}$$

In order to apply the IH, just like in some of the previous cases, we first replace possible occurrences of $\varphi \cdot \psi$ that appear in $\Gamma\{\varphi \circ \psi\}$:

$$\dfrac{\dfrac{\dfrac{d_{11}}{\Delta_1 \succ \varphi} \quad \dfrac{d_{12}}{\Delta_2 \succ \psi}}{\Delta_1 \circ \Delta_2 \succ \varphi \cdot \psi}\cdot\mathsf{R} \quad \dfrac{d'_2}{\Gamma\{\varphi \circ \psi\}\{\!|\varphi \cdot \psi|\!\}_{n-1} \succ \chi}}{\Gamma\{\varphi \circ \psi\}\{\!|\Delta_1 \circ \Delta_2|\!\}_{n-1} \succ \chi}\,\mathsf{mix} \tag{4.11}$$

Since $h(d_2') < h(d_2)$, by the IH $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma\{\varphi \circ \psi\}\{\!\{\Delta_1 \circ \Delta_2\}\!\}_{n-1} \succ \chi$. We now proceed with the replacement of φ by Δ_1 and ψ by Δ_2 in $\varphi \circ \psi$. Let us define $\Gamma' := \Gamma\{\varphi \circ \psi\}\{\!\{\Delta_1 \circ \Delta_2\}\!\}_{n-1}$. We have:

$$\dfrac{\dfrac{d'_{11}}{\Delta_1 \succ \varphi} \qquad \dfrac{(4.11)}{\Gamma'\{\underline{\varphi} \circ \psi\}\{\!\{\varphi\}\!\}_1 \succ \chi}}{\Gamma'\{\underline{\Delta_1} \circ \psi\}\{\!\{\Delta_1\}\!\}_1 \succ \chi}\,\mathsf{mix} \tag{4.12}$$

where $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma'\{\underline{\Delta_1} \circ \psi\}\{\!\{\Delta_1\}\!\}_1 \succ \chi$, since $\mathrm{cp}(\varphi) < \mathrm{cp}(\varphi \cdot \psi)$ and (4.11) applies a lower ranked instance of mix. Let us now define $\Gamma'' := \Gamma'\{\underline{\Delta_1} \circ \psi\}\{\!\{\Delta_1\}\!\}_m$. We replace ψ by Δ_2:

$$\dfrac{\dfrac{d'_{12}}{\Delta_2 \succ \psi} \qquad \dfrac{(4.12)}{\Gamma''\{\Delta_1 \circ \underline{\psi}\}\{\!\{\psi\}\!\}_1 \succ \chi}}{\Gamma''\{\Delta_1 \circ \underline{\Delta_2}\}\{\!\{\Delta_2\}\!\}_1 \succ \chi}\,\mathsf{mix} \tag{4.13}$$

As before, since $\mathrm{cp}(\psi) < \mathrm{cp}(\varphi \cdot \psi)$ and (4.12) applies a lower ranked instance of mix, we have $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma''\{\Delta_1 \circ \underline{\Delta_2}\}\{\!\{\Delta_2\}\!\}_1 \succ \chi$. It is immediate that $\Gamma''\{\Delta_1 \circ \underline{\Delta_2}\}\{\!\{\Delta_2\}\!\}_1$ coincides with $\Gamma\{\underline{\Delta_1 \circ \Delta_2}\}\{\!\{\Delta_1 \circ \Delta_2\}\!\}_n$ in (4.10), whence $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma\{\underline{\Delta_1 \circ \Delta_2}\}\{\!\{\Delta_1 \circ \Delta_2\}\!\}_n \succ \chi$. The treatment of the case $\wedge\mathsf{R}/\wedge\mathsf{L}$ is similar. Let us now consider the derivation:

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta \succ \varphi}}{\Delta \succ \varphi \vee \psi}\vee\mathsf{R}_1 \qquad \dfrac{\dfrac{d_{21}}{\Gamma\{\varphi\} \succ \chi} \qquad \dfrac{d_{22}}{\Gamma\{\psi\} \succ \chi}}{\Gamma\{\underline{\varphi \vee \psi}\}\{\!\{\varphi \vee \psi\}\!\}_n \succ \chi}\vee\mathsf{L}}{\Gamma\{\underline{\Delta}\}\{\!\{\Delta\}\!\}_n \succ \chi}\,\mathsf{mix} \tag{4.14}$$

Here too, we begin by replacing selected occurrences of the mix-formula that appear in $\Gamma\{\varphi\}$ (or, equivalently, in $\Gamma\{\psi\}$):

$$\dfrac{\dfrac{\dfrac{d_1}{\Delta \succ \varphi}}{\Delta \succ \varphi \vee \psi}\vee\mathsf{R}_1 \qquad \dfrac{d'_{21}}{\Gamma\{\varphi\}\{\!\{\varphi \vee \psi\}\!\}_{n-1} \succ \chi}}{\Gamma\{\varphi\}\{\!\{\Delta\}\!\}_{n-1} \succ \chi}\,\mathsf{mix} \tag{4.15}$$

By the IH, $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma\{\varphi\}\{\!\{\Delta\}\!\}_{n-1} \succ \chi$. Upon defining $\Gamma' := \Gamma\{\varphi\}\{\!\{\Delta\}\!\}_{n-1}$, we construct the derivation:

$$\dfrac{\dfrac{d'_1}{\Delta \succ \varphi} \qquad \dfrac{(4.15)}{\Gamma'\{\underline{\varphi}\}\{\!\{\varphi\}\!\}_1 \succ \chi}}{\Gamma'\{\underline{\Delta}\}\{\!\{\Delta\}\!\}_1 \succ \chi}\,\mathsf{mix}$$

So we have $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma'\{\underline{\Delta}\}\{\!\{\Delta\}\!\}_1 \succ \chi$, and since $\Gamma'\{\underline{\Delta}\}\{\!\{\Delta\}\!\}_1$ coincides with $\Gamma\{\underline{\Delta}\}\{\!\{\Delta\}\!\}_n$ in (4.14), we obtain $\vdash^{\mathrm{mf}}_{\mathsf{LMC}_\ltimes} \Gamma\{\underline{\Delta}\}\{\!\{\Delta\}\!\}_n \succ \chi$. The case $\vee\mathsf{R}_2/\vee\mathsf{L}$ follows exactly the same pattern. The proof for the $\Diamond\mathsf{R}/\Diamond\mathsf{L}$ and $\Box\mathsf{R}/\Box\mathsf{L}$ combinations follow by analogous, yet simpler, reasoning, and are thus left to the reader. □

Corollary 4.3.5. *The rule* cut *can be eliminated from* LMC.

Proof. Immediate from Remark 4.3.2 and Theorem 4.3.4. □

Remark 4.3.6. By Corollary 4.3.5, we immediately obtain that the equations $\Diamond 1 \approx 1$, $\Box x \cdot \Box 1 \approx \Box x$, and $\Box 1 \cdot \Box x \approx \Box x$ do not admit derivable counterparts in our system. Indeed, one readily checks that there are no cut-free derivations of the sequents $\Diamond 1 \succ 1$, $\Box x \succ \Box x \cdot \Box 1$, and $\Box x \succ \Box 1 \cdot \Box x$, and hence they are not derivable in LMC. This clearly highlights the extent to which closure ℓ-monoids approximate algebras of the form $\wp(\mathbf{F}_\Sigma)_+$, in which, by contrast, the above equations are satisfied.

4.4 Decidability

Corollary 4.3.5 guarantees that LMC enjoys the *subformula property*, that is, every formula occurring in the sequents of any cut-free derivation d in LMC is a subformula of some formula occurring in the endsequent of d. However, the presence of rules such as $\sqcap$C and T prevents the system from enjoying a corresponding *substructure property*, in the sense that there exist cut-free derivations in which certain structural terms occurring in the sequents are not subterms of the structural term appearing on the left side of the endsequent. More precisely, from this perspective many of the rules of LMC are non-analytic: this is the case for $\circ$A, $\sqcap$A, $\sqcap$E, $\sqcap$C, K, T, $\cdot$L, $\wedge$L, $\Diamond$L, $\Box$L, and 1L. This situation is common in display and display-like calculi, and implies that cut elimination may not automatically entail the decidability of the proof system [38]. Moreover, by a well-known result due to Kracht [107] (see also [106]), the problem of determining whether a given display calculus is decidable is itself undecidable. Consequently, decidability proofs must be developed on a case-by-case basis, relying on proof strategies that depend on the intrinsic features of the calculus under consideration (see, for instance, [147, 168, 169]).

In this section we provide a decidability proof for LMC, adapting Gentzen's classical argument for LK [76] (our presentation follows Paoli's exposition of Gentzen's proof [135, Section 2.1]). Our proof combines three normalisation procedures for LMC-derivations. The first is, naturally, cut elimination, established in Section 4.3. In addition, a backward proof search procedure for LMC must incorporate mechanisms for controlling backward applications of the rules T and $\sqcap$C, as well as for preventing proof-search loops arising from sequences of inference steps in which rule applications

undo the effects of previous ones. In Subsection 4.4.1 we address the normalisation of proofs with respect to applications of T, while Subsection 4.4.2 adapts Gentzen's classical contraction-control procedure for LK to our logic. We provide the decidability theorem in Subsection 4.4.3.

4.4.1 T-normalisation

We begin by studying how T interacts with the rules of $\mathsf{LMC} - \mathsf{cut}$. In particular, we address the following problem:

> Given a cut-free derivation containing subderivations of the forms
>
> $$\dfrac{\dfrac{\Delta \succ \psi}{\Gamma \succ \varphi}\,\mathsf{r}}{\Gamma' \succ \varphi}\,\mathsf{T} \qquad \dfrac{\dfrac{\Delta \succ \psi \qquad \Pi \succ \chi}{\Gamma \succ \varphi}\,\mathsf{r}}{\Gamma' \succ \varphi}\,\mathsf{T}$$
>
> under which conditions can the application of T can be permuted with the application of r so as to leave $\Gamma' \succ \varphi$ unchanged?

We therefore focus on the upward permutability of applications of T in cut-free derivations, with respect to rule applications occurring immediately above them. In what follows, for r a rule of LMC, we refer to applications of r in derivations as r-*applications*.

Definition 4.4.1. Let d be a cut-free derivation in LMC. A T-application in d is said to be *frozen* if it does not admit upward permutation with the immediately preceding rule application.

Structural rules We first consider the case of the structural rules $\circ\mathsf{A}$, $\circ\epsilon$, $\epsilon\circ$, $\sqcap\mathsf{A}$, $\sqcap\mathsf{W}_1$, $\sqcap\mathsf{E}$, $\sqcap\mathsf{C}$, K, and $\mathsf{4}$.

1. Consider an inference pattern of the form

   $$\dfrac{\dfrac{\Gamma\{((\Delta_1 \circ \Delta_2) \circ \Delta_3)\} \succ \varphi}{\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\} \succ \varphi}\,\circ\mathsf{A}}{\Gamma' \succ \varphi}\,\mathsf{T}$$

   Here, the T-application may act on one of the structural subterms Δ_1, Δ_2, or Δ_3, on structural subterms occurring elsewhere within $\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\}$, or on the entire $\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\}$, in the case where it is of the form $\langle\Pi\rangle$, for

some $\Pi \in \mathit{Struct}$. In all these situations, the T-application is never frozen, since it neither interferes with nor depends in any way on the preceding $\circ\mathsf{A}$-application. An analogous argument can be given for the rules $\circ\,\epsilon$, $\epsilon\,\circ$, $\sqcap\mathsf{A}$, $\sqcap\mathsf{E}$, and $\sqcap\mathsf{C}$.

2. We now examine the following derivation pattern:

$$\dfrac{\dfrac{\Gamma\{\Delta_1\} \succ \varphi}{\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi}\,\sqcap\mathsf{W}_1}{\Gamma' \succ \varphi}\,\mathsf{T}$$

If the T-application does not act on the newly introduced term Δ_2, then the permutation is admissible. What happens instead if the T-application modifies Δ_2? Clearly, in this case we are dealing with a frozen occurrence. Nevertheless, observe that in this particular situation the T-application can be eliminated altogether by performing the following transformation:

$$\dfrac{\dfrac{\dfrac{\Gamma\{\Delta_1\} \succ \varphi}{\Gamma\{\Delta_1 \sqcap \Delta_2\{\langle\Xi\rangle\}\} \succ \varphi}\,\sqcap\mathsf{W}_1}{\Gamma\{\Delta_1 \sqcap \Delta_2\{\Xi\}\} \succ \varphi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\Gamma\{\Delta_1\} \succ \varphi}{\Gamma\{\Delta_1 \sqcap \Delta_2\{\Xi\}\} \succ \varphi}\,\sqcap\mathsf{W}_1 \qquad \text{(RE1)}$$

More generally, in an LMC-derivation, we call a $\sqcap\mathsf{W}_1$-application T-*redundant* if it introduces a structural term containing a structural diamond that is subsequently removed by a T-application.

3. It remains to account for K and $\mathsf{4}$. Let us start from the following inference pattern:

$$\dfrac{\dfrac{\Gamma\{\langle\Delta_1\rangle \circ \langle\Delta_2\rangle\} \succ \varphi}{\Gamma\{\langle\Delta_1 \circ \Delta_2\rangle\} \succ \varphi}\,\mathsf{K}}{\Gamma' \succ \varphi}\,\mathsf{T}$$

The T-application is frozen whenever $\langle\Delta_1 \circ \Delta_2\rangle$ does not occur within the scope of a structural diamond and the T-application transforms $\Gamma\{\langle\Delta_1 \circ \Delta_2\rangle\}$ into $\Gamma\{\Delta_1 \circ \Delta_2\}$. Otherwise, the permutation is allowed, including the cases in which the T-application acts on Δ_1 or Δ_2.

As for the rule $\mathsf{4}$, we consider inference patterns of the following form:

$$\dfrac{\dfrac{\Gamma\{\langle\Delta\rangle\} \succ \varphi}{\Gamma\{\langle\langle\Delta\rangle\rangle\} \succ \varphi}\,\mathsf{4}}{\Gamma' \succ \varphi}\,\mathsf{T}$$

If the T-application removes one of the two structural diamonds surrounding Δ, we call the corresponding $\mathsf{4}$-application T-*redundant* and perform the following transformation:

$$\dfrac{\dfrac{\dfrac{\Xi \succ \psi}{\Gamma\{\langle\Delta\rangle\} \succ \varphi}\,\mathsf{r}}{\Gamma\{\langle\langle\Delta\rangle\rangle\} \succ \varphi}\,\mathsf{4}}{\Gamma\{\langle\Delta\rangle\} \succ \varphi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\Xi \succ \psi}{\Gamma\{\langle\Delta\rangle\} \succ \varphi}\,\mathsf{r} \tag{RE2}$$

If, instead, when traversing the syntactic tree of $\langle\langle\Delta\rangle\rangle$ top-down, the T-application operates within the first subterm Δ' of $\langle\langle\Delta\rangle\rangle$ that is not of the form $\langle\Pi\rangle$ for any $\Pi \in \mathit{Struct}$, then the permutation is unproblematic. The same holds if the T-application acts elsewhere within Γ (outside $\langle\langle\Delta\rangle\rangle$).

Logical rules We begin by observing that T can never be applied immediately after an instance of $\mathsf{1R}$, and that applying T—when possible—after instances of $\bot\mathsf{L}$ and $\top\mathsf{R}$ yields, in any case, further instances of $\bot\mathsf{L}$ and $\top\mathsf{R}$, respectively.

4. Let r be a right-introduction rule that does not modify the left side(s) of its premiss(es). When a T-application occurs immediately after an r-application, the permutation is always allowed. Consider, for instance, the case $\mathsf{r} = \wedge\mathsf{R}$. We have the transformation:

$$\dfrac{\dfrac{\Gamma \succ \varphi \qquad \Gamma \succ \psi}{\Gamma \succ \varphi \wedge \psi}\,\wedge\mathsf{R}}{\Gamma' \succ \varphi \wedge \psi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\dfrac{\Gamma \succ \varphi}{\Gamma' \succ \varphi}\,\mathsf{T} \qquad \dfrac{\Gamma \succ \psi}{\Gamma' \succ \psi}\,\mathsf{T}}{\Gamma' \succ \varphi \wedge \psi}\,\wedge\mathsf{R}$$

   The cases of $\vee\mathsf{R}_1$, $\vee\mathsf{R}_2$, $\cdot\mathsf{1}$, and $\mathsf{1}\cdot$ are analogous.

5. Let r now be a right-introduction rule that modifies the left side(s) of its premiss(es). The only such rule that is binary is $\cdot\mathsf{R}$. Without loss of generality, we start from an inference pattern of the following form:

$$\dfrac{\dfrac{\Gamma_1 \succ \varphi \qquad \Gamma_2 \succ \psi}{\Gamma_1 \circ \Gamma_2 \succ \varphi \cdot \psi}\,\cdot\mathsf{R}}{\Gamma_1' \circ \Gamma_2 \succ \varphi \cdot \psi}\,\mathsf{T}$$

   that is, the T-application removes a structural diamond occurring in Γ_1. The permutation is then carried out by commuting T solely with respect to the left premiss of $\cdot\mathsf{R}$:

$$\dfrac{\dfrac{\Gamma_1 \succ \varphi}{\Gamma_1' \succ \varphi}\,\mathsf{T} \qquad \Gamma_2 \succ \psi}{\Gamma_1' \circ \Gamma_2 \succ \varphi \cdot \psi}\,\cdot\mathsf{R}$$

The symmetric case, in which the T-application acts on Γ_2, is treated analogously.

If r is unary, then either $\mathsf{r} = \Diamond\mathsf{R}$ or $\mathsf{r} = \Box\mathsf{R}$. In the latter case, it is clear that a T-application occurring immediately after a $\Box\mathsf{R}$-application can never be frozen, as—whenever applied in this order—the two rules do not interfere with each other. The following transformation is thus allowed:

$$\dfrac{\dfrac{\langle\Gamma\rangle \succ \varphi}{\Gamma \succ \Box\varphi}\,\Box\mathsf{R}}{\Gamma' \succ \Box\varphi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\dfrac{\langle\Gamma\rangle \succ \varphi}{\langle\Gamma'\rangle \succ \varphi}\,\mathsf{T}}{\Gamma \succ \Box\varphi}\,\Box\mathsf{R}$$

Let now $\mathsf{r} = \Diamond\mathsf{R}$. Consider the following inference pattern:

$$\dfrac{\dfrac{\Gamma \succ \varphi}{\langle\Gamma\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}}{\Delta \succ \Diamond\varphi}\,\mathsf{T}$$

If $\Delta = \Gamma$ (that is, if the T-application removes the structural diamond introduced by the $\Diamond\mathsf{R}$-application), we distinguish the following two cases:

- $\Gamma = \langle\Pi\rangle$ for some $\Pi \in \mathit{Struct}$. The permutation can be carried out via the following transformation:

$$\dfrac{\dfrac{\langle\Pi\rangle \succ \varphi}{\langle\langle\Pi\rangle\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}}{\langle\Pi\rangle \succ \Diamond\varphi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\dfrac{\langle\Pi\rangle \succ \varphi}{\Pi \succ \varphi}\,\mathsf{T}}{\langle\Pi\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}$$

- $\Gamma \neq \langle\Pi\rangle$ for every $\Pi \in \mathit{Struct}$. The T-application is frozen.

If $\Delta \neq \Gamma$, then permutation is always permitted. In particular, we have two cases:

- $\Gamma \neq \langle\Pi\rangle$ for every $\Pi \in \mathit{Struct}$ and the T-application removes a structural diamond within Γ. We have the transformation:

$$\dfrac{\dfrac{\Gamma \succ \varphi}{\langle\Gamma\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}}{\langle\Gamma'\rangle \succ \Diamond\varphi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\dfrac{\Gamma \succ \varphi}{\Gamma' \succ \varphi}\,\mathsf{T}}{\langle\Gamma'\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}$$

- $\Gamma = \langle\Pi\rangle$ for some $\Pi \in \mathit{Struct}$ and the T-application removes a structural diamond within Π:

$$\dfrac{\dfrac{\langle\Pi\rangle \succ \varphi}{\langle\langle\Pi\rangle\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}}{\langle\langle\Pi'\rangle\rangle \succ \Diamond\varphi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\dfrac{\langle\Pi\rangle \succ \varphi}{\langle\Pi'\rangle \succ \varphi}\,\mathsf{T}}{\langle\langle\Pi'\rangle\rangle \succ \Diamond\varphi}\,\Diamond\mathsf{R}$$

6. We conclude with the left-introduction rules. It is immediate that permutation is unproblematic with respect to $\cdot\mathsf{L}$, $\wedge\mathsf{L}$, $\vee\mathsf{L}$, $\Diamond\mathsf{L}$, and $1\mathsf{L}$. Indeed, when a T-application occurs immediately after an application of any of these rules, it can never be frozen, even in the case where the introduced formula occurs within the structural diamond eliminated by T. For instance, consider the following transformation:

$$\dfrac{\dfrac{\Gamma\{\langle\varphi\rangle\} \succ \chi \qquad \Gamma\{\langle\psi\rangle\} \succ \chi}{\Gamma\{\langle\varphi \vee \psi\rangle\} \succ \chi}\,\vee\mathsf{L}}{\Gamma\{\varphi \vee \psi\} \succ \chi}\,\mathsf{T} \quad \dashrightarrow \quad \dfrac{\dfrac{\Gamma\{\langle\varphi\rangle\} \succ \chi}{\Gamma\{\varphi\} \succ \chi}\,\mathsf{T} \qquad \dfrac{\Gamma\{\langle\psi\rangle\} \succ \chi}{\Gamma\{\psi\} \succ \chi}\,\mathsf{T}}{\Gamma\{\varphi \vee \psi\} \succ \chi}$$

The only left-introduction rule for which permutation with T requires a side condition is $\boxplus\mathsf{L}$, where the formula $\boxplus\varphi$ is introduced together with a surrounding structural diamond:

$$\dfrac{\dfrac{\Gamma\{\varphi\} \succ \psi}{\Gamma\{\langle\boxplus\varphi\rangle\} \succ \psi}\,\boxplus\mathsf{L}}{\Gamma' \succ \psi}\,\mathsf{T}$$

Here, T is frozen if and only if the replacement of φ for $\langle\boxplus\varphi\rangle$ via $\boxplus\mathsf{L}$ is not performed within the scope of any pre-existing structural diamond and T transforms $\Gamma\{\langle\boxplus\varphi\rangle\}$ into $\Gamma\{\boxplus\varphi\}$. Otherwise, the permutation is allowed (details are left to the reader).

In light of the foregoing discussion, we introduce a notion of normal form for cut-free proofs in LMC, formulated in terms of the non-permutability of T-applications.

Definition 4.4.2. Let d be a cut-free derivation in LMC. We say that d is in T-*normal form* if every T-application occurring in d is frozen and d contains neither T-redundant $\sqcap\mathsf{W}_1$-applications nor detours induced by T-redundant $\mathsf{4}$-applications.

Lemma 4.4.3. *If* $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$*, then* $\Gamma \succ \varphi$ *admits a derivation in* T*-normal form.*

Proof. Suppose that $\Gamma \succ \varphi$ is provable in LMC. By Corollary 4.3.5, it admits a cut-free derivation d. If d contains no T-applications, the claim holds trivially. Otherwise, d can be transformed into a derivation in T-normal form by maximally permuting all T-applications upwards and, by applying the transformations in (RE1) and (RE2), progressively eliminating any frozen T-applications that result from T-redundant $\sqcap\mathsf{W}_1$-applications, as well as any detours arising from T-redundant $\mathsf{4}$-applications. □

Lemma 4.4.4. *Let d be a derivation in T-normal form. The number of T-applications occurring in d is bounded above by the total number of $\Diamond\mathsf{R}$- and $\Box\mathsf{L}$-applications occurring in d.*

Proof. Left to the reader. □

4.4.2 Contraction control

Definition 4.4.5. Let $\Gamma \in \mathit{Struct}$. The set $T(\Gamma, \sqcap)$ is defined by the grammar

$$\Pi ::= \Gamma \mid \Pi_1 \sqcap \Pi_2,$$

where $\Pi_1, \Pi_2 \in T(\Gamma, \sqcap)$. Elements of $T(\Gamma, \sqcap)$ are called $(\Gamma, \sqcap)$-*terms*. The Γ-*multiplicity* $\mu_\Gamma(\Pi)$ of a $(\Gamma, \sqcap)$-term Π is defined by induction on the structure of Π as follows:

$$\mu_\Gamma(\Pi) = \begin{cases} 1 & \text{if } \Pi = \Gamma, \\ \mu_\Gamma(\Pi_1) + \mu_\Gamma(\Pi_2) & \text{if } \Pi = \Pi_1 \sqcap \Pi_2. \end{cases}$$

Definition 4.4.6. For $n \in \mathbb{N}$, a structural term $\Gamma \in \mathit{Struct}$ is *n-reduced* if, for every $\Delta \prec \Gamma$, and for every $\Pi \in T(\Delta, \sqcap)$ such that $\Pi \preceq \Gamma$, we have $\mu_\Delta(\Pi) \leq n$.

Definition 4.4.6 involves two levels of universal quantification, which require careful unfolding. To determine whether a structural term Γ is n-reduced, proceed as follows:

1. Identify all $\Delta \in \mathit{Struct}$ such that $\Delta \prec \Gamma$.
2. For each such Δ, compute the Δ-multiplicity of every $(\Delta, \sqcap)$-term occurring in Γ, that is, the number of occurrences of Δ in each $(\Delta, \sqcap)$-term that is a proper subterm of Γ. Define $M_\Delta := \max\{\mu_\Delta(\Pi) \mid \Pi \in T(\Delta, \sqcap) \text{ and } \Pi \prec \Gamma\}$.
3. Compute $M := \max\{M_\Delta \mid \Delta \prec \Gamma\}$. If $M \leq n$, then Γ is n-reduced; otherwise, it is not.

For the reader's benefit, we provide a concrete example.

Example 4.4.7. Let $\Gamma := ((\varphi \sqcap \varphi) \sqcap \varphi) \sqcap (\psi \sqcap (\varphi \sqcap \varphi))$. The proper subterms of Γ are:

$$\varphi \qquad \psi \qquad \varphi \sqcap \varphi \qquad (\varphi \sqcap \varphi) \sqcap \varphi \qquad \psi \sqcap (\varphi \sqcap \varphi).$$

For $\Delta \prec \Gamma$, we can list all $(\Delta, \sqcap)$-terms occurring in Γ, together with their Δ-multiplicities. For instance, in the case $\Delta = \varphi$ we have:

$$\varphi,\ 1 \qquad \varphi \sqcap \varphi,\ 2 \qquad (\varphi \sqcap \varphi) \sqcap \varphi,\ 3.$$

Thus, $M_\varphi = 3$. We repeat this procedure for the remaining terms. The resulting data are summarised in Table 4.2.

Δ	$(\Delta, \sqcap)$-terms occurring in Γ_1	M_Δ
φ	φ, $\varphi \sqcap \varphi$, $(\varphi \sqcap \varphi) \sqcap \varphi$	3
ψ	ψ	1
$\varphi \sqcap \varphi$	$\varphi \sqcap \varphi$	1
$(\varphi \sqcap \varphi) \sqcap \varphi$	$(\varphi \sqcap \varphi) \sqcap \varphi$	1
$\psi \sqcap (\varphi \sqcap \varphi)$	$\psi \sqcap (\varphi \sqcap \varphi)$	1

Table 4.2: Multiplicity maxima for elements of $\bigcup_{\Delta \prec \Gamma} T(\Delta, \sqcap)$ occurring in Γ.

We conclude that Γ is a 3-reduced term.

We now define a translation to convert each structural term into its 1-reduced counterpart (which is unique by Definition 4.4.6).

Definition 4.4.8. Let $^\bullet$: *Struct* $\to$ *Struct* be the function such that:

1. $\Gamma^\bullet = \Gamma$, for all $\Gamma \in Fm \cup \{\epsilon\}$;
2. $^\bullet$ behaves homomorphically with respect to $\circ$ and $\langle - \rangle$;
3. If $\Gamma = \Delta_1 \sqcap \Delta_2$ and $\Gamma \notin T(\Xi, \sqcap)$ for any $\Xi \in$ *Struct*, then $\Gamma^\bullet = \Delta_1^\bullet \sqcap \Delta_2^\bullet$;
4. For all $\Gamma \in$ *Struct* and all $\Pi \in T(\Gamma, \sqcap)$, $\Pi^\bullet = \Gamma^\bullet$.

Proposition 4.4.9. *The following conditions hold:*

1. *For all* $\Gamma \in$ *Struct*, $\Gamma^\bullet$ *is* 1*-reduced;*
2. $^\bullet$ *is idempotent;*
3. *A structural term is a fixed point of* $^\bullet$ *iff it is* 1*-reduced.*

Proof. The proofs of items 1 and 2 are left to the reader. We only show item 3. Let $\Gamma \in \mathit{Struct}$. If Γ is not 1-reduced, then there exist $\Delta, \Pi \prec \Gamma$ such that $\Pi \in T(\Delta, \sqcap)$ and $\mu_{\Delta}(\Pi) > 1$. It follows that $\Gamma \neq \Gamma^{\bullet}$, since in $\Gamma^{\bullet}$, Π is reduced to $\Delta^{\bullet}$. This establishes the left-to-right implication. Conversely, suppose that Γ is 1-reduced. In this case, $^{\bullet}$ commutes with all structural connectives occurring in Γ without performing any reduction. It follows that $\Gamma = \Gamma^{\bullet}$. □

Definition 4.4.10. A sequent $\Delta \succ \varphi$ is a *contraction* of a sequent $\Gamma \succ \varphi$ if it is derivable in LMC from $\Gamma \succ \varphi$ by a finite, possibly empty sequence of applications of the rule $\sqcap\mathsf{C}$. We say that $\Delta \succ \varphi$ is an *n-reduced contraction* (*n-reduction*, for short) of $\Gamma \succ \varphi$ if it is a *contraction* of $\Gamma \succ \varphi$ and Δ is n-reduced.

It is immediate that every sequent of the form $\Gamma^{\bullet} \succ \varphi$ is the 1-reduction of $\Gamma \succ \varphi$.

Proposition 4.4.11. *Let $\Delta \succ \varphi$ be any n-reduced contraction of an LMC-sequent $\Gamma \succ \varphi$. Then $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$ iff $\vdash_{\mathsf{LMC}} \Delta \succ \varphi$*

Proof. The proof is straightforward. If $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$, then $\Delta \succ \varphi$ can be derived from $\Gamma \succ \varphi$ by a possibly empty sequence of iterated applications of $\sqcap\mathsf{C}$. Conversely, if $\vdash_{\mathsf{LMC}} \Delta \succ \varphi$, then $\Gamma \succ \varphi$ can be obtained from $\Delta \succ \varphi$ by some (possibly zero) successive applications of $\sqcap\mathsf{W}_1$ and/or $\sqcap\mathsf{W}_2$. □

Corollary 4.4.12. *$\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$ iff $\vdash_{\mathsf{LMC}} \Gamma^{\bullet} \succ \varphi$, for any LMC-sequent $\Gamma \succ \varphi$.*

Let d be an LMC-derivation. We denote by $\beta(d)$ the set of branches of d, understood as maximal paths in the derivation tree from the end-sequent of d to an instance of an initial sequent. For $\mathbf{b} \in \beta(d)$, we write $|\mathbf{b}|$ for the length of $\mathbf{b}$.

Definition 4.4.13. An LMC-derivation d is *n-reduced* if, for every $\mathbf{b} \in \beta(d)$, every sequent occurring in $\mathbf{b}$ is n-reduced.

Lemma 4.4.14. *Let a sequent $\Gamma \succ \varphi$ be provable in LMC. Then there exists a cut-free, 3-reduced derivation of $\Gamma^{\bullet} \succ \varphi$.*

Proof. By virtue of Corollary 4.3.5, we shall consider a cut-free derivation d of $\Gamma \succ \varphi$. The proof proceeds by induction on the height of derivations.

The base case is immediate. For the inductive step, we prove the Lemma with respect to the following classes of rules.

(A) Structural rules for $\circ$, $\langle - \rangle$, and ϵ. Let $\Gamma \succ \varphi$ be a sequent of the form $\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\} \succ \varphi$, and suppose it is derived as follows:

$$\dfrac{\dfrac{d}{\Gamma\{((\Delta_1 \circ \Delta_2) \circ \Delta_3)\} \succ \varphi}}{\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\} \succ \varphi} \circ \mathsf{A} \tag{4.16}$$

We denote by $\underline{((\Delta_1 \circ \Delta_2) \circ \Delta_3)}$ the occurrence of $((\Delta_1 \circ \Delta_2) \circ \Delta_3)$ distinguished in the premiss of $\circ\mathsf{A}$. By the induction hypothesis (IH), there exists a cut-free, 3-reduced derivation

$$\dfrac{d'}{[\Gamma\{((\Delta_1 \circ \Delta_2) \circ \Delta_3)\}]^\bullet \succ \varphi} \tag{4.17}$$

Let us define the shorthands $\Gamma_p := \Gamma\{((\Delta_1 \circ \Delta_2) \circ \Delta_3)\}$ and $\Gamma' := \Gamma_p^\bullet$. The occurrence $\underline{((\Delta_1 \circ \Delta_2) \circ \Delta_3)}$ may appear in Γ_p in three guises:

(a) As an immediate subterm of a structural term Ξ whose main operator is not $\sqcap$, where Ξ possibly occurs within a larger $(\Pi, \sqcap)$-term, with $\Xi \preceq \Pi$;

(b) As an immediate subterm of a structural term Ξ whose main connective is $\sqcap$, but whose other immediate subterm is not a $((\Delta_1 \circ \Delta_2) \circ \Delta_3, \sqcap)$-term, where, again, Ξ possibly occurs within a larger $(\Pi, \sqcap)$-term, with $\Xi \preceq \Pi$;

(c) As a subterm of a $((\Delta_1 \circ \Delta_2) \circ \Delta_3, \sqcap)$-term Ξ, itself possibly occurring within a larger $(\Pi, \sqcap)$-term, with $((\Delta_1 \circ \Delta_2) \circ \Delta_3) \preceq \Pi$.

In all cases, Γ' will contain an occurrence of the term

$$((\Delta_1 \circ \Delta_2) \circ \Delta_3)^\bullet = (\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet,$$

appearing within $\Xi^\bullet$, where Ξ is as in (a), (b), or (c) above, depending on the shape of Γ_p. The inference steps leading to the 1-reduction of the endsequent of (4.16) must be carried out with respect to such occurrence of $(\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet$, thus we rewrite (4.17) as

$$\dfrac{d'}{\Gamma'\{((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet)\} \succ \varphi}$$

Example 4.4.15. The identification of the aforementioned occurrence of $(\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet$ in Γ' is a crucial step in our proof. We therefore present three examples corresponding to cases (a), (b), and (c), in which we single out (by underlining) the relevant occurrence of $(\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet$.

(a) If $\Gamma_p := \Delta \circ \langle \underline{((\Delta_1 \circ \Delta_2) \circ \Delta_3)} \rangle$, then $\Gamma' := \Delta^\bullet \circ \langle \underline{((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet)} \rangle$

(b) If $\Gamma_p := ((\Delta_1 \circ \Delta_2) \circ \Delta_3) \sqcap [(\underline{((\Delta_1 \circ \Delta_2) \circ \Delta_3)} \sqcap \Pi) \sqcap (((\Delta_1 \circ \Delta_2) \circ \Delta_3) \sqcap \Pi)]$, then $\Gamma' := ((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet) \sqcap (\underline{((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet)} \sqcap \Pi^\bullet)$

(c) If $\Gamma_p := \underline{((\Delta_1 \circ \Delta_2) \circ \Delta_3)} \sqcap ((\Delta_1 \circ \Delta_2) \circ \Delta_3)$, then $\Gamma' := \underline{((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet)}$

We now continue with the proof. We extend (4.17) so as to obtain:

$$\dfrac{\dfrac{d'}{\Gamma'\{((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet)\} \succ \varphi}{\Gamma'\{(\Delta_1^\bullet \circ (\Delta_2^\bullet \circ \Delta_3^\bullet))\} \succ \varphi} \circ \mathsf{A} \tag{4.18}$$

Within the endsequent of (4.18), the distinguished occurrence of $(\Delta_1^\bullet \circ (\Delta_2^\bullet \circ \Delta_3^\bullet))$ resulting by the $\circ\mathsf{A}$-application, satisfies one of the following conditions:

- It is an immediate subterm of a structural term whose main operator is not $\sqcap$;
- It is an immediate subterm of a structural term whose main operator is $\sqcap$, but whose other immediate subterm is not a copy of $(\Delta_1^\bullet \circ (\Delta_2^\bullet \circ \Delta_3^\bullet))$;
- It is the immediate subterm of a structural term whose main operator is $\sqcap$ and whose other immediate subterm is itself a copy of $(\Delta_1^\bullet \circ (\Delta_2^\bullet \circ \Delta_3^\bullet))$.

Since $\Gamma'\{((\Delta_1^\bullet \circ \Delta_2^\bullet) \circ \Delta_3^\bullet)\}$ is 1-reduced, in the first two cases (4.18) directly yields the 1-reduction of the sequent $\Gamma\{(\Delta_1 \circ (\Delta_2 \circ \Delta_3))\} \succ \varphi$ in (4.16). In the third case, one has to extend (4.18) by a single application of $\sqcap\mathsf{C}$. In all cases, the resulting derivation is 3-reduced.

The rules $\circ\epsilon$, $\epsilon\circ$, K, T, and $\mathsf{4}$ are treated analogously.

(B) Structural rules for $\sqcap$. Now let $\Gamma \succ \varphi$ be the sequent $\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi$, and suppose it is derived as follows:

$$\dfrac{\dfrac{d}{\Gamma\{\Delta_1\} \succ \varphi}}{\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi} \sqcap\mathsf{W}_1 \tag{4.19}$$

By the IH, there exists a 3-reduced derivation

$$\dfrac{d'}{[\Gamma\{\Delta_1\}]^\bullet \succ \varphi} \tag{4.20}$$

Setting $\Gamma_p := \Gamma\{\Delta_1\}$ and $\Gamma' := \Gamma_p^\bullet$, and applying the same line of reasoning used in (A), we conclude that Γ' contains an occurrence of $\Delta_1^\bullet$ relative to which the inference rules leading to the 1-reduction of the endsequent of (4.19) must be applied (details are left to the reader). Hence, (4.20) can be rewritten as

$$\frac{d'}{\Gamma'\{\Delta_1^\bullet\} \succ \varphi} \tag{4.21}$$

If $\Delta_1 \sqcap \Delta_2$ does not belong to $T(\Xi, \sqcap)$ for any $\Xi \in \mathit{Struct}$, then (4.21) can be extended by a $\sqcap\mathsf{W}_1$-application to yield the 1-reduction of $\Gamma\{\Delta_1 \sqcap \Delta_2\} \succ \varphi$. Otherwise, (4.21) itself is the desired derivation.

We now turn to the rule $\sqcap\mathsf{A}$. Consider a derivation of the form

$$\frac{\dfrac{d}{\Gamma\{((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)\} \succ \varphi}}{\Gamma\{(\Delta_1 \sqcap (\Delta_2 \sqcap \Delta_3))\} \succ \varphi}\,\sqcap\mathsf{A} \tag{4.22}$$

We denote by $\underline{((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)}$ the occurrence of $((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)$ distinguished in the premiss of $\sqcap\mathsf{A}$. By the IH, there exists a cut-free, 3-reduced derivation

$$\frac{d'}{[\Gamma\{((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)\}]^\bullet \succ \varphi} \tag{4.23}$$

We set $\Gamma_p := \Gamma\{((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)\}$ and $\Gamma' := \Gamma_p^\bullet$. Once again (details are routine and therefore omitted), Γ' must contain an occurrence of the term $((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)^\bullet$ relative to which the inference rules leading to the 1-reduction of the endsequent of (4.22) must be applied. Accordingly, we rewrite (4.23) as

$$\frac{d'}{\Gamma'\{((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)^\bullet\} \succ \varphi}$$

A cut-free, 3-reduced derivation of $[\Gamma\{(\Delta_1 \sqcap (\Delta_2 \sqcap \Delta_3))\}]^\bullet \succ \varphi$ can now be obtained in one of the following ways:

- If $((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3) \in T(\Xi, \sqcap)$ for some $\Xi \in \mathit{Struct}$, then

  $$((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)^\bullet = \Xi^\bullet = (\Delta_1 \sqcap (\Delta_2 \sqcap \Delta_3))^\bullet$$

  by Definition 4.4.8(4), and hence (4.23) itself is the required derivation;

- If $((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3) \notin T(\Xi, \sqcap)$ for all $\Xi \in \mathit{Struct}$, we have two cases:

– If $\Delta_1 \sqcap \Delta_2 \in T(\Xi', \sqcap)$ for some $\Xi' \in \mathit{Struct}$, then

$$((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)^\bullet = \Xi'^\bullet \sqcap \Delta_3^\bullet.$$

Observe that $\Delta_1^\bullet = \Delta_2^\bullet = \Xi'^\bullet$. Hence the required derivation is

$$\dfrac{\dfrac{d'}{\Gamma'\{\Xi'^\bullet \sqcap \Delta_3^\bullet\} \succ \varphi}}{\Gamma'\{\Xi'^\bullet \sqcap (\Xi'^\bullet \sqcap \Delta_3^\bullet)\} \succ \varphi}\, \sqcap\mathsf{W}_2$$

where clearly $\Gamma'\{\Xi'^\bullet \sqcap (\Xi'^\bullet \sqcap \Delta_3^\bullet)\} = [\Gamma\{(\Delta_1 \sqcap (\Delta_2 \sqcap \Delta_3))\}]^\bullet$;

– If $\Delta_1 \sqcap \Delta_2 \notin T(\Xi', \sqcap)$ for all $\Xi' \in \mathit{Struct}$, then

$$((\Delta_1 \sqcap \Delta_2) \sqcap \Delta_3)^\bullet = ((\Delta_1^\bullet \sqcap \Delta_2^\bullet) \sqcap \Delta_3^\bullet),$$

and the required derivation is

$$\dfrac{\dfrac{d'}{\Gamma'\{((\Delta_1^\bullet \sqcap \Delta_2^\bullet) \sqcap \Delta_3^\bullet)\} \succ \varphi}}{\Gamma'\{(\Delta_1^\bullet \sqcap (\Delta_2^\bullet \sqcap \Delta_3^\bullet))\} \succ \varphi}\, \sqcap\mathsf{A}$$

where again $\Gamma'\{(\Delta_1^\bullet \sqcap (\Delta_2^\bullet \sqcap \Delta_3^\bullet))\} = [\Gamma\{(\Delta_1 \sqcap (\Delta_2 \sqcap \Delta_3))\}]^\bullet$.

The proofs for $\sqcap\mathsf{E}$ and $\sqcap\mathsf{C}$ are simpler, and are therefore left to the reader.

(C) Logical rules. For right-introduction rules, the argument is straightforward; for example, let $\Gamma \succ \varphi$ be the sequent $\Gamma_1 \circ \Gamma_2 \succ \varphi \cdot \psi$, and consider a cut-free derivation

$$\dfrac{\dfrac{d_1}{\Gamma_1 \succ \varphi} \quad \dfrac{d_2}{\Gamma_2 \succ \psi}}{\Gamma_1 \circ \Gamma_2 \succ \varphi \cdot \psi}\, \cdot\mathsf{R} \tag{4.24}$$

By the IH, there exist cut-free, 3-reduced derivations of $\Gamma_1^\bullet \succ \varphi$ and $\Gamma_2^\bullet \succ \psi$. Applying $\cdot\mathsf{R}$ to the latter sequents, and using Definition 4.4.8(2), we immediately obtain a derivation of $(\Gamma_1 \circ \Gamma_2)^\bullet \succ \varphi \cdot \psi$.

As for the left-introduction rules, we consider only $\wedge\mathsf{L}$, leaving the remaining cases to the reader. Let $\Gamma \succ \varphi$ have the form $\Gamma\{\psi \wedge \chi\} \succ \varphi$, and suppose it is derived as follows:

$$\dfrac{\dfrac{d}{\Gamma\{\psi \sqcap \chi\} \succ \varphi}}{\Gamma\{\psi \wedge \chi\} \succ \varphi}\, \wedge\mathsf{L} \tag{4.25}$$

By the IH, there exists a cut-free, 3-reduced derivation

$$\frac{d'}{[\Gamma\{\psi \sqcap \chi\}]^\bullet \succ \varphi} \tag{4.26}$$

We define $\Gamma_p := \Gamma\{\psi \sqcap \chi\}$ and $\Gamma' := \Gamma_p^\bullet$. By proceeding similarly to (A) and (B), we can identify in Γ' an occurrence of $(\psi \sqcap \chi)^\bullet$ relative to which the inference steps leading to $[\Gamma\{\psi \wedge \chi\}]^\bullet \succ \varphi$ must be applied. Hence (4.26) can be rewritten as

$$\frac{d'}{\Gamma'\{(\psi \sqcap \chi)^\bullet\} \succ \varphi}$$

We now consider the following cases:

- If $\psi = \chi$, then $(\psi \sqcap \chi)^\bullet = \psi^\bullet = \psi$ by Definition 4.4.8(1,4). We can therefore replace the reference occurrence of $(\psi \sqcap \chi)^\bullet$ in Γ' with an occurrence of φ and extend (4.26) as follows:

  $$\dfrac{\dfrac{d'}{\Gamma'\{\psi\} \succ \varphi}}{\Gamma'\{\psi \wedge \psi\} \succ \varphi} \wedge\mathsf{W}_1$$

  If the newly introduced formula $\psi \wedge \psi$ occurs within the scope of a structural meet whose other operand is also a copy of $\varphi \wedge \varphi$, then a single application of $\sqcap\mathsf{C}$ suffices to obtain the desired derivation; otherwise, we already obtained the desired derivation.

- If $\psi \neq \chi$, then $(\psi \sqcap \chi)^\bullet = \psi^\bullet \sqcap \chi^\bullet = \psi \sqcap \chi$ by Definition 4.4.8(1,3). We can therefore replace the reference occurrence of $(\psi \sqcap \chi)^\bullet$ in Γ' with an occurrence of $\varphi \sqcap \psi$ and proceed to extending (4.26) as follows:

  $$\dfrac{\dfrac{d'}{\Gamma'\{\psi \sqcap \chi\} \succ \varphi}}{\Gamma'\{(\psi \wedge \chi)\} \succ \varphi} \wedge\mathsf{L}$$

  As in the previous case, if the newly introduced formula $\psi \wedge \chi$ occurs within the scope of a structural meet whose other operand is a copy of $\psi \wedge \chi$, then an application of $\sqcap\mathsf{C}$ yields the required derivation; otherwise, we are done. □

4.4.3 Concise proofs

Definition 4.4.16. A cut-free LMC-derivation d is said to be *concise* iff all of the following conditions are satisfied:

1. d is 3-reduced;

2. d is in T-normal form;

3. no sequent occurs more than once in d.

Proposition 4.4.17. $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$ *iff there exists a concise proof of* $\Gamma^\bullet \succ \varphi$.

Proof. If $\Gamma \succ \varphi$ is provable, then, by Corollary 4.3.5 and Lemma 4.4.14, there exists a cut-free, 3-reduced derivation d of $\Gamma^\bullet \succ \varphi$. Applying T-normalisation to d yields a derivation d' of $\Gamma^\bullet \succ \varphi$ in T-normal form. We prune the proof tree by removing all paths instantiating sequences of inference steps (if any) that give rise to repeated occurrences of sequents. In doing so, we remove, for example, subderivations of the following forms:

$$\dfrac{\dfrac{\Delta\{((\Xi_1 \divideontimes \Xi_2) \divideontimes \Xi_3)\} \succ \varphi}{\Delta\{(\Xi_1 \divideontimes (\Xi_2 \divideontimes \Xi_3))\} \succ \varphi}\,\divideontimes\mathsf{A}}{\Delta\{((\Xi_1 \divideontimes \Xi_2) \divideontimes \Xi_3)\} \succ \varphi}\,\divideontimes\mathsf{A} \qquad \dfrac{\dfrac{\Delta\{\Xi_1 \sqcap \Xi_2\} \succ \varphi}{\Delta\{\Xi_2 \sqcap \Xi_1\} \succ \varphi}\,\sqcap\mathsf{E}}{\Delta\{\Xi_1 \sqcap \Xi_2\} \succ \varphi}\,\sqcap\mathsf{E}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta\{\Xi\} \succ \varphi}{\Delta\{\Xi \sqcap \Xi\} \succ \varphi}\,\sqcap\mathsf{W}_1}{\Delta\{(\Xi \sqcap \Xi) \sqcap \Xi\} \succ \varphi}\,\sqcap\mathsf{W}_1}{\Delta\{\Xi \sqcap \Xi\} \succ \varphi}\,\sqcap\mathsf{C}}{\Delta\{\Xi\} \succ \varphi}\,\sqcap\mathsf{C}$$

where $\divideontimes \in \{\circ, \sqcap\}$. We thus obtain a concise derivation of $\Gamma^\bullet \succ \varphi$. Conversely, if there exists a concise derivation of $\Gamma^\bullet \succ \varphi$, then $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$ by Corollary 4.4.12. □

Proposition 4.4.18. *For any provable* 1*-reduced sequent* $\Gamma^\bullet \succ \varphi$*, there exist only finitely many concise proofs of it.*

Proof. By Corollary 4.3.5, in any concise derivation d of $\Gamma^\bullet \succ \varphi$ there is an upper bound on the number of structural terms occurring in d. Moreover, by Lemmas 4.4.4 and 4.4.14, together with Definition 4.4.16, the number of occurrences of structural terms in d is bounded from above. It follows that there is likewise an upper bound on the number of sequent occurrences in d. Finally, d is finitely branching, since $|\beta(d)|$ coincides with the number of initial sequents occurring in d, which is finite. □

Theorem 4.4.19. LMC *is decidable.*

Proof. Given a sequent $\Gamma \succ \varphi$, by Proposition 4.4.17, in order to decide whether it is a theorem of LMC it suffices to exhibit a concise derivation of $\Gamma^\bullet \succ \varphi$. By Proposition 4.4.18, if $\vdash_{\mathsf{LMC}} \Gamma \succ \varphi$, then the backward proof search for a concise derivation of $\Gamma^\bullet \succ \varphi$ is terminating. If the search fails, then no concise derivation of $\Gamma^\bullet \succ \varphi$ exists, and hence $\nvdash_{\mathsf{LMC}} \Gamma \succ \varphi$. □

Corollary 4.4.20. $\mathfrak{LMC}$ *has a decidable equational theory.*

Proof. Immediate from Theorems 4.2.3 and 4.4.19. □

4.5 Related work

LMC is not the first substructural logic to feature a residuated pair of modal operators. The fundamental role played by adjunction in modal logic is well established, as evidenced by Dunn's Gaggle Theory [57, 58, 59] and related display calculi (see, e.g., [79]), as well as by the extensive body of work on categorial type logics developed over the past three decades [13, 14, 110, 127, 128, 129]. It is to the latter research programme that we owe significant advances in the proof-theoretic treatment of residuated or Galois-connected unary modalities, beginning with Moortgat's (non-associative) modal Lambek calculus $\mathsf{NL}(\Diamond)$ [127], from which we took both the logical and the structural modal rules for LMC.

However, as noted in [119], despite such a large number of contributions, the development of Gentzen-style systems for *positive*, *implication-free* structures equipped with adjoint pairs of unary operators is a relatively recent research direction. Inspired by Kashima's nested sequents for tense logic [97], Sadrzadeh and Dyckhoff [152] initiated this line of inquiry by introducing a tree-style sequent calculus for bounded distributive lattices endowed with *families* of residuated pairs of the form $\langle \diamondsuit, \Box \rangle$ (backward dimond/forward box). Here, although we focus on structures equipped with a single residuated pair, we work within a richer algebraic framework that includes a (non-commutative) monoid multiplication and imposes stronger conditions on the modal operators. In developing both our algebraic model for f-properties and LMC, we have deliberately chosen to begin with a small set of fundamental operations. From the viewpoint of rule apparatus, however, the calculus is not strictly minimal due to the inclusion of K, T, and $\mathsf{4}$.

Part II

Behavioural analysis of dynamic epistemic multi-agent systems

Chapter 5

Epistemic Hennessy–Milner Logic

The formal descriptions of the evolution of concurrent systems, used for specifying (and verifying) behavioural properties, rely on different families of modal logics.

In *state-based approaches*, the aim is to specify *what* happens in systems' states along particular executions. In this setting, system configurations correspond to "snapshots" at specific instants in time. Depending on whether a linear-time or branching-time framework is adopted, behavioural properties can be expressed in a wide range of formalisms, ranging from the classical *Linear Temporal Logic* (LTL) [139], *Computation Tree Logic* (CTL) [40], and CTL* [61] (which famously subsumes LTL and CTL), to *hyperlogics* [44].

In *action-based approaches*, the focus shifts to capturing *how* a system can evolve once a particular state is reached. The inherently temporal aspect of configuration change is therefore secondary to the formal specification of which actions a system can or cannot perform while executing a program. In this context, behavioural properties are most effectively captured using dynamic logics [88], a family of non-classical formalisms with modal operators indexed over (variables for) individual elementary actions or structured sets of actions (programs). The intended models are clearly based on LTSs. The development of such logics was initiated in [143] to provide a semantical counterpart to Hoare Logic [92]. The simplest dynamic logic is *Hennessy–Milner Logic* (HML) [91], which extends classical propositional logic with connectives of the form $\langle\pi\rangle$ and $[\pi]$ (for π an action tag) specifying conditions satisfied by a system in states reached by executing the action π. It is worth noting that both HML and CTL* naturally embed into the modal μ-calculus [104].

In both of these frameworks, computational infrastructures are modelled by considering each system as a whole. In *multi-agent approaches* to behavioural analysis, properties are described in terms of system components viewed as agents performing different tasks, interacting, and exchanging information. On the one hand, temporal formalisms for multi-agent systems are abundant in the literature, including *Alternating-time Temporal Logic* (ATL) [11], *Strategic Logic* (SL) [36], *Alternating-time Temporal Epistemic Logic* (ATEL) [163], as well as many extensions of standard temporal logics with agent-based modalities (see, e.g., [151]). On the other hand, extensions of action-based logics have been successfully applied to modelling and verification of multi-agent systems, building on the rich framework of *Dynamic Epistemic Logic* (DEL) [165].

By adopting a multi-agent perspective, this chapter introduces EHML (Epistemic Hennessy–Milner Logic), a logic for the analysis of dynamic epistemic multi-agent systems. These systems are represented using a novel model of computation, which we call a *Kripke labelled transition system* (*KLTS*). Informally, a KLTS is a two-layer structure consisting of an underlying LTS, where each state is associated with a multi-agent Kripke frame, and whose actions represent frame-transforming operations. When the associated frames are epistemic, the transitions of a KLTS naturally model the evolution of each agent's knowledge along system executions. We therefore begin by defining KLTSs.

5.1 Kripke labelled transition systems

As anticipated above, the notion of a KLTS combines that of an LTS with that of a multi-modal Kripke frame. For LTSs, we adopt Definition 2.1.1, with the additional assumption that the action set Act is finite. We denote by $\mathfrak{LTS}$ the class of LTSs so defined. We now proceed to define frames.

Definition 5.1.1. A (*multi-modal*) *Kripke frame* is a pair $\mathcal{F} = \langle W, \{R_i\}_{i\in I}\rangle$ where W is a non-empty set of *possible worlds* and $\{R_i\}_{i\in I}$ is a finite family of binary *accessibility relations* over W. We denote by $\mathfrak{K}$ the class of all Kripke frames. By a *pointed* Kripke frame we mean a pair $\langle \mathcal{F}, w_0\rangle$, where $\mathcal{F} = \langle W, \{R_i\}_{i\in I}\rangle$ is a Kripke frame and w_0 is a distinguished element of W called the *current world*.

LTSs and Kripke frames are formally equivalent structures. Let $A_1, \ldots, A_n$ be non-empty sets and let $R \subseteq \prod_{k=1}^{n} A_k$. For $i \leq n$, we can define an A_i-indexed family of

$(n-1)$-ary relations $\{R_a\}_{a\in A_i}$ over

$$\prod_{k=1}^{i-1} A_k \times \prod_{k=i+1}^{n} A_k$$

where, for all $a_1 \in A_1, \ldots, a_{i-1} \in A_{i-1}, a_{i+1} \in A_{i+1}, \ldots, a_n \in A_n$,

$$\langle a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n\rangle \in R_a \quad \text{iff} \quad \langle a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_n\rangle \in R.$$

It follows that each LTS $\mathcal{T} = \langle S, Act, T\rangle$ may be viewed as a Kripke frame

$$\phi(\mathcal{T}) = \langle S, \{T_{\mathtt{a}}\}_{\mathtt{a}\in Act}\rangle$$

where each relation $T_{\mathtt{a}}$ is a binary accessibility relation defined from T using the above construction. *Vice versa*, suppose we are given non-empty sets $A_1, \ldots, A_{n-1}$ and a family of $(n-1)$-ary relations $\{R_a\}_{a\in A}$ over $\prod_{k=1}^{n-1} A_k$. Then, for $i \leq n-1$, we can define a new n-ary relation

$$R \subseteq \prod_{k=1}^{i} A_k \times A \times \prod_{k=i+1}^{n-1} A_k$$

such that, for all $a_1 \in A_1, \ldots, a_{n-1} \in A_{n-1}$,

$$\langle a_1, \ldots, a_i, a, a_{i+1}, \ldots, a_{n-1}\rangle \in R \quad \text{iff} \quad \langle a_1, \ldots, a_i, a_{i+1}, \ldots, a_n\rangle \in R_a.$$

It follows that each Kripke frame $\mathcal{F} = \langle W, \{R\}_{i\in I}\rangle$ corresponds to an LTS

$$\lambda(\mathcal{F}) = \langle W, I, R\rangle$$

where R is a transition relation defined from $\{R_i\}_{i\in I}$ and I using the above construction. In light of these considerations, the following holds.

Lemma 5.1.2. *There is a one-to-one correspondence between* $\mathfrak{LTS}$ *and* $\mathfrak{K}$ *implemented by the maps* $\mathcal{T} \mapsto \phi(\mathcal{T})$ *and* $\mathcal{F} \mapsto \lambda(\mathcal{F})$.

Remark 5.1.3. Kripke frames can be equivalently defined as triples $\mathcal{F} = \langle W, I, r\rangle$ where W is a non-empty set of possible worlds, I is a non-empty set of indices, and $r\colon I \to \wp(W \times W)$ is a map assigning to each $i \in I$ a relation $r(i) \subseteq W^2$.

Kripke frames form the basis of the semantics of various modal logics. In the case of epistemic logics, they model knowledge in terms of information accessibility. In such a setting, the set I represents a set of rational agents, and the relations R_i specify which worlds are compatible with each agent's knowledge. As here we focus exclusively on this perspective, we will henceforth refer to multi-modal Kripke frames as *multi-agent* Kripke frames, and write Ag instead of I to emphasise that the indices identifying the relations—and the corresponding modal operators—refer to agents.

Definition 5.1.4. A *Kripke labelled transition system* (*KLTS*) is a quadruple

$$\mathcal{S} = \langle \mathcal{T}, At, Ag, r \rangle$$

where:

- $\mathcal{T} = \langle S, Act, T, s_0 \rangle$ is an LTS;
- At is a non-emtpy set of atomic propositions;
- Ag is a non-empty set of agents;
- $r \colon Ag \times S \to \wp(\wp(At) \times \wp(At))$ is a function assigning to each agent $i \in Ag$ at each state $s \in S$ an accessibility relation $r(i, s) \subseteq \wp(At)^2$.

We denote by $\mathfrak{KLTS}$ the class of al KLTSs.

In a KLTS $\mathcal{S}$, the function r associates each state s of the underlying LTS $\mathcal{T}$ with a frame $\mathcal{F}_s = \langle \wp(At), \{r(i, s)\}_{i \in Ag} \rangle$. It is useful to consider the alternative characterization provided in Remark 5.1.3: using partial function application, we can parameterise r over S and decompose it into an S-indexed family

$$\Big\{ r_s : Ag \to \wp(\wp(At) \times \wp(At)) \Big\}_{s \in S}$$

This yields frames of the form $\mathcal{F}_s = \langle \wp(At), Ag, r_s \rangle$.

When analysing the behaviour of a system modelled by a KLTS, the information generated as the system evolves through different configurations unfolds at two distinct levels:

- an *external* level, corresponding to transitions in the underlying LTS;

- an *internal* level, which describes how possible worlds in frames are related through the accessibility relations.

Note that each accessibility relation $r_s(i)$ relates elements of $\wp(At)$ and represents the actual observational power of agent i in state s with respect to the propositions in At. In other words, $r_s(i)$ describes the agent's capability of distinguishing the possible worlds identified by the values of the propositions.

Under the indistinguishability interpretation of epistemic logic, $r_s(i)$ encodes informational indistinguishability between possible worlds. More precisely, $\langle X, Y\rangle \in r_s(i)$ means that, in state s, agent i cannot determine whether the system is in a state where exactly the propositions in X hold or in a state where exactly the propositions in Y hold. Consequently, both X and Y are consistent with the agent's knowledge in s. By virtue of this interpretation, we henceforth assume that the accessibility relations are equivalence relations.

Example 5.1.5. If $\langle \{p\} \cup X, \{p\} \cup Y\rangle \in r_s(i)$ for any choice of $X, Y \in \wp(At)$, then in state s all worlds in which p holds are indistinguishable from the viewpoint of agent i. Moreover, if $\langle \{p\} \cup X, Y\rangle \notin r_s(i)$ whenever $p \notin Y$, we conclude that agent i distinguishes precisely those pairs of worlds that differ in the valuation of p. As we will later realise, this means that, in s, agent i *knows* the truth value of p and is ignorant of any other proposition.

5.2 Syntax and semantics

We now introduce the syntax and semantics of EHML. Our logic is designed to reason about KLTSs whose internal frames are epistemic and therefore arises as a combination of HML and the normal multimodal logic $\mathsf{S5}_n$. Unlike previous work in this direction—most notably [100]—EHML prevents knowledge modalities from ranging over dynamic modalities, as, in the intended semantics, the external behaviour of a KLTS determines the evolution of the agents' knowledge, and it is therefore unreasonable to allow agents to know how the system will evolve.

We base the syntax of EHML on an initial layer of $\mathsf{S5}_n$-formulas, which we refer to as *internal*.

Definition 5.2.1. For At a fixed set of atomic propositions and Ag a finite set of agent tags, the set Int of *internal formulas* is defined by the BNF

$$\psi ::= p \mid \top \mid \neg\beta \mid \beta_1 \wedge \beta_2 \mid \mathbf{K}_i\beta$$

where $p \in At$, $i \in Ag$, and $\beta, \beta_1, \beta_2 \in Int$.

For $\psi \in Int$, $\mathrm{sub}(\psi)$ is the set of all subformulas of ψ, and $\mathrm{sub}_0(\psi) := \mathrm{sub}(\psi) \cap At$. We embed internal formulas into the syntax of EHML by enclosing them within the delimiters $\lfloor$ and $\rfloor$ so as to mark their occurrences. We thus define the set $\lfloor Int \rfloor := \{\lfloor\psi\rfloor \mid \psi \in Int\}$ of *delimited internal formulas*. Accordingly, for any $\Delta \subseteq Int$, we set $\lfloor\Delta\rfloor := \{\lfloor\psi\rfloor \mid \psi \in \Delta\}$. We can now turn to the definition of EHML-formulas.

Definition 5.2.2. Let $At_+ := At \cup \lfloor Int \rfloor$. For Act a finite set of action tags, the set Fm of EHML-*formulas* is defined by the following grammar

$$\varphi ::= \xi \mid \top \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \langle\pi\rangle\alpha$$

where $\xi \in At_+$, $\alpha, \alpha_1, \alpha_2 \in Fm$, and $\pi \in Act$.

For both grammars introduced in Definitions 5.2.1 and 5.2.2, disjunction ($\vee$), material implication ($\rightarrow$), and material equivalence ($\leftrightarrow$) are defined as usual. In the syntax of internal formulas, $\mathbf{K}_i\psi$ formulas are read as "agent i knows that ψ", and are referred to as *epistemic formulas*. In the syntax of EHML, $\langle\pi\rangle\varphi$ formulas are read as "after the execution of the action π, φ holds", and are referred to as *dynamic formulas*. For an action tag π, the dual modality $[\pi]$ of $\langle\pi\rangle$ is defined by $[\pi]\varphi := \neg\langle\pi\rangle\neg\varphi$. Analogously to the case of internal formulas, for $\varphi \in Fm$, $\mathrm{sub}(\varphi)$ is the set of all subformulas of φ, and $\mathrm{sub}_0(\varphi) := \mathrm{sub}(\varphi) \cap At$. Moreover, we set $\mathrm{sub}_+(\varphi) := \mathrm{sub}(\varphi) \cap \lfloor Int \rfloor$. By a *pure* HML-*formula* we mean an EHML-formula φ such that $\mathrm{sub}_+(\varphi) = \emptyset$.

We introduce some machinery for formula manipulation. For $\varphi \in Fm$, we write $\varphi\{\xi_1, \ldots, \xi_n\}$ to indicate that $\xi_1, \ldots, \xi_n \in At_+$ occur in φ, and we denote by $\varphi\{\xi_1/\varphi_1, \ldots, \xi_n/\varphi_n\}$ the formula obtained from φ by uniformly substituting ξ_1 with φ_1, …, ξ_n with φ_n. We adopt the same convention for internal formulas: $\psi\{p_1, \ldots, p_n\}$ and $\psi\{p_1/\psi_1, \ldots, p_n/\psi_n\}$ denote, respectively, occurrence and uniform substitution in the usual sense. Moreover, for $\varphi \in Fm$ (resp. $\psi \in Int$) and $\varphi' \in \mathrm{sub}(\varphi)$ (resp. $\psi' \in \mathrm{sub}(\psi)$), we write $\varphi\{\varphi'\}$ (resp. $\psi\{\psi'\}$) to denote the choice of an arbitrary but fixed occurrence of φ' (resp. ψ') within φ (resp. ψ).

We now provide a Kripke-style semantics for EHML.

Definition 5.2.3. Let $\mathcal{S} = \langle \mathcal{T}, At, Ag, r \rangle$ be a KLTS with underlying LTS $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. A *model* over $\mathcal{S}$ is a pair $\mathcal{M} = \langle \mathcal{S}, v \rangle$, where $v \colon S \to \wp(At)$ is a valuation function.

Just as each state of a KLTS is associated with a Kripke frame, to evaluate epistemic formulas in a model $\mathcal{M} = \langle \mathcal{S}, v \rangle$, an appropriately defined Kripke model $\mathcal{M}_s$ must be associated with each state s of $\mathcal{S}$.

Definition 5.2.4. Let $\mathcal{M} = \langle \mathcal{S}, v \rangle$ be a model over a KLTS $\mathcal{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. For a state $s \in S$, the *s-model* relative to $\mathcal{M}$ is the Kripke model $\mathcal{M}_s = \langle \mathcal{F}_s, \mathrm{id} \rangle$, where $\mathcal{F}_s = \langle \wp(At), Ag, r_s \rangle$ is the Kripke frame associated with s and id is the identity function on $\wp(At)$.

We now proceed to define the intended models of EHML.

Definition 5.2.5. Let $\mathcal{M}$ be a model over a KLTS $\mathcal{S} = \langle \mathcal{T}, At, Ag, r \rangle$ with underlying LTS $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. We say that $\mathcal{M}$ is an EHML-*model* if the direct image $r[Ag \times S]$ of r is a family of equivalence relations on $\wp(At)$. We define:

$$\mathfrak{EHML} := \{\mathcal{S} \in \mathfrak{KLTS} \mid \langle \mathcal{S}, v \rangle \text{ is an } \mathsf{EHML}\text{-model, for } v \text{ a valuation over } \mathcal{S}\}.$$

This definition implies that, in an EHML-model, every s-model is a model for $\mathsf{S5}_n$.

Definition 5.2.6. Let $\mathcal{M} = \langle \mathcal{S}, v \rangle$ be an EHML-model over $\mathcal{S} = \langle \mathcal{T}, At, Ag, r \rangle$, where $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. For a state $s \in S$, the *satisfaction* of an internal formula generated over At with respect to the s-model $\mathcal{M}_s = \langle \mathcal{F}_s, \mathrm{id} \rangle$ and a world $X \in \wp(At)$ is recursively defined as follows:

(a) $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} p$ *iff* $p \in \mathrm{id}(X)$
(b) $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \top$
(c) $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \neg\psi$ *iff* $\mathcal{M}_s, X \nVdash_{\mathsf{S5}_n} \psi$
(d) $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \psi_1 \wedge \psi_2$ *iff* $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \psi_1$ *and* $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \psi_2$
(e) $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \mathbf{K}_i\psi$ *iff* $\forall Y. \langle X, Y \rangle \in r_s(i)$ *implies* $\mathcal{M}_s, Y \Vdash_{\mathsf{S5}_n} \psi$

Definition 5.2.7. Let $\mathcal{M} = \langle \mathcal{S}, v \rangle$ be an EHML-model over $\mathcal{S} = \langle \mathfrak{T}, At, Ag, r \rangle$, where $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. We define the *satisfaction* of an EHML-formula generated over At_+

with respect to $\mathfrak{M}$ and a state $s \in S$ as follows:

$$
\begin{array}{ll}
(1) & \mathcal{M}, s \Vdash p \textit{ iff } p \in v(s) \\
(1_+) & \mathcal{M}, s \Vdash \lfloor \psi \rfloor \textit{ iff } \mathcal{M}_s, v(s) \Vdash_{\mathsf{S5}_n} \psi \\
(2) & \mathcal{M}, s \Vdash \top \\
(3) & \mathcal{M}, s \Vdash \neg\varphi \textit{ iff } \mathcal{M}, s \nVdash \varphi \\
(4) & \mathcal{M}, s \Vdash \varphi_1 \wedge \varphi_2 \textit{ iff } \mathcal{M}, s \Vdash \varphi_1 \textit{ and } \mathcal{M}, s \Vdash \varphi_2 \\
(5) & \mathcal{M}, s \Vdash \langle\pi\rangle\varphi \textit{ iff } \exists s'.\, \langle s, \pi, s'\rangle \in T \textit{ and } \mathcal{M}, s' \Vdash \varphi
\end{array}
$$

Definition 5.2.8. Let φ be an EHML-formula.

1. φ is *true* in an EHML-model $\mathcal{M} = \langle \mathcal{S}, v\rangle$ (written $\mathcal{M} \Vdash \varphi$) if $\mathcal{M}, s \Vdash \varphi$, for each state s of $\mathcal{S}$;

2. φ is *valid in* $\mathcal{S} \in \mathfrak{EHML}$ (written $\mathcal{S} \Vdash \varphi$) if it is true in every EHML-model over $\mathcal{S}$;

3. φ is *valid in* $\mathfrak{EHML}$ (written $\mathfrak{EHML} \Vdash \varphi$) if $\mathcal{S} \Vdash \varphi$, for all $\mathcal{S} \in \mathfrak{EHML}$.

Truth and validity are extended to sets of formulas as usual: for $\Gamma \subseteq Fm$, we write $\mathcal{M} \Vdash \Gamma$ (resp. $\mathcal{S} \Vdash \Gamma$, $\mathfrak{EHML} \Vdash \Gamma$) if $\mathcal{M} \Vdash \varphi$ (resp. $\mathcal{S} \Vdash \varphi$, $\mathfrak{EHML} \Vdash \varphi$), for all $\varphi \in \Gamma$.

Definition 5.2.9. Let $\Gamma \cup \{\varphi\} \subseteq Fm$. The formula φ is a (*local*) *semantic consequence* of Γ over $\mathfrak{EHML}$ (henceforth, $\mathfrak{EHML}$-*consequence*) if $\mathcal{M}, s \Vdash \Gamma$ implies $\mathcal{M}, s \Vdash \varphi$, for every EHML-model $\mathcal{M}$ and every state s of $\mathcal{M}$. Two EHML-formulas φ_1 and φ_2 are $\mathfrak{EHML}$-*equivalent* if and only if each is an $\mathfrak{EHML}$-consequence of the other.

Throughout this section, we shall also make use of the standard notions of validity and semantic consequence for $\mathsf{S5}_n$- and HML-formulas. Let $\mathfrak{S}5$ denote the class of reflexive, transitive, and symmetric Kripke frames. For $\psi \in Int$, we write $\mathfrak{S}5 \Vdash \psi$ to indicate that ψ is valid in $\mathfrak{S}5$. Given a pure HML-formula φ, we write $\mathfrak{LTS} \Vdash \varphi$ (equivalently, $\mathfrak{K} \Vdash \varphi$) to mean that φ is valid in $\mathfrak{LTS}$ (equivalently, in $\mathfrak{K}$). Local semantic consequence over $\mathfrak{S}5$ and $\mathfrak{LTS}/\mathfrak{K}$ will be expressed using the same notation as in Definition 5.2.9, with the reference class indicated explicitly each time.

We now analyse in greater detail the semantic consequences of allowing delimited formulas in the syntax of EHML. First, we show that satisfaction of atomic formulas is invariant under delimitation, and the application of delimiters "commutes" with negation and conjunction.

Lemma 5.2.10. *Let $p \in At$ and let $\psi, \psi_1, \psi_2 \in Int$. Then, for any* EHML*-model $\mathcal{M} = \langle \mathcal{S}, v \rangle$ and any state s of $\mathcal{M}$:*

$$\mathcal{M}, s \Vdash \lfloor p \rfloor \quad \Leftrightarrow \quad \mathcal{M}, s \Vdash p \qquad (\mathrm{INV}_p)$$

$$\mathcal{M}, s \Vdash \lfloor \top \rfloor \quad \Leftrightarrow \quad \mathcal{M}, s \Vdash \top \qquad (\mathrm{INV}_\top)$$

$$\mathcal{M}, s \Vdash \lfloor \neg\psi \rfloor \quad \Leftrightarrow \quad \mathcal{M}, s \Vdash \neg\lfloor \psi \rfloor \qquad (\mathrm{D}\neg)$$

$$\mathcal{M}, s \Vdash \lfloor \psi_1 \wedge \psi_2 \rfloor \quad \Leftrightarrow \quad \mathcal{M}, s \Vdash \lfloor \psi_1 \rfloor \wedge \lfloor \psi_2 \rfloor \qquad (\mathrm{D}\wedge)$$

Proof. The proof of (INV_p) follows immediately from condition (a) in Definition 5.2.6 together with conditions (1) and (1_+) in Definition 5.2.7. The case of ($\mathrm{INV}_\top$) is analogous, except that (b) and (2) are employed in place of (a) and (1), respectively. Finally, ($\mathrm{D}\neg$) and ($\mathrm{D}\wedge$) are established by a routine induction on the syntactic complexity of formulas, relying on (1_+) and, respectively, on conditions (c)/(3) and (d)/(4). □

Definition 5.2.11 (*DeNF*)**.** An EHML-formula φ is in *delimiter normal form* if, for every $\lfloor \psi \rfloor \in \mathrm{sub}_+(\varphi)$, $\psi = \mathbf{K}_i\beta$, where $i \in Ag$ and $\beta \in Int$. We denote by *DeNF* the set of all EHML-formulas in delimiter normal form.

Note that, if φ is a pure HML-formula, then clearly $\varphi \in DeNF$.

Proposition 5.2.12. *Every $\varphi \in Fm$ can be transformed into an equivalent $\varphi^\dagger \in DeNF$.*

Proof. Immediate from Lemma 5.2.10. □

Definition 5.2.13. Let φ be an EHML-formula. If $\mathrm{sub}_+(\varphi) = \{\xi_1, \ldots, \xi_n\}$, we denote by $\varphi^{\mathtt{p}}$ a pure HML-formula of the form $\varphi\{\xi_1/p_1, \ldots, \xi_n/p_n\}$, where $p_1, \ldots, p_n \in At \setminus \mathrm{sub}_0(\varphi)$. If φ is a pure HML-formula, then we regard $\varphi^{\mathtt{p}}$ as an alternative notation for φ.

Remark 5.2.14. The superscript $^{\mathtt{p}}$ does not denote a function, since, for a given $\varphi \in Fm$ with $\mathrm{sub}_+(\varphi) \neq \emptyset$, there are multiple ways of choosing fresh atomic propositions from $At \setminus \mathrm{sub}_0(\varphi)$ to replace the delimited subformulas of φ. Consequently, for any $\Gamma \subseteq Fm$, we denote by $\Gamma^{\mathtt{p}}$ the set of all formulas $\varphi^{\mathtt{p}}$ with $\varphi \in \Gamma$, under the assumption that the transformation into pure formulas is carried out in a compatible way across Γ. Formally, for all distinct $\varphi, \varphi' \in \Gamma$ and all $\xi \in \mathrm{sub}_+(\varphi) \cap \mathrm{sub}_+(\varphi')$, once a choice of a proposition $p \in At \setminus \mathrm{sub}_0(\varphi)$ is fixed, the substitution ξ/p is performed in φ if and only if it is performed in φ'.

Proposition 5.2.15. *Let $\langle \mathcal{S}, v\rangle$ be an* EHML*-model over a KLTS $\mathcal{S} = \langle \mathcal{T}, At, Ag, r\rangle$ with $\mathcal{T} = \langle S, Act, T, s_0\rangle$. Then, for $\varphi \in Fm$ and $s \in S$, there exists $v' \colon S \to \wp(At)$ such that $\langle \mathcal{S}, v\rangle, s \Vdash \varphi$ iff $\langle \mathcal{S}, v'\rangle, s \Vdash \varphi^{\mathtt{p}}$.*

Proof. Left to the reader. □

The transformation of EHML-formulas into pure HML-formulas introduced in Definition 5.2.13 allows for a clearer understanding of the notion of validity in $\mathfrak{EHML}$. Trivially, any formula $\varphi \in Fm$ whose pure HML counterparts are valid in $\mathfrak{LTS}$ is valid in $\mathfrak{EHML}$. However, the converse does not hold in general: consider, for instance, the formula $\varphi \wedge \lfloor \psi \rfloor$, where $\varphi := [\pi](p \wedge \lfloor \mathbf{K}_j q \rfloor) \to [\pi]p$ and $\psi := \mathbf{K}_i p \to \neg \mathbf{K}_i \neg p$. Clearly, $\mathfrak{EHML} \Vdash \varphi \wedge \lfloor \psi \rfloor$, as $\mathfrak{LTS} \Vdash \varphi^{\mathtt{p}}$ and $\mathfrak{S5} \Vdash \psi$. However, $(\varphi \wedge \lfloor \psi \rfloor)^{\mathtt{p}}$ has the form $\varphi^{\mathtt{p}} \wedge r$[12], which is plainly not valid in $\mathfrak{LTS}$.

Lemma 5.2.16. *Let $\varphi \in Fm$. The following conditions hold:*

(a) if $\mathrm{sub}_+(\varphi) = \emptyset$, then $\mathfrak{EHML} \Vdash \varphi$ iff $\mathfrak{LTS} \Vdash \varphi$;

(b) if $\mathrm{sub}_+(\varphi) \neq \emptyset$, then $\mathfrak{LTS} \Vdash \varphi^{\mathtt{p}}$ implies $\mathfrak{EHML} \Vdash \varphi$;

(c) if $\mathfrak{EHML} \Vdash \varphi$ and $\mathfrak{LTS} \nVdash \varphi^{\mathtt{p}}$, then there exists $\lfloor \Delta \rfloor \subseteq \lfloor Int \rfloor$ such that $\mathfrak{S5} \Vdash \Delta$, and φ is an $\mathfrak{EHML}$-consequence of $\lfloor \Delta \rfloor$.

Proof. The proof of (a) and (b) is immediate. As for (c), suppose that $\mathfrak{EHML} \Vdash \varphi$ while $\mathfrak{LTS} \nVdash \varphi^{\mathtt{p}}$. Then the validity of φ in $\mathfrak{EHML}$ cannot (at least entirely) be accounted for by the LTS component of $\mathfrak{EHML}$-models. It follows that $\mathrm{sub}_+(\varphi) \neq \emptyset$: otherwise, by (a) and Definition 5.2.13, $\varphi^{\mathtt{p}}$, or equivalently φ, would be valid over $\mathfrak{LTS}$, contradicting the initial assumption. Three cases can then be distinguished. If $\varphi = \lfloor \psi \rfloor$ with $\mathfrak{S5} \Vdash \psi$, then we take $\lfloor \Delta \rfloor = \emptyset$, and the claim follows. If, by applying the delimiter-manipulation rules of Lemma 5.2.10, φ is $\mathfrak{EHML}$-equivalent to $\lfloor \psi \rfloor$ with $\mathfrak{S5} \Vdash \psi$, then we take $\lfloor \Delta \rfloor = \{\lfloor \psi \rfloor\}$, and the claim follows. Finally, if φ contains dynamic subformulas and $\mathfrak{LTS} \nVdash \varphi^{\mathtt{p}}$, then, by applying some (possibly none) of the delimiter-manipulation rules of Lemma 5.2.10, φ can be transformed into an $\mathfrak{EHML}$-equivalent formula φ' such that either $\mathfrak{LTS} \Vdash (\varphi')^{\mathtt{p}}$, or the $\mathfrak{EHML}$-validity of φ' is determined by the $\mathfrak{S5}$-validity of certain delimited internal formulas belonging to a set $\lfloor \Delta' \rfloor \subseteq \mathrm{sub}_+(\varphi')$. Accordingly, it suffices to take either $\lfloor \Delta \rfloor = \emptyset$ or $\lfloor \Delta \rfloor = \lfloor \Delta' \rfloor$. □

[12] Clearly, the choices of fresh atomic propositions in $(\varphi \wedge \lfloor \psi \rfloor)^{\mathtt{p}}$ and $\varphi^{\mathtt{p}}$ are assumed to be compatible in the sense of Remark 5.2.14.

5.3 Axiomatisation, completeness, and decidability

We now proceed to introduce a Hilbert-style proof system for EHML, whose axioms and inference rules are listed in Table 5.1.

HML axioms

(CL) All classical tautologies
(K_π) $[\pi](\varphi_1 \to \varphi_2) \to ([\pi]\varphi_1 \to [\pi]\varphi_2)$
(DUAL_π) $\langle\pi\rangle\varphi \leftrightarrow \neg[\pi]\neg\varphi$

$\mathsf{S5}_n^d$ axioms

(CL^d) All delimited classical tautologies
(K_i^d) $\lfloor \mathbf{K}_i(\psi_1 \to \psi_2) \to (\mathbf{K}_i\psi_1 \to \mathbf{K}_i\psi_2)\rfloor$
(T_i^d) $\lfloor \mathbf{K}_i\psi \to \psi \rfloor$
(5_i^d) $\lfloor \neg\mathbf{K}_i\neg\psi \to \mathbf{K}_i\neg\mathbf{K}_i\neg\psi \rfloor$

Inference rules

$$\frac{\varphi_1 \qquad \varphi_1 \to \varphi_2}{\varphi_2}\ (\mathrm{MP}) \qquad \frac{\varphi}{[\pi]\varphi}\ (\mathrm{NEC}_\pi) \qquad \frac{\lfloor\psi\rfloor}{\lfloor\mathbf{K}_i\psi\rfloor}\ (\mathrm{NEC}_i^d)$$

$$\frac{\varphi\{\lfloor\neg\psi\rfloor\}}{\varphi\{\neg\lfloor\psi\rfloor\}}\ (\mathrm{D}\neg) \qquad \frac{\varphi\{\lfloor\psi_1 \wedge \psi_2\rfloor\}}{\varphi\{\lfloor\psi_1\rfloor \wedge \lfloor\psi_2\rfloor\}}\ (\mathrm{D}\wedge) \qquad \frac{\varphi\{\xi\}}{\varphi\{\lfloor\xi\rfloor\}}\ (\mathrm{INV})$$

$$\frac{\varphi\{\xi_1,\ldots,\xi_n\}}{\varphi\{\varphi_1/\xi_1,\ldots,\varphi_n/\xi_n\}}\ (\mathrm{SUB}) \qquad \frac{\lfloor\psi\{p_1,\ldots,p_n\}\rfloor}{\lfloor\psi\{\psi_1/p_1,\ldots,\psi_n/p_n\}\rfloor}\ (\mathrm{SUB}^d)$$

where the double inference line in (D¬), (D∧), and (INV) indicates that the rules can be applied in both directions. Moreover, in (INV), we require that $\xi \in At \cup \{\top\}$.

Table 5.1: Hilbert-style proof system for EHML.

To analyse KLTS dynamics, the system includes an HML component, which is defined by the axioms (CL), (K_π), and (DUAL_π), together with the rules (MP), (NEC_π), and (SUB). Note that, in light of Lemma 5.1.2, this is just a variant (over Fm) of the normal multimodal system K_n, where n is the cardinality of the set of action tags used in the construction of EHML-formulas. The rules (D¬), (D∧), and (INV) control the interaction between the internal and the external syntactic layers, allowing the delimiters surrounding internal formulas to be manipulated consistently with the equivalence results of Lemma 5.2.10. Finally, to reason about the internal level of EHML-models, we

require a variant of $\mathsf{S5}_n$, denoted $\mathsf{S5}_n^d$, defined over delimited internal formulas and axiomatised by (CL^d), (K_i^d), (T_i^d), and (5_i^d). As the reader will have noticed, this epistemic component of EHML includes (NEC_i^d) and (SUB^d) as the counterparts of the standard necessitation and substitution rules, but lacks the rule

$$\frac{\lfloor\psi_1\rfloor \qquad \lfloor\psi_1 \to \psi_2\rfloor}{\lfloor\psi_2\rfloor}\,(\text{MP}^d)$$

providing the internal-language counterpart of (MP). We now show that (MP^d) is, in fact, implicit in the system and can be derived.

Lemma 5.3.1. (MP^d) *is derivable in* EHML.

Proof. Let $\circledast$ be an arbitrary composition of connectives from $\{\wedge, \neg, \top\}$. By using ($\text{D}\neg$), ($\text{D}\wedge$), and (INV), one immediately derives the following rule:

$$\frac{\varphi\{\lfloor\circledast(\psi_1, \ldots, \psi_n)\rfloor\}}{\varphi\{\circledast(\lfloor\psi_1\rfloor, \ldots, \lfloor\psi_n\rfloor)\}}\,(\text{D}\circledast)$$

Consequently, for $\circledast = \to$, (MP^d) is derived via the proof schema

1.	$\lfloor\psi_1\rfloor$	d_1
2.	$\lfloor\psi_1 \to \psi_2\rfloor$	d_2
3.	$\lfloor\psi_1\rfloor \to \lfloor\psi_2\rfloor$	($\text{D}\to$), 2
4.	$\lfloor\psi_2\rfloor$	(MP), 1, 3

where d_1 and d_2 are two EHML-derivations of $\lfloor\psi_1\rfloor$ and $\lfloor\psi_1 \to \psi_2\rfloor$, respectively. □

We now provide a characterization of the theorems of EHML. Clearly, considered in isolation, both HML and $\mathsf{S5}_n^d$ generate theorems of EHML.

Proposition 5.3.2. *Let* $\Gamma \cup \{\varphi\} \subseteq Fm$ *and* $\Delta \cup \{\psi\} \subseteq Int$. *Then:*

1. $\Gamma \vdash_{\mathsf{HML}} \varphi$ *iff* $\Gamma^{\mathtt{p}} \vdash_{\mathsf{K}_n} \varphi^{\mathtt{p}}$, *for* $(\Gamma \cup \{\varphi\})^{\mathtt{p}}$ *constructed as in Remark 5.2.14;*

2. $\lfloor\Delta\rfloor \vdash_{\mathsf{S5}_n^d} \lfloor\psi\rfloor$ *iff* $\Delta \vdash_{\mathsf{S5}_n} \psi$.

Proof. Left to the reader. □

It remains, however, to account for the hybrid rules ($\text{D}\neg$), ($\text{D}\wedge$), and (INV).

Proposition 5.3.3. *Let $\psi \in Int$. If $\vdash_{\mathsf{S5}_n} \psi$ and φ is obtained from $\lfloor\psi\rfloor$ by applying any of the rules* (D$\neg$), (D$\wedge$), *or* (INV), *then $\vdash_{\mathsf{EHML}} \varphi$. In particular, $\vdash_{\mathsf{EHML}} \lfloor\psi\rfloor^{\dagger}$.*

Let us now define E as the extension of $\mathsf{S5}_n^d$ obtained by adding the rules (D$\neg$), (D$\wedge$), and (INV).

Lemma 5.3.4. *Let φ be an* EHML*-formula. Then $\vdash_{\mathsf{EHML}} \varphi$ if and only if:*

(a) $\vdash_{\mathsf{E}} \varphi$, or

(b) there exists a set Φ of E-theorems such that $\Phi \vdash_{\mathsf{HML}} \varphi$.

Proof. The non-trivial direction of the lemma is the right-to-left one. Suppose that $\vdash_{\mathsf{E}} \varphi$. It then follows immediately that $\vdash_{\mathsf{EHML}} \varphi$, by Proposition 5.3.3 and by the fact that every theorem of $\mathsf{S5}_n^d$ is also a theorem of EHML. Now, suppose that $\Phi \vdash_{\mathsf{HML}} \varphi$ for some $\Phi \subseteq \mathrm{Thm}(\mathsf{E})$. Since this condition concerns derivability in HML, we distinguish two cases. If $\Phi = \emptyset$, then $\vdash_{\mathsf{HML}} \varphi$, and hence $\vdash_{\mathsf{EHML}} \varphi$. If $\Phi \neq \emptyset$, since any theorem of E is a theorem of EHML, then φ is derivable from a set of EHML-theorems, and therefore it is itself an EHML-theorem. □

Observe that, in the previous lemma, (a) implies (b): if $\vdash_{\mathsf{E}} \varphi$, then φ is an HML-consequence of $\{\varphi\}$.

Proposition 5.3.5. *If $\vdash_{\mathsf{EHML}} \varphi_1 \leftrightarrow \varphi_2$ and $\vdash_{\mathsf{EHML}} \varphi\{\varphi_1/\xi\}$ then $\vdash_{\mathsf{EHML}} \varphi\{\varphi_2/\xi\}$.*

Theorem 5.3.6. *Let φ be an* EHML*-formula. Then $\vdash_{\mathsf{EHML}} \varphi$ iff $\mathfrak{EHML} \Vdash \varphi$.*

Proof. We leave it to the reader the easy exercise of verifying the soundness of the inference rules of EHML. The left-to-right direction follows immediately from the standard soundness results for K_n and $\mathsf{S5}_n$, together with Definitions 5.2.6, 5.2.7, 5.2.8, and Lemmas 5.2.10 and 5.3.4. As for the converse, suppose that $\nvdash_{\mathsf{EHML}} \varphi$. By Lemma 5.3.4, we have:

1. φ does not contain dynamic subformulas and $\nvdash_{\mathsf{E}} \varphi$. Due to the rules (D$\neg$), (D$\wedge$), and (INV), φ is equivalent, modulo interderivability in E, to a delimited internal formula $\lfloor\psi\rfloor$. Hence, $\nvdash_{\mathsf{S5}_n^d} \lfloor\psi\rfloor$, and we obtain:

$$\begin{array}{lll} \nvdash_{\mathsf{S5}_n^d} \lfloor\psi\rfloor & \Leftrightarrow \nvdash_{\mathsf{S5}_n} \psi & \text{by Proposition 5.3.2(2)} \\ & \Leftrightarrow \mathfrak{S5} \nVdash \psi & \text{by weak completeness of } \mathsf{S5}_n \text{ [87]} \\ & \Leftrightarrow \mathfrak{EHML} \nVdash \lfloor\psi\rfloor & \text{by Definitions 5.2.7 and 5.2.8} \end{array}$$

2. $\Phi \nvdash_{\mathsf{HML}} \varphi$, for every set of E-theorems Φ. In particular:

   (A) For $\Phi = \emptyset$, $\nvdash_{\mathsf{HML}} \varphi$. Then

$$\begin{array}{lll} \nvdash_{\mathsf{HML}} \varphi & \Leftrightarrow \nvdash_{\mathsf{K}_n} \varphi^{\mathfrak{p}} & \text{by Proposition 5.3.2(1)} \\ & \Leftrightarrow \mathfrak{K} \nVdash \varphi^{\mathfrak{p}} & \text{by weak completeness of } \mathsf{K}_n \text{ [87]} \\ & \Leftrightarrow \mathfrak{LTS} \nVdash \varphi^{\mathfrak{p}} & \text{by Lemma 5.1.2} \\ & \Leftrightarrow \mathfrak{EHML} \nVdash \varphi & \text{by Proposition 5.2.15} \end{array}$$

   (B) Let $\Phi \neq \emptyset$. We are thus in the situation where, for every non-empty set Φ of E-theorems, no HML-derivation assuming Φ derives φ as a conclusion. Moreover, since the system E is sound with respect to $\mathfrak{EHML}$, we have $\mathfrak{EHML} \Vdash \alpha$ for every $\alpha \in \Phi$. Assuming that $(\Phi \cup \{\varphi\})^{\mathfrak{p}}$ is constructed following Remark 5.2.14, we have:

$$\begin{array}{lll} \Phi \nvdash_{\mathsf{HML}} \varphi & \Leftrightarrow \Phi^{\mathfrak{p}} \nvdash_{\mathsf{K}_n} \varphi^{\mathfrak{p}} & \text{by Proposition 5.3.2(1)} \\ & \Leftrightarrow \Phi^{\mathfrak{p}} \nVdash \varphi^{\mathfrak{p}} \text{ in } \mathfrak{K} & \text{by strong completeness of } \mathsf{K}_n \text{ [165]} \\ & \Leftrightarrow \exists \mathcal{F} \in \mathfrak{K} : \mathcal{F} \Vdash \Phi^{\mathfrak{p}} \text{ and } \mathcal{F} \nVdash \varphi^{\mathfrak{p}} & \text{by def. of } \mathfrak{K}\text{-consequence} \\ & \Leftrightarrow \exists \mathcal{T} \in \mathfrak{LTS} : \mathcal{T} \Vdash \Phi^{\mathfrak{p}} \text{ and } \mathcal{T} \nVdash \varphi^{\mathfrak{p}} & \text{by Lemma 5.1.2} \end{array}$$

   We conclude that $\mathfrak{LTS} \nVdash \varphi^{\mathfrak{p}}$. Since $\varphi^{\mathfrak{p}}$ is a pure HML-formula, by Lemma 5.2.16(a) we have that $\mathfrak{EHML} \Vdash \varphi^{\mathfrak{p}}$ iff $\mathfrak{LTS} \Vdash \varphi^{\mathfrak{p}}$. Hence, $\mathfrak{EHML} \nVdash \varphi^{\mathfrak{p}}$. We need to show that $\mathfrak{EHML} \nVdash \varphi$. Suppose, towards a contradiction, that $\mathfrak{EHML} \Vdash \varphi$. Then the antecedent of Lemma 5.2.16(c) is satisfied. It follows that there exists a set $\lfloor \Delta \rfloor$ of delimited $\mathfrak{S5}$-valid formulas such that φ is an $\mathfrak{EHML}$-consequence of $\lfloor \Delta \rfloor$. We therefore reconsider the three cases distinguished in Lemma 5.2.16(c):

   (B1) $\varphi = \lfloor \psi \rfloor$ with $\mathbb{S}5 \Vdash \psi$. Hence, φ would be a theorem of E, and we would have $\{\varphi\} \vdash_{\mathsf{HML}} \varphi$, contradicting the initial assumption. It follows that $\mathfrak{EHML} \nVdash \varphi$;

   (B2) by applying the delimiter-manipulation rules of Lemma 5.2.10, φ is $\mathfrak{EHML}$-equivalent to $\lfloor \psi \rfloor$ with $\mathfrak{S5} \Vdash \psi$. Again, φ would be a theorem of E, and we would have $\{\varphi\} \vdash_{\mathsf{HML}} \varphi$, contradicting the initial assumption. Hence, $\mathfrak{EHML} \nVdash \varphi$;

   (B3) φ contains dynamic subformulas and, by applying some (possibly none) delimiter-manipulation rules, φ can be transformed into an $\mathfrak{EHML}$-equivalent

formula φ' such that either $\mathfrak{LTS} \Vdash (\varphi')^{\mathtt{p}}$, or the $\mathfrak{EHML}$-validity of φ' is determined by the $\mathfrak{S5}$-validity of certain delimited internal formulas belonging to a set $\lfloor\Delta'\rfloor \subseteq \mathrm{sub}_+(\varphi')$. Under these conditions, the derivation of φ' from $\lfloor\Delta'\rfloor$ in EHML relies solely on the HML machinery; hence,

$$\lfloor\Delta'\rfloor \vdash_{\mathsf{HML}} \varphi'.$$

Moreover, in E we can derive a family of biconditionals expressing the delimiter manipulations applied in transforming φ into φ'. These formulas have the form

$$\lfloor\circledast(\beta_1, \ldots, \beta_n)\rfloor \leftrightarrow \circledast(\lfloor\beta_1\rfloor, \ldots, \lfloor\beta_n\rfloor),$$

where $\circledast$ is a combination of connectives from $\{\wedge, \neg, \top\}$. Let $\Phi_{\leftrightarrow}$ denote the set of all such biconditionals. We therefore have

$$\lfloor\Delta'\rfloor \cup \Phi_{\leftrightarrow} \vdash_{\mathsf{HML}} \varphi,$$

contradicting the initial assumption. It follows that $\mathfrak{EHML} \nVdash \varphi$. □

Example 5.3.7. We provide a simple example to clarify item (B3) in the above proof. Let

$$\alpha_1 := [\pi](p \wedge \lfloor \mathbf{K}_j q\rfloor) \to [\pi]p \qquad \psi := \mathbf{K}_i p \to \neg\mathbf{K}_i\neg p \qquad \alpha_2 := \lfloor \mathbf{K}_i p\rfloor \to \lfloor\neg\mathbf{K}_i\neg p\rfloor$$

Clearly $\mathfrak{S5} \Vdash \psi$, so we have $\mathfrak{EHML} \Vdash \lfloor\psi\rfloor$, and by Lemma 5.2.10, $\lfloor\psi\rfloor$ and α_2 are $\mathfrak{EHML}$-equivalent. It is immediate to give an HML-proof of α_1:

1.	$(p \wedge \lfloor \mathbf{K}_j q\rfloor) \to p$	CL	
2.	$[\pi]((p \wedge \lfloor \mathbf{K}_j q\rfloor) \to p) \to ([\pi](p \wedge \lfloor \mathbf{K}_j q\rfloor) \to [\pi]p)$	(K_π)	(d_{α_1})
3.	$[\pi]((p \wedge \lfloor \mathbf{K}_j q\rfloor) \to p)$	NEC_π, 1	
4.	$[\pi](p \wedge \lfloor \mathbf{K}_j q\rfloor) \to [\pi]p$	MP, 2, 4	

Therefore $\mathfrak{LTS} \Vdash \alpha_1^{\mathtt{p}}$ and, by Lemma 5.2.16(b), $\mathfrak{EHML} \Vdash \alpha_1$. Let now $\varphi := \alpha_1 \wedge \alpha_2$. Clearly, $\mathfrak{EHML} \Vdash \varphi$ whereas $\mathfrak{LTS} \nVdash \varphi^{\mathtt{p}}$. Letting $\varphi' := \alpha_1 \wedge \lfloor\psi\rfloor$, one readily verifies that φ and φ' are $\mathfrak{EHML}$-equivalent. Note that $\mathfrak{EHML} \Vdash \varphi'$ *because* $\mathfrak{S5} \Vdash \psi$, whence $\{\lfloor\psi\rfloor\} \Vdash \varphi'$, and consequently $\{\lfloor\psi\rfloor\} \Vdash \varphi$. The first entailment is provable in HML:

1.	α_1	(d_{α_1})	
2.	$\lfloor\psi\rfloor$	Assumption	
3.	$\alpha_1 \to (\lfloor\psi\rfloor \to (\alpha_1 \wedge \lfloor\psi\rfloor))$	(CL)	$(d_{\varphi'})$
4.	$\lfloor\psi\rfloor \to (\alpha_1 \wedge \lfloor\psi\rfloor)$	(MP), 1, 3	
5.	$\alpha_1 \wedge \lfloor\psi\rfloor$	(MP), 2, 4	

We can easily construct a HML-derivation of φ that uses $\lfloor\psi\rfloor$. To this end, we construct a proof in E of the formula $\varepsilon := \lfloor\psi\rfloor \leftrightarrow \alpha_2$:

1.	$\lfloor(\mathbf{K}_i p \to \neg\mathbf{K}_i\neg p) \leftrightarrow (\mathbf{K}_i p \to \neg\mathbf{K}_i\neg p)\rfloor$	CL^d	
2.	$\lfloor\mathbf{K}_i p \to \neg\mathbf{K}_i\neg p\rfloor \leftrightarrow \lfloor\mathbf{K}_i p \to \neg\mathbf{K}_i\neg p\rfloor$	(D↔), 1	(d_ε)
3.	$\lfloor\mathbf{K}_i p \to \neg\mathbf{K}_i\neg p\rfloor \leftrightarrow (\lfloor\mathbf{K}_i p\rfloor \to \lfloor\neg\mathbf{K}_i\neg p\rfloor)$	(D→), 1	

We can now obtain a derivation of φ from $\lfloor\psi\rfloor$ and ε:

1.	$\alpha_1 \wedge \lfloor\psi\rfloor$	$(d_{\varphi'})$	
2.	$\lfloor\psi\rfloor \leftrightarrow \alpha_2$	(d_ε)	(d_φ)
3.	$\alpha_1 \wedge \lfloor\psi\rfloor$	Proposition 5.3.5, 1, 2	

We now turn to establish decidability results for EHML. By exploiting the two-layer structure of EHML, the theorems we aim for can be inherited from those concerning K_n and $\mathsf{S5}_n$ (see [87] for further details).

Theorem 5.3.8. EHML *is decidable.*

Proof. Let φ be an EHML-formula. If $\varphi = \lfloor\psi\rfloor$, then, by Lemma 5.3.4 and Proposition 5.3.2, $\vdash_{\mathsf{EHML}} \lfloor\psi\rfloor$ if and only if $\vdash_{\mathsf{S5}_n^d} \lfloor\psi\rfloor$, which in turn holds if and only if $\vdash_{\mathsf{S5}_n} \psi$. Hence, in this case, the algorithmic verification of whether φ is a theorem reduces to the decision procedure for $\mathsf{S5}_n$. Suppose now that φ is not a delimited internal formula. We distinguish two cases. If $\mathrm{sub}_+(\varphi) = \emptyset$, then the decision procedure for K_n applies directly. Otherwise, let $\mathrm{sub}_+(\varphi) = \{\lfloor\psi_1\rfloor, \ldots, \lfloor\psi_n\rfloor\}$. We define the following algorithm:

1. We construct a pure HML-formula φ^{p} and apply the decision procedure for K_n. If $\vdash_{\mathsf{K}_n} \varphi^{\mathsf{p}}$, then, by Proposition 5.3.2, $\vdash_{\mathsf{EHML}} \varphi$. Otherwise, we proceed to Step 2.

2. We construct the formula $\varphi_{\min}$ (which is $\mathfrak{EHML}$-equivalent to φ by Lemma 5.2.10) by reducing each occurrence of $\lfloor\psi_1\rfloor, \ldots, \lfloor\psi_n\rfloor$ to the corresponding formulas $\lfloor\psi_1\rfloor^\dagger, \ldots, \lfloor\psi_n\rfloor^\dagger \in \mathit{DeNF}$. We proceed to Step 3.

3. Let $X := \mathrm{sub}_+(\varphi_{\min})$, and let $Y := \bigcup\{Z \subseteq X \mid \vdash_{\mathsf{S5}_n} \beta, \text{ for all } \lfloor\beta\rfloor \in Z\}$. Suppose $Y = \{\lfloor\beta_1\rfloor, \ldots, \lfloor\beta_m\rfloor\}$. We construct the formula

$$\alpha := \varphi_{\min}\{\lfloor\beta_1\rfloor/\top, \ldots, \lfloor\beta_m\rfloor/\top\},$$

which is clearly $\mathfrak{EHML}$-equivalent to $\varphi_{\min}$. We have the following two cases:

(a) $Y = X$. We apply the decision procedure for K_n. If $\vdash_{\mathsf{K}_n} \alpha$, then $\vdash_{\mathsf{EHML}} \alpha$, therefore, by Theorem 5.3.6, $\vdash_{\mathsf{EHML}} \varphi_{\min}$, whence $\vdash_{\mathsf{EHML}} \varphi$; otherwise, $\nvdash_{\mathsf{EHML}} \alpha$ and therefore $\nvdash_{\mathsf{EHML}} \varphi$.

(b) $Y \subsetneq X$. We construct a formula α' (which is $\mathfrak{EHML}$-equivalent to α by Lemma 5.2.10) by, whenever possible, shifting the delimiters around each $\lfloor\beta\rfloor \in X \setminus Y$ so as to include the formula in which $\lfloor\beta\rfloor$ occurs as an immediate subformula. If $\alpha' \neq \alpha$, then, setting $\varphi_{\min} := \alpha'$, we repeat the procedure of Step 3. If $\alpha' = \alpha$, we apply the decision procedure for K_n to $(\alpha')^{\mathtt{p}}$. If $\vdash_{\mathsf{K}_n} (\alpha')^{\mathtt{p}}$, then $\vdash_{\mathsf{EHML}} \alpha'$, therefore, by Theorem 5.3.6, $\vdash_{\mathsf{EHML}} \varphi_{\min}$, whence $\vdash_{\mathsf{EHML}} \varphi$; otherwise, $\nvdash_{\mathsf{EHML}} (\alpha')^{\mathtt{p}}$ and therefore $\nvdash_{\mathsf{EHML}} \varphi$.

Observe that the algorithm described above always terminates, since at each iteration of Step 3 the cardinality of X strictly decreases. □

Remark 5.3.9. A natural next step in our investigation would be to consider the model checking problem for EHML. However, as addressing this issue falls beyond the scope of the present work, we confine ourselves to some general considerations. Given a finite EHML-model $\mathcal{M} = \langle \mathcal{S}, v \rangle$ and an EHML-formula φ, the most immediate approach to determining whether there exists a state s of $\mathcal{S}$ such that $\mathcal{M}, s \Vdash \varphi$ is to build on standard model checking procedures for normal multi-modal logics [87]. We outline below what we take to be a plausible strategy. Inevitably, the possible occurrence of delimited subformulas within EHML-formulas leads us to proceed by cases:

1. If $\mathrm{sub}_+(\varphi) = \emptyset$, we check whether φ is satisfied by $\phi(\mathcal{T})$, where $\mathcal{T}$ is the underlying LTS of $\mathcal{M}$ and $\phi\colon \mathfrak{LTS} \to \mathfrak{K}$ is the map defined in Section 5.1.

2. If $\mathrm{sub}_+(\varphi) = \{\lfloor\beta_1\rfloor, \ldots, \lfloor\beta_n\rfloor\}$, we proceed in two steps:

   2.1 For each state s of $\mathcal{S}$ and each $1 \leq k \leq n$, we check whether $\mathcal{M}_s, v(s) \Vdash \beta_k$. This yields a labelling of the states of $\mathcal{S}$ with formulas of the form $\lfloor\beta_k\rfloor$ or $\neg\lfloor\beta_k\rfloor$, depending on whether β_k is satisfied in the corresponding $\mathsf{S5}_n$-models;

   2.2 We then check whether $\varphi^{\mathtt{p}}$ is satisfied by $\phi(\mathcal{T})$, where $\varphi^{\mathtt{p}}$ is obtained from φ by uniformly replacing each occurrence of a delimited subformula with a placeholder atomic proposition. Satisfaction of placeholders is defined according to the labelling given in the previous step, i.e., an atomic proposition p_k corresponding to $\lfloor\beta_k\rfloor$ is true at a state s of $\mathcal{S}$ if and only if $\mathcal{M}_s, v(s) \Vdash \beta_k$.

5.4 A Hennessy–Milner theorem for EHML

EHML brings about the following notion of behavioral equivalence.

Definition 5.4.1 (Bisimulation)**.** Let $\mathcal{M} = \langle \mathcal{S}, v \rangle$ be a model over a KLTS $\mathcal{S} = \langle \mathcal{T}, At, Ag, r \rangle$ with $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. An equivalence relation $\mathscr{B} \subseteq S \times S$ is a *bisimulation* iff whenever $\langle s, t \rangle \in \mathscr{B}$ it holds that:

1. $v(s) = v(t)$;
2. if $\langle s, a, s' \rangle \in T$ then $\exists t'.\, \langle t, a, t' \rangle \in T$ and $\langle s', t' \rangle \in \mathscr{B}$;
3. there exists an equivalence relation $\mathscr{B}_{st}$ between the worlds of the Kripke models $\mathcal{M}_s = \langle \mathcal{F}_s, \mathrm{id} \rangle$ (with $\mathcal{F}_s$ pointed at $v(s)$) and $\mathcal{M}_t = \langle \mathcal{F}_t, \mathrm{id} \rangle$ (with $\mathcal{F}_t$ pointed at $v(t)$) such that $\langle v(s), v(t) \rangle \in \mathscr{B}_{st}$ and for any $X, Y \in \wp(At)$, whenever $\langle X, Y \rangle \in \mathscr{B}_{st}$ then:
   - $X = Y$;
   - if $\langle X, X' \rangle \in r_s(i)$ for $i \in Ag$, then $\exists Y'.\, \langle Y, Y' \rangle \in r_t(i)$ and $\langle X', Y' \rangle \in \mathscr{B}_{st}$.

Conditions 1 and 3 resemble the definition of modal bisimulation for Kripke models [26], while condition 2 characterises the strong bisimulation for LTSs [90]. Two states s and t of a model $\mathcal{M} = \langle \mathcal{S}, v \rangle$ are bisimilar, written $s \sim_{\mathcal{M}} t$, if and only if there exists a bisimulation $\mathscr{B}$ such that $\langle s, t \rangle \in \mathscr{B}$. The correspondence theorem relates bisimilar states and equivalent states whenever the model is *image-finite*, i.e., for all states and actions, the image of s (under any accessibility relation) and the image of s, π (under the transition relation) are finite.

Definition 5.4.2. Let $\mathcal{M} = \langle \mathcal{S}, v \rangle$ be an EHML-model. Two states s, s' of $\mathcal{S}$ are *modal equivalent* (denoted $s \equiv_{\mathcal{M}} s'$) if, for any EHML-formula φ:

$$\mathcal{M}, s \Vdash \varphi \quad \text{iff} \quad \mathcal{M}, s' \Vdash \varphi.$$

It is easy to check that $\equiv_{\mathcal{M}}$ is an equivalence relation.

Theorem 5.4.3. *For any two states s, t of a image-finite model $\mathcal{M}$, $s \sim_{\mathcal{M}} t$ iff $s \equiv_{\mathcal{M}} t$.*

Proof. The proof is an adaptation of standard approaches [26, 90].

Left-ro-right implication: $s \sim_{\mathcal{M}} t$ implies $s \equiv_{\mathcal{M}} t$.
Let us assume that there exists a bisimulation $\mathscr{B}$ including the pair $\langle s, t\rangle$. We will show the result through an induction on the complexity of formulas.

The base case refers to the atomic formulas and trivially holds by definition of bisimulation since $v(s) = v(t)$. We leave it to the reader to prove that no EHML-formula whose main connective is Boolean, and no delimited internal formula $\lfloor\psi\rfloor$ such that ψ's main connective is Boolean, can distinguish bisimilar states.

In the case of a formula $\langle\pi\rangle\varphi$, suppose that $\mathcal{M}, s \Vdash \langle\pi\rangle\varphi$. Hence there exists $s' \in S$ such that $\langle s, \pi, s'\rangle \in T$ and $\mathcal{M}, s' \Vdash \varphi$. By Definition 5.4.1, there exists $t' \in S$ such that $\langle t, \pi, t'\rangle \in T$ and $\langle s', t'\rangle \in \mathscr{B}$. By the induction hypothesis (IH), it holds that $\mathcal{M}, t' \Vdash \varphi$ whence $\mathcal{M}, t \Vdash \langle\pi\rangle\varphi$, and therefore s and t cannot be distinguished by EHML-formulas prefixed by dynamic modalities.

It remains to be checked the case of delimited internal formulas of the form $\lfloor\mathbf{K}_i\psi\rfloor$. Suppose that $\mathcal{M}, s \Vdash \lfloor\mathbf{K}_i\psi\rfloor$. Thus, by Definition 5.2.6, $\mathcal{M}_s, v(s) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i\psi$. By Definition 5.4.1, there exists an equivalence relation $\mathscr{B}_{st}$ between the worlds of $\mathcal{M}_s = \langle\mathcal{F}_s, \mathrm{id}\rangle$ (with $\mathcal{F}_s$ pointed at $v(s)$) and $\mathcal{M}_t = \langle\mathcal{F}_t, \mathrm{id}\rangle$ (with $\mathfrak{F}_t$ pointed at $v(t)$) such that $\langle v(s), v(t)\rangle \in \mathscr{B}_{st}$.

We now show that $\mathcal{M}_s$ and $\mathcal{M}_t$ satisfy the same internal formulas and, therefore, $\mathcal{M}_t, v(t) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i\psi$, from which we derive $\mathcal{M}, t \Vdash \lfloor\mathbf{K}_i\psi\rfloor$.

- *Base case.* As $v(s) = v(t)$, $\mathcal{M}_s, v(s) \Vdash_{\mathsf{S5}_n} p$ iff $\mathcal{M}_t, v(t) \Vdash_{\mathsf{S5}_n} p$, for all $p \in At$.

- *Step.* Trivially, no formula constructed using only Boolean connectives can distinguish $v(s)$ from $v(t)$. So, let us consider $\mathcal{M}_s, v(s) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i\psi$. Now, let X be such that $\langle v(s), X\rangle \in r_s(i)$. Hence we have $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \psi$. By Definition 5.4.1, there exists Y such that $\langle v(t), Y\rangle \in r_t(i)$ with $\langle X, Y\rangle \in \mathscr{B}_{st}$ and, by induction hypothesis, $\mathcal{M}_t, Y \Vdash_{\mathsf{S5}_n} \psi$. As a consequence, again by Definition 5.4.1, $\mathcal{M}_t, v(t) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i\psi$, as expected.

Therefore, having a bisimulation relating two states is sufficient for the two states to verify the same EHML-formulas.

Right-to-left implication: $s \equiv_{\mathcal{M}} t$ implies $s \sim_{\mathcal{M}} t$.
We show (by contradiction) that $\equiv_{\mathcal{M}}$ itself is a bisimulation.

First, the condition $v(s) = v(t)$ trivially holds because its violation would mean that there exists a formula distinguishing the two states.

Second, take an arbitrary action π and suppose that there exists s' such that $\langle s, \pi, s' \rangle \in T$, but there does not exist t' such that $\langle t, \pi, t' \rangle \in T$ with $s' \equiv_{\mathcal{M}} t'$. Let S' be the finite set of states accessible from t through a π-labeled transition. S' is non-empty otherwise $\langle \pi \rangle \top$ would distinguish s from t. By assumption, for each $t_j \in S'$, $1 \leq j \leq |S'|$, there exists φ_j such that $\mathcal{M}, s' \Vdash \varphi_j$ and $\mathcal{M}, t' \nVdash \varphi_j$. Hence, it holds that $\mathcal{M}, s \Vdash \langle \pi \rangle \bigwedge_j \varphi_j$ (since there is s' such that $\langle s, \pi, s' \rangle \in T$ and satisfying $\bigwedge_j \varphi_j$), and $\mathcal{M}, t \nVdash \langle \pi \rangle \bigwedge_j \varphi_j$, thus contradicting the hypothesis. The same kind of reasoning applies to the symmetric case.

Third, assume that the worlds of the Kripke models $\mathcal{M}_s = \langle \mathcal{F}_s, \mathrm{id} \rangle$ (with $\mathcal{F}_s$ pointed at $v(s)$) and $\mathcal{M}_t = \langle \mathcal{F}_t, \mathrm{id} \rangle$ (with $\mathcal{F}_t$ pointed at $v(t)$) are not related by a binary equivalence relation $\mathscr{B}_{st}$ including $\langle v(s), v(t) \rangle$ as given in Definition 5.4.1. As a first consideration, note that $v(s) = v(t)$, hence there exists X such that $\langle v(s), X \rangle \in r_s(i)$, with $i \in Ag$, but there does not exist Y such that $\langle v(t), Y \rangle \in r_t(i)$ with $\langle X, Y \rangle \in \mathscr{B}_{st}$. Let S' be the finite set of worlds accessible from $v(t)$ through $r_t(i)$. S' is non-empty, otherwise $\mathbf{K}_i \neg \top$ would distinguish $v(s)$ from $v(t)$. By assumption, for each $Y_j \in S'$, $1 \leq j \leq |S'|$, there exists ψ_j such that $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \psi_j$ and $\mathcal{M}_t, Y \nVdash_{\mathsf{S5}_n} \psi_j$. But then $\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \neg \mathbf{K}_i \neg \bigwedge_j \psi_j$ (since there is X, accessible from $v(s)$, satisfying $\bigwedge_j \psi_j$) and $\mathcal{M}_t, v(t) \nVdash_{\mathsf{S5}_n} \neg \mathbf{K}_i \neg \bigwedge_j \psi_j$ (since by assumption and the semantics of the epistemic operator, $\mathcal{M}_t, v(t) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i \neg \bigwedge_j \psi_j$), thus contradicting the hypothesis. The same kind of reasoning applies to the symmetric case.

As $\equiv_{\mathcal{M}}$ satisfies the three conditions of Definition 5.4.1, it is a bisimulation and the proof is completed. □

5.5 Related work

Computational modeling and temporal logics The closest approach to our framework is proposed in [100], where LTSs are enriched with accessibility relations representing indistinguishability between states. The idea is that agents observe (do not control) the path of performed actions and, based on this knowledge, deduce what the actual state is. Hence, the semantics of the formulas of the underlying logic is given in terms of paths. Notably, such a logic, like EHML, is equipped with both temporal and epistemic modalities. An analogous approach is followed in [52] by using an epistemic μ-calculus with an application to security properties. Similarly, epistemic notions are incorporated in concurrent constraint programming paradigms for modeling knowledge

distribution in multi-agents systems [101, 85].

In the field of concurrency theory, some of the ideas presented here can be found in the study of temporal logics encompassing features from HML and modal μ-calculus [51]. As an example, a variant of CTL is defined in [161] to check properties over expressive models called L^2TSs, which are defined as extensions of Kripke structures with labellings over transitions. In these models, the idea is to combine transition labels expressing the action-based dynamic behavior of a system with state-based labels expressing the knowledge possessed in each state of the system. With respect to our proposal, no epistemic representation of derivable knowledge is given, so the study of the observational power of the agents is limited to the verification of state-based propositional logic formulas and on the model checking of temporal formulas.

Combining logics Within the broad framework of combining logics [34], EHML can be viewed as a kind of *parameterisation* (in the sense of [32]) of HML with $\mathsf{S5}_n$ as the parameter logic. In particular, although HML is not a temporal logic, the design of EHML and its KLTS-based semantics is related to the method of temporalising a logic introduced by Finger and Gabbay [63]. In that work, starting from a propositional temporal logic TL and an arbitrary propositional logic S (with a classical base), a system TL(S) is obtained by internalising S into TL, along lines closely analogous to the construction of EHML developed here. More specifically, Finger and Gabbay partition the set of S-formulas into two sets:

- *Boolean combinations*: the set of those S-formulas whose main connective is a Boolean operator;
- *monolithic formulas*: the complement of the set of Boolean combinations relative to the set of all S-formulas.

The set of TL(S)-formulas is then generated over the set of monolithic formulas by means of the Boolean connectives and the temporal operators of TL. The authors argue that the reason why the syntax of TL(S) is built through monolithic formulas is that a clause of the form

$$\text{if } \varphi \text{ is an S-formula, then } \varphi \text{ is a TL(S)-formula} \qquad (P)$$

would conflict with the closure of TL(S)-formulas under the application of Boolean connectives, insofar as the depth measure of formulas' parsing trees would no longer be

well defined. For instance, if φ is an S-formula, the parsing tree of the $\mathsf{TL}(\mathsf{S})$-formula $\neg\varphi$ would have depth 0 if $\neg\varphi$ is viewed as constructed via (P), whereas it would have depth 1 if $\neg\varphi$ is viewed as the $\mathsf{TL}(\mathsf{S})$-formula obtained by applying negation to the S-formula φ. The syntax of EHML in Definition 5.2.2 does not face a similar issue, due to delimitation of internal formulas. Finally, while Finger and Gabbay establish completeness and decidability for logics whose temporal component is linear, our semantics, although not temporal, is intrinsically branching, reflecting the possibly nondeterministic structure of LTSs.

Dynamic epistemic logics A well-known dynamic approach for epistemic models is represented by the dynamic epistemic logics (see, e.g., [165]). The effect of dynamic behaviors on Kripke models is represented, e.g., through a modal operator of the form $[\mathit{condition}]\varphi$. In particular, this operator is used to express that φ is true after the occurrence of a certain *condition*. The condition could be represented by an action or by a formula contributing to updating the Kripke model. Hence, φ is evaluated to true in the new version of the model rather than the current one. This dynamic mechanism characterises, e.g., the Public Announcement Logic PAL [18] and paves the way for reasoning about extensions of the notion of possessed knowledge for dynamic multi-agent systems, like, e.g., common and/or distributed knowledge.

In PAL, the *condition* is a formula γ, so that $[\gamma]\varphi$ is true in a state of a Kripke model if, whenever γ is true in the state, φ is true in the state after we eliminate all not-γ states and related arrows. The intuition is that γ represents a public announcement that enriches the knowledge possessed by all agents, thus motivating the elimination of all non-γ possibilities from consideration. This behaviour is similar to the broadcast mechanism implemented in the process algebra presented in Chapter 6

The logic DEL [162, 19, 28] is based on classical epistemic models integrated with *event models* which, in particular, define events and pre/post-conditions to the events. Temporal evolution is then computed from some initial epistemic model through a process of successive *product updates*. A product update states that an event may occur in a state satisfying the related pre-condition and causing a state update governed by the post-condition. Several variants of this approach are possible. For instance, based on the same event model of DEL, the proposal in [145] presents a framework encompassing explicitly a timekeeping relation modelling the passage of time. Event models and product updates model certain constraints (e.g., the relation between an event and the

formula expressing its pre/post-condition) while ignoring others (e.g., the precedence relation between events).

The logic ETL [162] describes how knowledge evolves over time, which is represented as a history of events. Hence, the perspective of the agents about knowledge refers to their capability of distinguishing histories. The logic of [167] mixes the epistemic operator **K** and the classical temporal operators in a multi-agent setting in which agents implement strategies; however, this framework is not action-based.

With respect to these approaches, in the KLTS model, the system dynamics rely on the action-based LTS model, with a Kripke model that is linked to every state of the LTS, and with no constraints between the two levels expressed in the form of pre/post-conditions. Instead, we specify the constraints at the level of the semantics of our process algebra (see Chapter 6), which we use to explicitly build concurrent processes and to implicitly encode the dynamics of Kripke frames within the execution of the actions.

In particular, as we shall see, our specification language models action-based synchronous communications between agents, which result in a transfer of knowledge within the Kripke models. These updates are based on operations that delete arrows (instead of eliminating possible worlds), similar as done in approaches, like, e.g., [102], or in a more general way, in [77]. However, we can also model the loss of knowledge—and therefore arrow addition—caused by the (re-)setting of propositions. The interaction mechanism of our framework is simulated by the action model of [18], where, however, the point-to-point communication is represented artificially as two announcements with different accessibility.

A natural benefit of our LTS-based semantics for modelling the dynamics of systems is that we immediately inherit the model-checking techniques associated with discrete-time models and dynamic logics in HML style. Moreover, in EHML, these capabilities are merged with the expressive power of epistemic logic.

Chapter 6

A process algebra with KLTS semantics

In this chapter, we adopt an incremental approach to present a KLTS-based specification language that we call ECCS (*Epistemic Calculus of Communicating Systems*). We begin by defining a process calculus inspired by classical CCS-style calculi with value passing [64, 80, 89, 94]. This calculus provides the basis for describing sequential process terms in isolation, and its semantics is given in terms of LTSs. We then introduce the notion of an ECCS-*agent*, which has a unique identity and is characterised by a behaviour specified as a process term. Finally, we define *networks* (*pools*) of communicating epistemic ECCS-agents, whose semantics is expressed in terms of KLTSs. The various components and stages of our specification methodology are illustrated by a running example introduced below.

Running example. The FIPA (Foundation for Intelligent Physical Agents) standard specifications [141, 140] describe basic interaction protocols for multi-agent systems that support interoperability, information exchange, and open service interaction [138]. Hence, they represent an ideal setting for the description of our methodology. For each protocol description, we have (at least) an agent sending requests, called Initiator, and (at least) an agent executing tasks, called Participant. In the following, we will describe the behaviour of agents for two FIPA protocol standards:

- *Query interaction protocol* (*QIP*): the Initiator wants to query whether a proposition p is true or false and sends a request of type *query-if* to the Participant.

Then, the Participant processes the request, makes a decision whether to accept or refuse it, stores the decision in a Boolean variable, and then notifies the Initiator of the value of the variable. In case of acceptance, the Participant may fail or reply with an *inform-t/f* communication that asserts the truth or falsehood of the proposition.

- *Contract net interaction protocol* (*CNIP*): the Initiator requires a service with certain requirements and issues a call soliciting proposals from a set of Participants offering the needed task. Each Participant may answer with a proposal that includes preconditions set out for the task. Until a given deadline, the Initiator waits for proposals, evaluates the received ones, and selects the Participant to perform the task. Finally, the chosen Participant communicates the result of the task.

6.1 Process terms of ECCS

ECCS is based on a syntax determined by elementary terms, called actions, and the well-formed combination of operators generating more complex terms, which are considered to be correctly structured programs that execute actions. For this reason, we call the terms of our calculus *process terms*.

Let A_τ be a set of action names (ranged over by $a, b, \ldots$), including the distinguished name τ, which represents an unobservable internal action. We write A for $A_\tau \setminus \{\tau\}$. Moreover, let *set* and *in* be two additional special action names. To model value passing [89, 94], we will use variables $x, y, \ldots, f, g, \ldots$, values $v, v', \ldots$ from fixed domains, and expressions $e, e', \ldots$ that usually represent simple values.

We model sequential processes through the operators of *action prefix*, *nondeterministic choice*, and *recursion*.

Definition 6.1.1. For At a fixed set of atomic propositions and Ag a set of agent variables, the set $Proc$ of *process terms* of the calculus for sequential processes is generated through the following syntax:

$$P ::= \underline{0} \mid \textstyle\sum_{k \in K} \pi_k \,.\, P_k \mid C(e_1, \ldots, e_n)$$
$$\pi ::= [\psi]b \mid a(y, f) \mid \bar{a}(J, \psi) \mid set(p, w)$$

where K is a finite indexing set, C is a constant name with the integer $n \geq 0$ being its arity, $b \in A_\tau$, $\psi \in Int$, $a \in A$, $J \subseteq Ag$, $p \in At$, and w is a Boolean value.

Let us explain the informal meaning of each operator. The constant $\underline{0}$ stands for the inactive, halted process. The summation operator represents a nondeterministic choice enacting one of the process terms $\pi_k.P_k$, with $k \in K$, which executes action π_k and then behaves as process term P_k. The constant C is used to express recursive processes with $n \geq 0$ parameters and must be associated with a defining equation of the form $C(x_1, \ldots, x_n) := P$.

The notation π stands for any action of one of the following forms:

- The *conditional action* $[\psi]b$ represents an action (named b) guarded by the internal formula ψ. As will be made precise in the next section, the intended interpretation of $[\psi]b$ is that the action b is executed whenever ψ is known by the agent. This notation should not be confused with the modality $[\pi]\varphi$ of HML, which expresses that every execution of action π leads to a state satisfying φ. In the following, we write b as shorthand for $[\top]b$.

- The *input action* $a(y, f)$ specifies that, when executing the action named a, an internal formula assigned to the variable f is received from an agent with identity assigned to the variable y.

- The *output action* $\bar{a}(J, \psi)$ specifies that when executing the action named a the formula ψ is sent to all the agents in the set J (*broadcast communication*). If $|J| = 1$ then we have a classical one-to-one interaction (in this case, we will use the simplified notation $\bar{a}(j, \psi)$). As we will see formally in the next Section, only known formulas can be transmitted to other agents.

- The *assignment action* $set(p, w)$ sets the proposition p to the Boolean value w.

As usual in calculi with value passing, each variable occurrence in a process term P is bound either by an input action or by a defining equation for a constant. For instance, x is bound in $C(x) := \bar{a}(x, p \wedge q) \,.\, C(x + 1)$ and in $a(x, f) \,.\, \bar{b}(x, \top) \,.\, \underline{0}$, but not in $\bar{a}(x, p \wedge q) \,.\, \underline{0}$. Moreover, we write i/x and ψ/f for substitutions of values for variables, and denote by $P[i/x, \psi/f]$ the result of substituting i (resp., ψ) for all free (not bound) occurrences of x (resp., f) in P.

Definition 6.1.2. The *behaviour* of a process term $P \in Proc$ is described in structural operational semantics style as the LTS rooted at P and defined by the least transition relation generated by the axioms and the rules of Table 6.1.

$$\pi \,.\, P \xrightarrow{\pi} P \quad \pi \in \{[\psi]b, \bar{a}(i,\psi), set(p,w)\}$$

$$a(y,f) \,.\, P \xrightarrow{a(i,\psi)} P[i/y, \psi/f] \quad \text{for all } i \in Ag \text{ and all } \psi \in Int$$

$$\frac{\pi \,.\, P \xrightarrow{\pi'} P}{\pi \,.\, P + E \xrightarrow{\pi'} P}$$

$$\frac{P[v_1/x_1, \ldots, v_n/x_n] \xrightarrow{\pi} P'}{C(e_1, \ldots, e_n) \xrightarrow{\pi} P'} \quad C(x_1, \ldots, x_n) := P \text{ and each } e_i \text{ evaluates to } v_i$$

Table 6.1: Semantic rules for sequential processes

Let us illustrate the semantic rules of Table 6.1. The first rule formalises the behaviour of the prefix operator for actions other than input actions. In particular, the rule specifies that such actions are enabled independently of any pre- or post-conditions on their parameters. Pre- and post-conditions will be introduced only when defining the semantics of the parallel composition of agents and the structures characterising the knowledge of these agents.

The second rule describes the behaviour of input actions. Note that, the process term enables the execution of a bunch of transitions that represent any possible assignment of the incoming internal formula and of the sending agent.

In the rule for the choice operator, E is a non-empty summation. Hence, if a prefix process term can execute an action π' and evolve to P (see the rule's premiss), then a summation including such a process term can execute π' and evolve to P, thus discarding the context E (see the rule conclusion). Hence, the operator models the standard nondeterministic choice among alternative process terms.

Finally, the recursion mechanism is defined as in classic value-passing CCS [80, 94].

6.2 From process terms to ECCS-agents

Process terms in the calculus defined above describe behavioural patterns. An ECCS-agent is a specific instance of a process term, equipped with a unique identity.

Definition 6.2.1. An ECCS-*agent* is a pair $\mathtt{A} = \langle i, P \rangle$, where $i \in Ag$ and $P \in Proc$. We refer to P as the *local behaviour* of agent i.

The semantics of an ECCS-agent $\langle i, P \rangle$ is given by the LTS expressing the behaviour of P, up to the renaming defined by the following semantic rule:

$$\frac{P \xrightarrow{\pi} P'}{\langle i, P \rangle \xrightarrow{i.\pi} \langle i, P' \rangle}$$

In the following examples, we describe the behavioural pattern of each type of agent involved in the two FIPA protocols, as well as two possible scenarios. Regarding the parameters of input/output actions, we will use $\top$ as a parameter of an output action whenever no specific formula must be communicated, and we use the symbol _ to denote an unspecified parameter of an input action whenever it is not used.

Example 6.2.2 (Specification of the FIPA QIP)**.** We assume that queries can be issued in relation to a finite set of propositions Pr. Then, the process term *Participant* can be defined as follows:

$$\begin{array}{rcl} Participant & := & \sum_{p \in Pr} \mathit{query\text{-}if}_p(x, \_).(\\ & & \quad \mathit{set}(acc, 0).\overline{\mathit{notify}}(x, \neg acc).Participant + \\ & & \quad \mathit{set}(acc, 1).\overline{\mathit{notify}}(x, acc).Query\_Exec_p(x)) \\ Query\_Exec_p(x) & := & \overline{\mathit{failure}}(x, \top).Participant + \\ & & \overline{\mathit{inform\text{-}t/f}}(x, p).Participant + \overline{\mathit{inform\text{-}t/f}}(x, \neg p).Participant \end{array}$$

The choice among the input actions named $\mathit{query\text{-}if}_p$, with $p \in Pr$, states that query requests about any proposition can be received. Then, a nondeterministic choice between two different assignments determines whether the request is accepted ($acc = 1$) or refused ($acc = 0$). The decision is sent to the same agent x that originated the query. If the query is executed, it can nondeterministically fail (see the action named *failure*) or return the value of proposition p as an outcome (see the two actions named

inform-t/f)). The process term *Initiator* is defined as follows:

$$\begin{array}{rcl} \mathit{Initiator} & := & \sum_{p \in Pr} \overline{\mathit{query\text{-}if}_p}(\mathit{MySQL\_Server}, \top).\mathit{notify}(\_, g_1).\mathit{Init\_Wait} \\ \mathit{Init\_Wait} & := & [\neg \mathit{acc}]\mathit{is\_refuse}.\mathit{Initiator} + [\mathit{acc}]\mathit{is\_agree}. \\ & & (\mathit{failure}(\_, \_).\mathit{Initiator} + \mathit{inform\text{-}t/f}(\_, g_2).\mathit{ok}.\mathit{Initiator}) \end{array}$$

The choice of the specific query is nondeterministic and is sent to the agent *MySQL_Server*, which is assumed to be the identity of the Participant of interest. Based on the formula g_1 received with the notification, acceptance (action *is_agree*) or rejection (action *is_refuse*) is enabled. In case of acceptance and subsequent success, the result of the query is received through the formula g_2 and the action *ok* is executed.

As a possible topology of a system executing such a FIPA protocol, consider a very simple scenario with a Participant *MySQL_Server* and an Initiator *Web_App*. Hence, we have the two ECCS-agents:

$$\langle \mathit{Web\_App}, \mathit{Initiator} \rangle \qquad \langle \mathit{MySQL\_Server}, \mathit{Participant} \rangle.$$

Example 6.2.3 (Specification of the FIPA CNIP)**.** We assume a task offered by the Participant agents belonging to a set J. Hence, the Initiator model is as follows:

$$\begin{array}{rcl} \mathit{Initiator}(x, f_{req}) & := & \overline{\mathit{call}}(J, \top).\mathit{Init\_Wait}(x, f_{req}) \\ \mathit{Init\_Wait}(x, f_{req}) & := & \mathit{propose}(y, f_{pre}).(\\ & & \quad [\mathit{acc}_y]\mathit{verify}.\overline{\mathit{accept}}(y, \top).\mathit{Init\_Exec}(x, f_{req}) + \\ & & \quad [\neg \mathbf{K}_x \mathit{acc}_y]\mathit{verify}.\overline{\mathit{reject}}(y, \top).\mathit{Init\_Wait}(x, f_{req})) + \\ & & [\psi_t]\mathit{timeout}.\mathit{Initiator}(x, f_{req}) \\ \mathit{Init\_Exec}(x, f_{req}) & := & \mathit{fail}(\_, \_).\mathit{Init\_Wait}(x, f_{req}) + \mathit{result}(\_, g).\mathit{ok}.\mathit{Initiator}(x, f_{req}) \end{array}$$

Process *Initiator* is parameterised by the identity x of the agent and by the formula f_{req} expressing the task requirements. The call for proposals is modelled as a broadcast to the set J of participants (see the action named *call*), after which the process term waits for proposals (see action *propose*) or the expiration of a timeout (modelled by a formula ψ_t guarding action *timeout*). A proposal from a participant agent y comes with task preconditions modelled by the formula f_{pre}. Depending on f_{req}, f_{pre}, and the knowledge of the initiator agent (which we will specify in Example 6.3.7), the proposal can be accepted (which is modelled by proposition acc_y). In such a case, the guard $[\mathit{acc}_y]$ enables the branch leading to acceptance. Contrariwise, if it is not possible to derive acc_y, i.e., $\neg \mathbf{K}_x \mathit{acc}_y$, the branch leading to rejection is enabled. Once a proposal

has been accepted, it is possible to detect a task failure or receive the outcome of the task through the input formula g (see process $Init\_Exec$). Each Participant model is as follows:

$$\begin{aligned} Participant(f_{pre}) := \quad & call(x, \_).(\\ & \quad \overline{propose}(x, f_{pre}).Part\_Wait(f_{pre}) + \\ & \quad [\psi_t]timeout.Participant(f_{pre})) \\ Part\_Wait(f_{pre}) := \quad & reject(\_, \_).Participant(f_{pre}) + \\ & accept(x, \_).(\\ & \quad \overline{fail}(x, \_).Participant(f_{pre}) + \\ & \quad \overline{result}(x, \psi_r).Participant(f_{pre})) \end{aligned}$$

A Participant agent is parameterised by a formula f_{pre} modelling the offer preconditions. When receiving a call, the agent can make a proposal (see action named *propose*) or abandon the protocol through the same timeout mechanism described above. After sending a proposal, the agent waits for the Initiator's decision and, if the offer is accepted, the task is executed. Then, the execution may fail or produce an outcome ψ_r that is transmitted to the Initiator.

Now, let us consider a scenario with a set $J = \{S_1, S_2\}$ of two Participant agents, with preconditions p_1 and p_2 associated to S_1 and S_2, respectively. Moreover, we have one Initiator agent C, with task requirements modelled by the formula $\neg r_1 \wedge r_2$. Therefore, we have the set of ECCS-agents:

$$T := \{\langle C, Initiator(C, \neg r_1 \wedge r_2)\rangle, \langle S_1, Participant(p_1)\rangle, \langle S_2, Participant(p_2)\rangle\}.$$

6.3 Modelling concurrent ECCS-agents

To model concurrency, ECCS-agents must be able to communicate with each other to form a network. Moreover, while so far we described agents only in terms of their local behaviour, now we need to model their knowledge, which will affect the way in which they can behave and interact. In the following, we combine the behaviour of several agents by integrating the notion of knowledge, which will allow us to specify how they can interact.

Definition 6.3.1. Let $I \subseteq Ag$ be a finite set of cardinality n. A *pool of* ECCS*-agents* is a tuple $\mathscr{P} = \langle \{\mathtt{A}_i\}_{i\in I}, \{R_i\}_{i\in I}, X\rangle$, where $\{\mathtt{A}_i\}_{i\in I}$ is a family of ECCS-agents, $\{R_i\}_{i\in I}$

is a family of equivalence relations over $\wp(At)$, and $X \subseteq At$ represents the valuation over At that makes true all and only the atomic propositions in X.

The behaviour of the set of ECCS-agents forming the pool depends on the local behaviour of each agent i. Each equivalence relation R_i represents the epistemic structure governing agent i's ability to distinguish possible worlds on the basis of truth-value assignments over At. Finally, X denotes the set of atomic propositions that are taken to be true with respect to $\{\mathtt{A}_i\}_{i\in I}$ (see Definition 6.3.2).

The agents of a pool execute their actions, either synchronously or autonomously, thus making the system dynamic. In particular, input and output actions represent synchronous communications that express knowledge transfer between agents. All the other actions are executed autonomously. In any case, the execution of an action might (1) be conditioned to the knowledge of some formula and (2) imply some change in X and the agents' knowledge.

Formally, such a joint knowledge-based and action-based behaviour is represented by a model over a KLTS describing the evolution of the pool of agents.

Definition 6.3.2. Let $\mathscr{P} = \langle \{\mathtt{A}_i\}_{i\in I}, \{R_i\}_{i\in I}, X\rangle$ be a pool of ECCS-agents of cardinality n. The semantics of $\mathscr{P}$ is given by a model $\mathcal{M} = \langle \mathcal{S}, v\rangle$ over a KLTS $\mathcal{S} = \langle \mathcal{T}, At, I, r\rangle$ with underlying LTS $\mathcal{T} = \langle S, Act, T, s_0\rangle$ such that:

- the states in S are pools of agents, with $s_0 = \mathscr{P}$ and $v(s_0) = X$. For each $i \in I$, $r_{s_0}(i) = R_i$ is the accessibility relation associated to i in s_0;
- T is the least transition relation generated by the rules of Table 6.2;
- for each $s \in S$ corresponding to a pool $\langle \{\mathtt{A}'_i\}_{i\in I}, \{R'_i\}_{i\in I}, X'\rangle$, we define $v(s) = X'$ and for each $i \in I$, $r_s(i) = R'_i$.

Intuitively, the pool tuple $\mathscr{P}$ defines the initial state of a KLTS model that is built by applying the rules of Table 6.2. Each new pool tuple added in such a way includes the information used to define the valuation function v and the Kripke frames associated with the KLTS states. We now illustrate the rules of Table 6.2.

The rule (*pool*) describes the asynchronous execution of an autonomous action of the form $[\psi]b$ by any agent j of the pool – see the second premiss of the rule. The action, named b, is subject to the condition given by ψ, meaning that it is enabled only if ψ is known by the agent j – see the first premiss of the rule. In particular, the knowledge

Let:

– $\mathcal{M}_s = \langle\langle \wp(At), I, \{R_i\}_{i\in I}\rangle, \mathrm{id}\rangle$ for $s = \langle \_, \{R_i\}_{i\in I}, \_\rangle$ a pool of agents $\mathscr{P}$

– $\mathit{diff}(\mathcal{M}_s, X, Y, \psi) = (\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \psi \curlywedge \mathcal{M}_s, Y \nVdash_{\mathsf{S5}_n} \psi) \curlyvee (\mathcal{M}_s, X \nVdash_{\mathsf{S5}_n} \psi \curlywedge \mathcal{M}_s, Y \Vdash_{\mathsf{S5}_n} \psi)$

– $\mathit{closure}(R_k) = R_k \cup \{\langle X, Y\rangle \mid \exists Z.\, \langle X, Z\rangle \in R_k \curlywedge \langle Z, Y\rangle \in R_k\}$

$$(\mathit{pool}) \qquad \frac{\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \mathbf{K}_j\psi \quad \mathtt{A}_j \xrightarrow{j.[\psi]b} \mathtt{A}'_j}{\mathscr{P} \xrightarrow{j.b} \mathscr{P}'} \qquad \text{where:}$$

– $\mathscr{P} = \langle\{\mathtt{A}_j\} \cup \{\mathtt{A}_i\}_{i\in I\setminus\{j\}}, \{R_i\}_{i\in I}, X\rangle$

– $\mathscr{P}' = \langle\{\mathtt{A}'_j\} \cup \{\mathtt{A}_i\}_{i\in I\setminus\{j\}}, \{R_i\}_{i\in I}, X\rangle$

$$(\mathit{set}) \qquad \frac{\mathtt{A}_j \xrightarrow{j.set(p,w)} \mathtt{A}'_j}{\mathscr{P} \xrightarrow{\tau} \mathscr{P}'} \qquad \text{where:}$$

– $\mathscr{P} = \langle\{\mathtt{A}_j\} \cup \{\mathtt{A}_i\}_{i\in I\setminus\{j\}}, \{R_j\} \cup \{R_i\}_{i\in I\setminus\{j\}}, X\rangle$

– $\mathscr{P}' = \langle\{\mathtt{A}'_j\} \cup \{\mathtt{A}_i\}_{i\in I\setminus\{j\}}, \{R'_j\} \cup \{R'_i\}_{i\in I\setminus\{j\}}, X'\rangle$

– $R'_j = R_j \backslash \{\langle Y, Y'\rangle \mid \mathit{diff}(\mathcal{M}_s, Y, Y', p)\}$

– $R'_i = \mathit{closure}(R_i \cup \{\langle\{p\} \cup Y, Y\rangle \mid p \notin Y\} \cup \{\langle Y, \{p\} \cup Y\rangle \mid p \notin Y\})$

– $X' = \begin{cases} X\setminus\{p\} & \text{if } w = 0 \\ X \cup \{p\} & \text{if } w = 1 \end{cases}$

$$(\mathit{com}) \qquad \frac{\mathcal{M}_s, X \Vdash_{\mathsf{S5}_n} \mathbf{K}_j\psi \quad \mathtt{A}_j \xrightarrow{j.\bar{a}(J,\psi)} \mathtt{A}'_j \quad \mathtt{A}_{i_1} \xrightarrow{i_1.a(j,\psi)} \mathtt{A}'_{i_1} \;\cdots\; \mathtt{A}_{i_n} \xrightarrow{i_n.a(j,\psi)} \mathtt{A}'_{i_n} \quad j \notin J}{\mathscr{P} \xrightarrow{\tau} \mathscr{P}'}$$

where:

– $\mathscr{P} = \langle\{\mathtt{A}_j\} \cup \{\mathtt{A}_{i_1}, \ldots \mathtt{A}_{i_h}\} \cup \{\mathtt{A}_k\}_{k\in I\setminus(\{j\}\cup J)}, \{R_{i_1}, \ldots, R_{i_h}\} \cup \{R_k\}_{k\in I\setminus J}, X\rangle$

– $\mathscr{P}' = \langle\{\mathtt{A}'_j\} \cup \{\mathtt{A}'_{i_1}, \ldots, \mathtt{A}'_{i_h}\} \cup \{\mathtt{A}_k\}_{k\in I\setminus(\{j\}\cup J)}, \{R'_{i_1}, \ldots, R'_{i_h}\} \cup \{R_k\}_{k\in I\setminus J}, X\rangle$

– $J = \{i_1, \ldots, i_h\} \subseteq I$

– $\forall i \in J.\ R'_i = R_i \backslash \{(Y, Y') \mid \mathit{diff}(\mathcal{M}_s, Y, Y', \psi)\}$

Table 6.2: Semantic rules for a pool of agents

of ψ by j is determined with respect to the Kripke model associated with the current pool tuple and the current truth assignment represented by X. Note that, the special case $\psi = \top$ denotes that no pre-conditions are necessary to perform the action. The resulting action is simply $j.b$ (as the knowledge of ψ by agent j is local to the agent), while both the knowledge structure and the truth assignment remain unchanged—see the conclusion of the rule.

The rule (*set*) describes the asynchronous execution of an autonomous action of the form $set(p, w)$ by any agent j of the pool. The side effect is that X is updated according to the assignment $p = w$ (see the definition of X'). The accessibility relations are also updated accordingly (see the definition of R'_j and R'_i, with $i \in I \backslash \{j\}$). On the one hand, the agent j performing the assignment acquires knowledge (if not yet possessed) of p. Hence, in R_j, all the possible worlds differing for the valuation of p (see function *diff*) cannot be mutually accessible anymore, as they are distinguishable by the value of p. Note that, as we will show, such suppression of connections ensures that the accessibility relation remains an equivalence. On the other hand, all the other agents $i \neq j$ lose knowledge (if previously possessed) of p, as the assignment is not considered public (as emphasised by the fact that the resulting action is a silent action τ). Therefore, in each accessibility relation of those agents, all the possible worlds differing only for the valuation of p must become mutually accessible, as they cannot be distinguished anymore. Note that such addition of connections considers the symmetric pairs and, through the *closure* operation, the transitive relations, thus ensuring, as we will show, that the accessibility relation remains an equivalence.

The most interesting rule is (*com*), which expresses a broadcast communication from an agent j performing the output action named a to the set of agents J, each of which performs the corresponding input named a [3] – see the premisses of the rule about the agent j executing the output containing the internal formula ψ and the agents in J receiving the formula through the corresponding input. The communication is synchronous, meaning that the agent j and all the agents in J advance simultaneously, as outlined in the conclusion of the rule. As in the case of the rule (*pool*), the communication is subject to a pre-condition related to the knowledge of the internal formula ψ by the agent j – see the first premiss of the rule. During the interaction, the knowledge of this formula is also transferred: all the agents in J acquire knowledge of ψ, and, as a consequence, all the accessibility relations of such agents are updated accordingly. In fact, they become able to distinguish those possible worlds that differ from each other

for the evaluation of ψ. Finally, the communication is private (the synchronisation result is a silent action τ), i.e., the knowledge transfer involves only the agents in J and no one else.

The given semantic rules allow us to preserve some interesting properties that are desirable for verification purposes.

Lemma 6.3.3. *Let $\mathcal{M} = \langle \mathcal{S}, v \rangle$ be a model for the semantics of a pool of agents. Then $\mathcal{S}$ is image-finite.*

This result immediately follows Definition 6.3.2 and the semantics of Table 6.2. As anticipated, another important result is that the semantics of Table 6.2 preserves the indistinguishability interpretation of the accessibility relations.

Theorem 6.3.4. *Let $\mathscr{P} = \langle \{\mathtt{A}_i\}_{i \in I}, \{R_i\}_{i \in I}, X \rangle$ be a pool of agents of cardinality n, with semantics given by a model $\mathcal{M} = \langle \mathcal{S}, v \rangle$ over a KLTS $\mathcal{S} = \langle \mathcal{T}, At, I, r \rangle$ with underlying LTS $\mathcal{T} = \langle S, Act, T, s_0 \rangle$. Then, for each $i \in I$ and for each $s \in S$, $r_s(i)$ is an equivalence relation.*

Proof. The proof proceeds by showing that given any state $s \in S$ such that, for each $i \in I$, $r_s(i)$ is an equivalence relation, then every state s' reachable from s through a transition $\langle s, \_, s' \rangle \in T$ satisfies the same condition. The result immediately follows from the fact that, in the initial state $s_0 = \mathscr{P}$, $\{R_i\}_{i \in I}$ is a family of equivalence relations.

By following the rules of Table 6.2, we first observe that the rules (*pool*) and (*in*) do not change any accessibility relation. Hence, the non-trivial cases are the rules (*set*) and (*com*), where the latter is a sub-case of the former as far as the updates to the relations are concerned. Hence, it is sufficient to show the effect of an action of the form $set(p, \_)$. Let us assume that the family $\{R_j\} \cup \{R_i\}_{i \in I \setminus \{j\}}$ associated with the state s is turned into the family $\{R'_j\} \cup \{R'_i\}_{i \in I \setminus \{j\}}$ when going from s to s'.

By definition, R'_j derives from R_j by deleting those pairs $\langle Y, Y' \rangle$ such that Y and Y' differ for the evaluation of p. Hence, it is easy to see that reflexivity and symmetry are preserved. Let us consider transitivity, by assuming that there exist Y, Y', Y'' such that $\langle Y, Y' \rangle, \langle Y', Y'' \rangle, \langle Y, Y'' \rangle \in R_j$ but only $\langle Y, Y' \rangle, \langle Y', Y'' \rangle \in R'_j$. Let $\langle Y, Y'' \rangle \notin R'_j$ since $p \in Y$ and $p \notin Y''$ (the opposite case is orthogonal). Then, $p \in Y'$ contradicts the assumption $\langle Y', Y'' \rangle \in R'_j$, while $p \notin Y'$ contradicts the assumption $\langle Y, Y' \rangle \in R'_j$. Hence, it cannot be that $\langle Y, Y' \rangle, \langle Y', Y'' \rangle \in R'_j$ and $\langle Y, Y'' \rangle \notin R'_j$.

By definition, for any i we have that R'_i derives from R_i by adding those pairs $\langle Y, Y' \rangle$ such that Y and Y' differ only for the evaluation of p. Hence, it is easy to see that reflexivity and symmetry are preserved. Let us consider transitivity, by assuming that there exist Y, Y', Y'' such that $\langle Y, Y' \rangle \in R_i$, $\langle Y', Y'' \rangle \notin R_i$ and $\langle Y, Y' \rangle, \langle Y', Y'' \rangle \in R'_i$. Hence, by the *closure* operation, we have that $\langle Y, Y'' \rangle \in R'_i$ too.

This concludes the proof for rule (*set*) and, therefore, the theorem holds. □

Corollary 6.3.5. *The semantics of a pool of agents is an* EHML*-model.*

Based on our running examples, we now describe the behaviours of the FIPA scenarios illustrated in Examples 6.2.2 and 6.2.3.

Example 6.3.6. Let us consider the QIP with two agents. We recall that the propositions of interest are *acc* and the set Pr of propositions that can be the subject of a query. The initial pool tuple is:

$$\mathscr{P}_0 = \langle \{ \langle Web\_App, Initiator \rangle, \langle MySQL\_Server, Participant \rangle \}, \{R_{WA}, R_{SQL}\}, X \rangle$$

where the accessibility relation R_{WA} of the agent Web_App is such that there is only one class to which all the propositions belong (initially, no proposition is known to Web_App); the accessibility relation R_{SQL} of the agent $MySQL\_Server$ contains only the reflexive pairs (i.e., each possible world is a singleton, meaning that the MySQL server knows the value of each proposition); the set X is such that *acc* is assigned the initial value 0 and each $p \in Pr$ the Boolean value corresponding to the current status of p in the database.

Figure 6.1 shows a portion of the model underlying the behaviour of this pool of agents in the case of a query related to proposition $p \in Pr$, which is assumed to be true in the database. For the sake of readability, the accessibility relations are not reported, and for each unobservable τ-transition we also indicate the name of the visible action(s) from which it derives. For instance, state $\mathcal{P}'_0$ is reached after the synchronization between the actions named $query\text{-}if_p$ and yields the same relations and valuation of state $\mathscr{P}_0$. Afterward, state $\mathscr{P}_1$ (resp., $\mathcal{P}_2$) is reached when $MySQL\_Server$ sets to 0 (resp., 1) the value of proposition *acc*, thus updating X accordingly. The following τ-transition, originated via the synchronization involving the actions named *notify*, allows Web_App to receive either the formula *acc* or the formula $\neg acc$, and then to execute either the action is_refuse or the action is_agree. Note that, before the notification, Web_App

did not know *acc*, while after the notification, the truth value of the proposition is known. In the case *MySQL_Server* executes the query, the choice between failure and the actions denoting success is nondeterministic, while the choice between the two actions named *inform-t/f* is deterministic and depends on the truth value of proposition p. In any case, a synchronization with *Web_App* occurs. Finally, we point out that from the standpoint of the agents' local behaviour, states $\mathscr{P}_0$, $\mathscr{P}_1''$, $\mathscr{P}_3$, and $\mathscr{P}_5$ would collapse to the same (initial) state. Instead, they differ in the knowledge of the two agents.

Based on the described model, it can be observed what follows:

- The EHML-formula $\langle\tau\rangle\langle\tau\rangle\langle\tau\rangle\langle is\_agree\rangle\langle\tau\rangle\lfloor\mathbf{K}_{WA}(p)\rfloor$ is satisfied in the initial state of the model, as shown by the path leading to $\mathscr{P}_4$, where *Web_App* has been informed that p is true.
- The EHML-formula

$$\langle ok\rangle\top \rightarrow \bigvee_{p\in Pr} \lfloor \mathbf{K}_{WA}(p) \vee \mathbf{K}_{WA}(\neg p)\rfloor$$

  is true in the model. In fact, if *Web_App* is ready to execute the action *ok* then it knows the truth value of at least one proposition in the database.
- $\mathscr{P}_0 \not\equiv \mathscr{P}_5$, because $\mathbf{K}_{WA}(p)$ holds in the latter state but not in the former state.
- The difference among $\mathscr{P}_0$, $\mathscr{P}_1''$, and $\mathscr{P}_3$ is given by what the two agents know about the value of *acc*. If such a variable were reset to zero by agent *MySQL_Server* via the *set* action at the end of each query instance, then the three states would collapse.

Example 6.3.7. Building on the topology presented at the end of Example 6.2.3, the initial pool tuple for the contract net interaction protocol with three agents is:

$$\mathscr{P}_0 = \langle T, \{R_C, R_{S_1}, R_{S_2}\}, X\rangle.$$

where $X := \{r_2, p_1, p_2\}$ denotes the task requirements and preconditions at hand. Each relation R_{S_i} is defined in such a way that S_i knows only the value of its own precondition p_i, while relation R_C is defined as follows: C knows $\neg r_1$ and r_2, which represent the two task requirements; C knows $r_1 \wedge p_1 \rightarrow acc_1$, $\neg r_1 \wedge p_3 \rightarrow acc_1$, and $r_2 \wedge p_2 \rightarrow acc_2$;

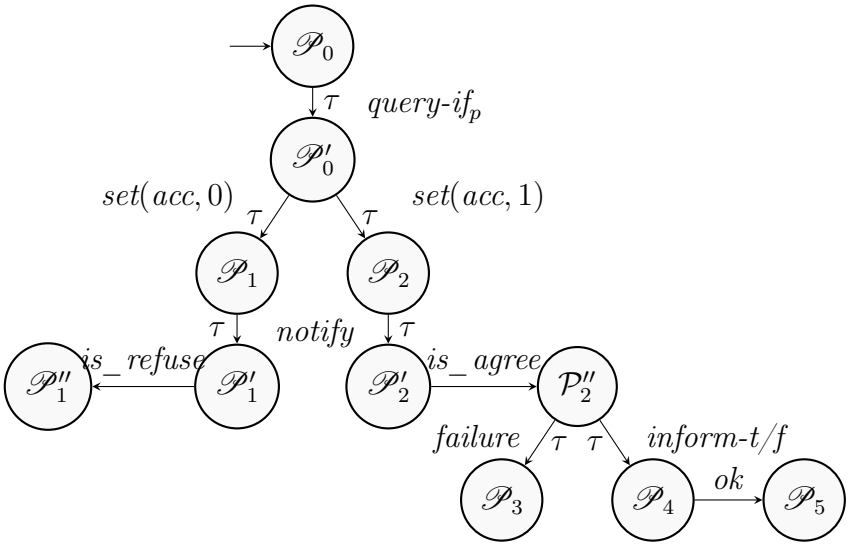


Figure 6.1: Portion of the KLTS underlying the pool of agents of the FIPA QIP.

C does not know anything else. Hence, it is easy to see that initially, C is not able to make any decision without issuing a call for proposals. Moreover, we are also assuming that formula ψ_t is not known to the agents, meaning that no timeout action will be enabled.

The evolution of the system starting from $\mathscr{P}_0$ can be followed in Figure 6.2.3. In particular, regarding the behaviour of the agents, the synchronization over the actions named *call* is an example of the application of the rule (*com*) in the case of broadcast communication. Hence, agent C synchronises simultaneously with the two agents S_1 and S_2 (state $\mathscr{P}'_0$). Afterwards, agent C can receive the two proposals. Let us consider the case in which the proposal from S_1 arrives first (see state $\mathscr{P}_1$ of the left-hand branch). At this stage, C learns p_1 but still cannot deduce anything about acc_1. In particular, the worlds $\{r_2, p_1, acc_1\}$ and $\{r_2, p_1\}$ are related by R_C. Hence, C knows that acc_1 is not known and, therefore, the branch leading to rejection is enabled, see the states $\mathscr{P}'_1$ and $\mathscr{P}''_1$. Then, it is the turn of the proposal from S_2 (see state $\mathscr{P}_{12}$), which allows C to know acc_2, because C already knows r_2, $r_2 \wedge p_2 \rightarrow acc_2$ and has just acquired the knowledge of p_2. In practice, the world $\{r_2, p_2, acc_2\}$ is connected by R_C only to states including all the three propositions. Therefore, the proposal is accepted (see the states $\mathscr{P}'_{12}$ and $\mathscr{P}''_{12}$), with subsequent failure (state $\mathscr{P}_C$) or success of the task delivery (state $\mathscr{P}$). Going back to the race between the two proposals (see

state $\mathscr{P}'_0$), let us consider the case in which agent C receives the proposal from S_2 first (see state $\mathscr{P}_2$ of the right-hand branch). For the same motivations surveyed above, the proposal is accepted (see states $\mathscr{P}'_2$ and $\mathscr{P}''_2$). At this stage, the task either is completed successfully (state $\mathscr{P}_{S_1}$), or fails (state $\mathscr{P}_{21}$). In the latter case, the proposal of S_1 is considered, which, however, is rejected for the same motivations illustrated in the other branch, see states $\mathscr{P}'_{21}$, $\mathscr{P}''_{21}$, and $\mathscr{P}_C$. Now, it is worth observing and comparing the three states that we have not expanded yet, i.e., $\mathscr{P}$, $\mathscr{P}_C$, and $\mathscr{P}_{S_1}$. State $\mathscr{P}$ enables the local action *ok* of agent C leading to a state which is like $\mathcal{P}_0$ and, in addition, is such that agent C knows formula ψ_r, which is the result of the task delivery, and the preconditions p_1 and p_2 of the two participants. In state $\mathscr{P}_C$, the two participant agents are in their initial local state, while agent C is stuck in the local process term *Init_ Wait*, since the task has not been obtained and no timeout mechanism is enabled. In state $\mathscr{P}_{S_1}$, agent C is ready for a new call after the execution of the local action *ok*, S_2 is in its initial local state, while agent S_1 is still waiting to send its proposal after the call, since, again, no timeout mechanism is enabled. These last two situations emphasise the need for enabling the timeout mechanism—modeled through the action *timeout* guarded by the condition ψ_t—in order to avoid starvation of the agents.

Finally, based on the described model, we can observe that:

- The EHML formula $\langle\tau\rangle\langle\tau\rangle\langle \mathit{verify}\rangle\langle\tau\rangle\lfloor \mathbf{K}_C \mathit{acc}_2\rfloor$ is satisfied in the initial state of the model, while the EHML formula $\langle\tau\rangle\langle\tau\rangle\langle \mathit{verify}\rangle\langle\tau\rangle\lfloor \mathbf{K}_C \mathit{acc}_1\rfloor$ is false in the model, as emphasised by the alternative paths of Figure 6.2.

- The EHML formula $\lfloor\neg\mathbf{K}_C\psi_r\rfloor \rightarrow \neg\langle \mathit{ok}\rangle\top$ is true in the model, since the action *ok* is executed by agent C only when ψ has been notified by the participant delivering the task.

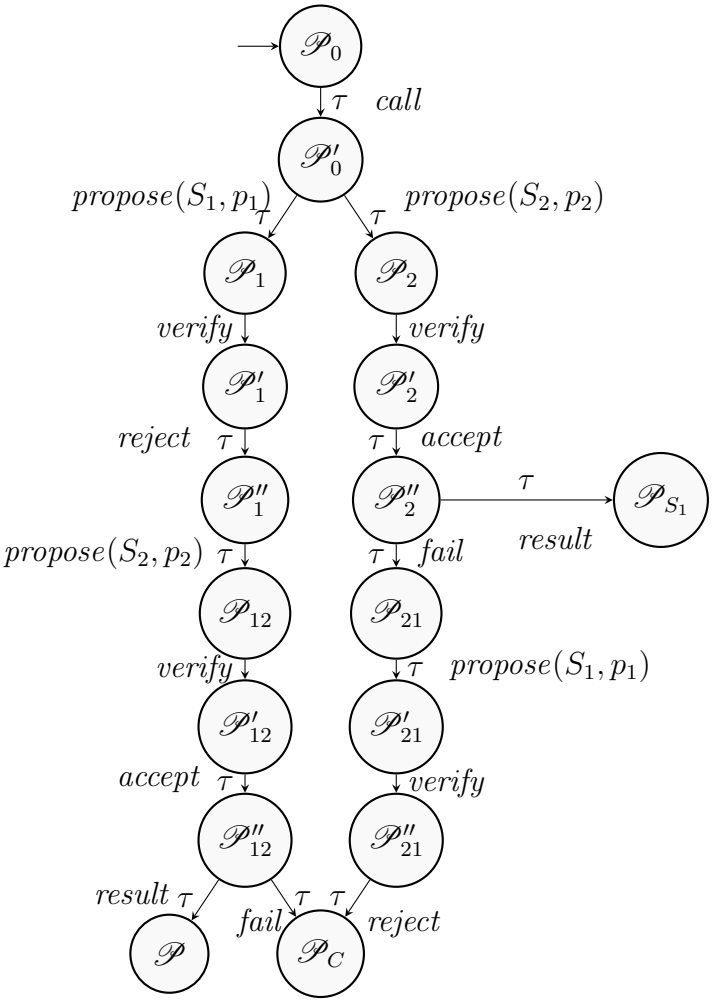


Figure 6.2: Portion of the KLTS underlying the pool of agents of the FIPA CNIP.

Conclusion

We now discuss possible directions for future research arising from the results presented in Part I and Part II.

Part I

In order to develop more effective tools for reasoning about the structure of f-properties, we plan to extend our research in two parallel directions which, we hope, will eventually converge.

1. We aim to provide a complete axiomatisation of $\wp(\Sigma^*)$ as a bounded ℓ-monoid, together with a structural proof-theoretic counterpart.

2. We intend to study expansions of closure ℓ-monoids obtained by adding operations such as the divisions of residuated lattices or the Kleene star. Besides offering a coherent further development of our algebraic theory of f-properties, we believe that the proof-theoretic counterparts of these enriched structures are intrinsically interesting in their own right. In particular, it would be worthwhile, both for LMC and its extensions, to address the problem of obtaining cut-free completeness results, as achieved in [103] for the Distributive Full Lambek Calculus, and in [70] for Infinitary Action Logic.

In pursuing these objectives, if necessary, we will explore alternative proof-theoretic frameworks such as nested sequents [114, 152] or deep inference [29].

LMC is intended to complement model checking by providing a tool for the compositional analysis of the f-properties to be verified. It is therefore necessary to investigate how model checking algorithms can be combined with the use of LMC and to assess the impact of this integration on their efficiency. Moreover, we believe it would be fruitful

to relate our calculus to other proof-theoretic approaches to the analysis of concurrent systems, in particular those developed in [1, 2, 83, 84].

From a strictly algebraic perspective, we first intend to further develop the study of the variety of closure ℓ-monoids by establishing representation and structure theorems, and by investigating possible connections with the theory of enriched Płonka sums [69]. Moreover, as the reader will have noticed, the term "closure ℓ-monoid" is clearly inspired by McKinsey and Tarski's *closure algebras* [117, 118], which provide the algebraic semantics for S4. In addition, our terminological choice reflects the fact that the prefix-closure operation plays a central and distinctive role in our theory, and "closure ℓ-monoid" is meant to emphasise this. However, one might argue that, in view of the residuation between $\Diamond$ and $\boxplus$, a more appropriate name for our structures would be "monadic ℓ-monoids", by analogy with Halmos' *monadic Boolean algebras* [86], the algebraic semantics of S5. Indeed, letting ν be the similarity type of closure ℓ-monoids, our structures in fact seem to be m-ν-lattices in the sense of [39]; thus it would be interesting to investigate how $\mathfrak{LMC}$ relates to the general theory of one-variable fragments of first-order logics.

Part II

As a first research direction, building on the considerations in Remark 5.3.9, we intend to provide a model checking algorithm for EHML, establish complexity results, and develop a corresponding software tool, possibly leveraging techniques from existing model checkers such as TAPAs [33], CADP [74], DEMO-S5 [166] or mdk-verifier [111]. A further line of research is to situate EHML more precisely within the landscape of combining logics, in particular by relating it to approaches grounded in Goguen and Burstall's *theory of institutions* [78] (see, for instance, [132]). Moreover, given the high generality of our framework, it would also be of interest to investigate the mapping of DEL/ETL models to ours (e.g., along the same line of [162]) as well as possible connections with coalgebraic modal logics [108], thereby enabling a sharper and more principled comparison with existing approaches. Finally, we aim to assess the applicability of variants of EHML to the modelling of hyperproperties [41] in distributed and reactive systems, beginning with concrete case studies (e.g., [67]).

Starting from the process-algebraic framework we developed, several extensions can be envisioned. For instance, the semantics of our communication mechanisms assumes

that only known truth can be transferred. Hence, we do not currently manage (possibly false) beliefs and the communication of information that is inconsistent with an agent's knowledge or belief. This would require the introduction of belief modalities and the treatment of contradictions resulting from the communication between agents. Interestingly, this would open to the modelling of malicious agents sharing false information and, therefore, a theory of fake news [4, 131], trust, and reputation [6, 7, 9, 159]. Along the same lines, further modalities could be added to the epistemic component of our model.

Dealing with inconsistencies is also a problem to face in the present model without bringing up the notion of belief. In particular, an unsuccessful formula is a formula that might become false as soon as it is communicated, like, e.g., in the case of $p \wedge \neg \mathbf{K}_j p$ whenever agent i communicates it to agent j [164]. Several studies investigate the syntactic form of potential unsuccessful formulas, in particular in the setting of public announcements for multi-agent systems [153]. Obviously, even in our framework, such forms can be recognised.

Bibliography

[1] Matteo Acclavio and Giulia Manara. Proofs as Execution Trees for the π-Calculus. arXiv:2411.08847v2, 2025.

[2] Matteo Acclavio, Giulia Manara, and Fabrizio Montesi. Formulas as Processes, Deadlock-Freedom as Choreographies. In Viktor Vafeiadis, editor, *Programming Languages and Systems. 34th European Symposium on Programming, ESOP 2025*, volume 15694 of *Lecture Notes in Computer Science*, pages 23–55, Cham, 2025. Springer. doi:10.1007/978-3-031-91118-7_2.

[3] Alessandro Aldini. Design and Verification of Trusted Collective Adaptive Systems. *ACM Transactions on Modeling and Computer Simulation*, 28(2), 2018. doi:10.1145/3155337.

[4] Alessandro Aldini. On the modeling and verification of the spread of fake news, algebraically. *Journal of Logic and Computation*, 32(6):1272–1291, 2022. doi:10.1093/logcom/exac015.

[5] Alessandro Aldini. A process algebraic framework for multi-agent dynamic epistemic systems. In Ugo de'Liguoro, Matteo Palazzo, and Luca Roversi, editors, *Proceedings of the 25th Italian Conference on Theoretical Computer Science (ICTCS 2024)*, pages 269–283, 2024. URL: https://ceur-ws.org/Vol-3811/paper010.pdf.

[6] Alessandro Aldini, Gianluca Curzi, Pierluigi Graziani, and Mirko Tagliaferri. Trust evidence logic. In Jiřina Vejnarová and Nic Wilson, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty. 16th European Conference, ECSQARU 2021, Prague, Czech Republic, September 21–24, 2021, Pro-*

ceedings, volume 12897 of *Lecture Notes in Artificial Intelligence*, pages 575–589, Cham, 2021. Springer. doi:10.1007/978-3-030-86772-0_41.

[7] Alessandro Aldini, Gianluca Curzi, Pierluigi Graziani, and Mirko Tagliaferri. A probabilistic modal logic for context-aware trust based on evidence. *International Journal of Approximate Reasoning*, 169:Article 109167, 2024. doi:10.1016/j.ijar.2024.109167.

[8] Alessandro Aldini and Ludovico Fusco. Bridging concurrency theory and epistemic models: a formal framework for dynamic multi-agent systems. *Journal of Logic, Language and Information*, 2026. doi:10.1007/s10849-026-09461-3.

[9] Alessandro Aldini and Mirko Tagliaferri. Logics to reason formally about trust computation and manipulation. In Andrea Saracino and Paolo Mori, editors, *Emerging Technologies for Authorization and Authentication. Second International Workshop, ETAA 2019, Luxembourg City, Luxembourg, September 27, 2019*, volume 11967 of *Lecture Notes in Computer Science*, pages 1–15, Cham, 2020. Springer. doi:10.1007/978-3-030-39749-4_1.

[10] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985. doi:10.1016/0020-0190(85)90056-0.

[11] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002. doi:10.1145/585265.585270.

[12] Marlow Anderson and Todd Feil. *Lattice-Ordered Groups: An Introduction*, volume 3 of *Reidel Texts in the Mathematical Sciences*. Reidel, Dordrecht, 1988.

[13] Carlos Areces and Raffaella Bernardi. Analyzing the Core of Categorial Grammar. *Journal of Logic, Language and Information*, 13(2):121–137, 2004. doi:10.1023/B:JLLI.0000024730.34743.fa.

[14] Carlos Areces, Raffaella Bernardi, and Michael Moortgat. Galois Connections in Categorial Type Logic. *Electronic Notes in Theoretical Computer Science*, 53:3–20, 2003. doi:10.1016/S1571-0661(05)82570-8.

[15] Jos C.M. Baeten and Davide Sangiorgi. Concurrency Theory: A Historical Perspective on Coinduction and Process Calculi. In Dov M. Gabbay, Jörg H. Siekmann, and John Woods, editors, *Handbook of the History of Logic. Volume 9: Computational Logic*, pages 399–442. North-Holland, Amsterdam, 2014.

[16] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, 2008.

[17] Musard Balliu, Mads Dam, and Gurvan Le Guernic. Epistemic temporal logic for information flow security. In *PLAS '11: Proceedings of the 6th ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, New York, 2011. ACM. doi:10.1145/2166956.2166962.

[18] Alexandru Baltag and Lawrence S. Moss. Logics for Epistemic Programs. *Synthese*, 139:165–224, 2004. doi:10.1023/B:SYNT.0000024912.56773.5e.

[19] Alexandru Baltag, Lawrence S. Moss, and Sławomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In Horacio Arló-Costa, Vincent F. Hendricks, and Johan van Benthem, editors, *Readings in Formal Epistemology: Sourcebook*, pages 773–812. Springer, , 2016. doi:10.1007/978-3-319-20451-2_38.

[20] Kai Bavendiek and Sibylle Schupp. A process calculus for privacy-preserving protocols in location-based service systems. *Journal of Logical and Algebraic Methods in Programming*, 125, 2022. doi:10.1016/j.jlamp.2021.100735.

[21] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Alternating-time Temporal Logic on Finite Traces. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 77–83, 2018. doi:10.24963/ijcai.2018/11.

[22] Nuel Belnap. Display Logic. *Journal of Philosophical Logic*, 11(4):375–417, 1982. doi:10.1007/BF00284976.

[23] Clifford Bergman. *Universal Algebra: Fundamentals and Selected Topics*. CRC Press, Boca Raton, 2012.

[24] Garrett Birkhoff. On the Structure of Abstract Algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31(4):433–454, 1935. doi:10.1017/S0305004100013463.

[25] Garrett Birkhoff. *Lattice Theory*, volume XXV of *AMS Colloquium Publications*. AMS, Providence, 1948.

[26] Patrick Blackburn, Maarten De Rijke, and Yde Venema. *Modal logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2002.

[27] Thomas Scott Blyth and Melvin Fiva Janowitz. *Residuation Theory*. Pergamon Press, New York, 1972.

[28] Thomas Bolander. A gentle introduction to epistemic planning: The DEL approach. *Electronic Proceedings in Theoretical Computer Science*, 243:1–22, 2017. doi:10.4204/EPTCS.243.1.

[29] Kai Brünnler. Deep sequent systems for modal logic. *Archive for Mathematical Logic*, 48:551–577, 2009. doi:10.1007/s00153-009-0137-3.

[30] Stanley N. Burris and Hanamantagouda P. Sankappanavar. A Course in Universal Algebra, Millennium Edition, 2012. Available at https://www.math.uwaterloo.ca/~snburris/htdocs/UALG/univ-algebra2012.pdf.

[31] Manuela Busaniche, Nikolaos Galatos, and Miguel Andrés Marcos. Twist Structures and Nelson Conuclei. *Studia Logica*, 110:949–987, 2022. doi:10.1007/s11225-022-09988-z.

[32] Carlos Caleiro, Cristina Sernadas, and Amílcar Sernada. Parameterisation of Logics. In José Luiz Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques. 13th International Workshop, WADT'98, Lisbon, Portugal, April 2-4, 1998, Selected Papers*, pages 48–63, Berlin-Heidelberg, 1999. Springer-Verlag. doi:10.1007/3-540-48483-3_4.

[33] Francesco Calzolai, Rocco De Nicola, Michele Loreti, and Francesco Tiezzi. TAPAs: A Tool for the Analysis of Process Algebras. In Kurt Jensen, Wil M.P. Aalst, and Jonathan Billington, editors, *Transactions on Petri Nets*

and Other Models of Concurrency I, volume 5100 of *Lecture Notes in Computer Science*, pages 54–70. Springer, Berlin-Heidelberg, 2008. doi:10.1007/978-3-540-89287-8_4.

[34] Walter Carnielli and Marcelo Esteban Coniglio. Combining Logics. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2025. URL: https://plato.stanford.edu/archives/win2025/entries/logic-combining/.

[35] Rohit Chadha, Stéphanie Delaune, and Steve Kremer. Epistemic Logic for the Applied Pi Calculus. In David Lee, Antónia Lopes, and Arnd Poetzsch-Heffter, editors, *Formal Techniques for Distributed Systems. Joint 11th IFIP WG 6.1 International Conference FMOODS 2009 and 29th IFIP WG 6.1 International Conference FORTE 2009, Lisboa, Portugal, June 9-12, 2009, Proceedings*, pages 182–197, Berlin-Heidelberg, 2009. Springer. doi:10.1007/978-3-642-02138-1_12.

[36] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010. Special Issue: 18th International Conference on Concurrency Theory (CONCUR 2007). URL: https://www.sciencedirect.com/science/article/pii/S0890540110000192, doi:10.1016/j.ic.2009.07.004.

[37] Jinsheng Chen, Giuseppe Greco, Alessandra Palmigiano, and Apostolos Tzimoulis. Syntactic Completeness of Proper Display Calculi. *ACM Transactions on Computational Logic*, 23(4):Article No.: 23, 2022. doi:10.1145/3529255.

[38] Agata Ciabattoni, Revantha Ramanayake, and Heinrich Wansing. Hypersequent and Display Calculi — a Unified Perspective. *Studia Logica*, 102(6):1245–1294, 2014. doi:10.1007/s11225-014-9566-z.

[39] Petr Cintula, George Metcalfe, and Naomi Tokuda. One-Variable Fragments of First-Order Logics. *The Bulletin of Symbolic Logic*, 30(2):253–278, 2024. doi:10.1017/bsl.2024.22.

[40] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Dexter Kozen, editor, *Logics*

of Programs, pages 52–71, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. doi:10.1007/BFb0025774.

[41] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.

[42] Alfred Hoblitzelle Clifford and Gordon Bamford Preston. *The Algebraic Theory of Semigroups*, volume 7 (Part I) of *Mathematical Surveys and Monographs*. AMS, Providence, 1961.

[43] Alfred Hoblitzelle Clifford and Gordon Bamford Preston. *The Algebraic Theory of Semigroups. Volume II*, volume 7 (Part II) of *Mathematical Surveys and Monographs*. AMS, Providence, 1967.

[44] Norine Coenen, Bernd Finkbeiner, Christopher Hahn, and Jana Hofmann. The Hierarchy of Hyperlogics. In *Proc. of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '19), Article No.: 39*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785713.

[45] Almudena Colacito, Nikolaos Galatos, George Metcalfe, and Simon Santschi. From distributive ℓ-monoids to ℓ-groups, and back again. *Journal of Algebra*, 601:129–148, 2022. doi:10.1016/j.jalgebra.2022.02.012.

[46] George F. Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design (4th edition)*. Addison-Wesley, Boston, 2005.

[47] Patrick Cousot. Methods and Logics for Proving Programs. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics*, pages 842–993. Elsevier/MIT Press, Amsterdam/Cambridge, MA, 1990.

[48] Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 2002.

[49] Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-Past Linear Temporal and Dynamic Logic on Finite Traces. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI '20*, pages 4959–4965, 2021. URL: https://www.ijcai.org/proceedings/2020/0690.pdf.

[50] Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In Francesca Rossi, editor, *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13*, pages 854–860, 2013. URL: https://www.ijcai.org/Proceedings/13/Papers/132.pdf.

[51] Rocco De Nicola and Frits Vaandrager. Action versus state based logics for transition systems. In Irène Guessarian, editor, *Semantics of Systems of Concurrent Processes*, pages 407–419, , 1990. Springer. doi:10.1007/3-540-53479-2_17.

[52] Francien Dechesne, Mohammad Reza Mousavi, and Simona Orzan. Operational and Epistemic Approaches to Protocol Analysis: Bridging the Gap. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning. 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 226–241, Berlin-Heidelberg, 2007. Springer. doi:10.1007/978-3-540-75560-9_18.

[53] Francien Dechesne and Yanjing Wang. To know or not to know: epistemic approaches to security protocol verification. *Synthese*, 177:51–76, 2010. doi:10.1007/s11229-010-9765-8.

[54] Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal Logics in Computer Science*, volume 58 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016.

[55] Edsger W. Dijkstra. Solution of a Problem in Concurrent Programming Control. *Commun. ACM*, 8(9):569, 1965. doi:10.1145/365559.365617.

[56] Jon Michael Dunn. A 'Gentzen' System for Positive Relevant Implication (Abstract). *Journal of Symbolic Logic*, 38(2):356–357, 1973.

[57] Jon Michael Dunn. Gaggle theory: An abstraction of Galois connections and residuation, with applications to negation, implication, and various logical operators. In Jan van Eijck, editor, *Logics in AI. Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA 1990)*, volume 478 of *Lecture Notes in Computer Science*, pages 31–51, 1991. doi:10.1007/BFb0018431.

[58] Jon Michael Dunn. Partial-Gaggles Applied to Logics with Restricted Structural Rules. In Kosta Došen and Peter Schroeder-Heister, editors, *Substructural Logics*, pages 63–108. Clarendon, Oxford, 1993.

[59] Jon Michael Dunn. Gaggle Theory Applied to Intuitionistic, Modal and Relevance logic. In Ingolf Max and Werner Stelzner, editors, *Logik und Mathematik. Frege-Kolloquium Jena*, volume 5 of *Perspektiven der Analytischen Philosophie*, pages 335–368. Berlin, De Gruyter, 1995.

[60] Samuel Eilenberg. *Automata, Languages, and Machines. Volume A*. Academic Press, New York-London, 1974.

[61] E. Allen Emerson and Joseph Y. Halpern. “Sometimes" and “not never" revisited: on branching versus linear time (preliminary report). In *Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL ’83, page 127–140, New York, NY, USA, 1983. ACM. doi:10.1145/567067.567081.

[62] Ernest Allen Emerson. Temporal and Modal Logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics*, pages 995–1072. Elsevier/MIT Press, Amsterdam/Cambridge, MA, 1990.

[63] Marcelo Finger and Dov M. Gabbay. Adding a Temporal Dimension to a Logic System. *Journal of Logic, Language, and Information*, 1(3):203–233, 1992. doi:10.1007/BF00156915.

[64] Wan Fokkink. *Introduction to Process Algebra*. Springer, Berlin-Heidelberg, 2007.

[65] László Fuchs. *Partially Ordered Algebraic Systems*. Pergamon Press, Oxford, 1963.

[66] László Fuchs. Riesz groups. *Annali della Scuola Normale Superiore di Pisa - Scienze Fisiche e Matematiche, Serie 3*, 19(1):1–34, 1965. URL: https://eudml.org/doc/83340.

[67] Ludovico Fusco and Alessandro Aldini. Hyperproperties for Safe and Secure RFID Systems. In Gianni D’Angelo, Flaminia Luccio, and Francesco Palmieri,

editors, *Proceedings of the 8th Italian Conference on Cyber Security (ITASEC 2024) Salerno, Italy, April 8-12, 2024*, page Paper 20, 2024. URL: https://ceur-ws.org/Vol-3731/paper20.pdf.

[68] Ludovico Fusco and Alessandro Aldini. A Cut-Free Sequent Calculus for the Analysis of Finite-Trace Properties in Concurrent Systems. arXiv:2512.03164v2, 2025.

[69] Ludovico Fusco and Francesco Paoli. Enriched Płonka sums. *Algebra Universalis*, 87:Article No. 3, 2026. doi:10.1007/s00012-025-00910-x.

[70] Wesley Fussner, Simon Santschi, and Borja Sierra Miranda. Algebraic Proof Theory for Infinitary Action Logic. arXiv:2501.18231, 2025.

[71] Dov Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, pages 163—-173, New York, 1980. ACM. doi:10.1145/567446.567462.

[72] Nikolaos Galatos and Peter Jipsen. Distributive residuated frames and generalized bunched implication algebras. *Algebra Universalis*, 78:303–336, 2017. doi:10.1007/s00012-017-0456-x.

[73] Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski, and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*, volume 151 of *Studies in Logic and The Foundations of Mathematics*. Elsevier, Amsterdam, 2007.

[74] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Serwe Wendelin. CADP 2011: a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, 15(2):89–107, 2013. doi:10.1007/s10009-012-0244-z.

[75] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1935. doi:10.1007/BF01201353.

[76] Gerhard Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, 39:405–431, 1935. doi:10.1007/BF01201363.

[77] Patrick Girard, Jeremy Seligman, and Fenrong Liu. General dynamic dynamic logic. In Thomas Bolander, Torben Braüner, Silvio Ghilardi, and Lawrence Moss, editors, *9th Conf. on Advances in Modal Logic*, volume 9 of *Advances in Modal Logic*, pages 239–260, , 2012. College Publications.

[78] Joseph A. Goguen and Rod M. Burstall. Institutions: abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992. doi:10.1145/147508.147524.

[79] Rajeev Goré. Gaggles, Gentzen and Galois: how to display your favourite substructural logic. *Logic Journal of the IGPL*, 6(5):669–694, 1998. doi:10.1093/jigpal/6.5.669.

[80] Roberto Gorrieri and Cristian Versari. *Introduction to Concurrency Theory - Transition Systems and CCS*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Cham, 2015.

[81] George Grätzer. *Lattice Theory: Foundation*. Birkhäuser, Basel, 2011.

[82] Giuseppe Greco, Peter Jipsen, Fei Liang, Alessandra Palmigiano, and Apostolos Tzimoulis. Algebraic Proof Theory for LE-logics. *ACM Transactions on Computational Logic*, 25(1):Article No.: 6, 2024. doi:10.1145/3632526.

[83] Alessio Guglielmi. Concurrency and Plan Generation in a Logic Programming Language with a Sequential Operator. In Pascal Van Hentenryck, editor, *Proceedings of the Eleventh International Conference on Logic Programming*, pages 240–254, Cambridge, MA, 1994. MIT Press. doi:10.7551/mitpress/4316.003.0030.

[84] Alessio Guglielmi. Sequentiality by Linear Implication and Universal Quantification. In Jörg Desel, editor, *Structures in Concurrency Theory. Proceedings of the International Workshop on Structures in Concurrency Theory (STRICT), Berlin, 11–13 May 1995*, pages 160–174, London, 1995. Springer. doi:10.1007/978-1-4471-3078-9_11.

[85] Michell Guzman, Stefan Haar, Salim Perchy, Camilo Rueda, and Frank D. Valencia. Belief, knowledge, lies and other utterances in an algebra for space and extrusion. *Journal of Logical and Algebraic Methods in Programming*, 86(1):107–133, 2017. doi:10.1016/j.jlamp.2016.09.001.

[86] Paul R. Halmos. Algebraic logic, I. Monadic boolean algebras . *Compositio Mathematica*, 12(1):217–249, 1954–1956. URL: https://eudml.org/doc/88816.

[87] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992. doi:10.1016/0004-3702(92)90049-4.

[88] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[89] M. Hennessy. A proof system for communicating processes with value-passing. *Formal Aspects of Computing*, 3:346–366, 1991. doi:10.1007/BF01642508.

[90] Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In Jaco W. de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming. Seventh Colloquium, Noordwijkerhout, The Netherlands, July 14-18, 1980. Proceedings*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309, Berlin-Heidelberg, 1980. Springer. doi:10.1007/3-540-10003-2_79.

[91] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

[92] Charles Anthony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576—-580, October 1969. doi:10.1145/363235.363259.

[93] Charles Anthony Richard Hoare. *Communicating Sequential Processes*. Prentice Hall International, Hoboken, 1985.

[94] Shuqin Huang, Yongzhi Cao, Hanpin Wang, and Wanling Qu. Value-passing CCS with noisy channels. *Theoretical Computer Science*, 433:43–59, 2012. doi:10.1016/j.tcs.2012.03.002.

[95] Bjarni Jónsson and Constantine Tsinakis. Relation algebras as residuated Boolean algebras. *Algebra Universalis*, 30:469–478, 1993. doi:10.1007/BF01195378.

[96] Daniel M. Kan. Adjoint Functors. *Transactions of the American Mathematical Society*, 87(2):294–329, 1958. doi:10.2307/1993102.

[97] Ryo Kashima. Cut-Free Sequent Calculi for Some Tense Logics. *Studia Logica*, 53(1):119–135, 1994. doi:10.1007/BF01053026.

[98] Robert M. Keller. Formal Verification of Parallel Programs. *Communications of the ACM*, 19(7):371–384, 1976. doi:10.1145/360248.360251.

[99] Eiji Kiriyama and Hiroakira Ono. The Contraction Rule and Decision Problems for Logics without Structural Rules. *Studia Logica*, 50(2):299–319, 1991. doi:10.1007/BF00370189.

[100] Sophia Knight, Radu Mardare, and Prakash Panangaden. Combining Epistemic Logic and Hennessy-Milner Logic. In Robert L. Constable and Alexandra Silva, editors, *Logic and Program Semantics. Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday*, volume 7230 of *Lecture Notes in Computer Science*, pages 219–243, Berlin-Heidelberg, 2012. Springer. doi:10.1007/978-3-642-29485-3_14.

[101] Sophia Knight, Catuscia Palamidessi, Prakash Panangaden, and Frank D. Valencia. Spatial and Epistemic Modalities in Constraint-Based Process Calculi. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR 2012- Concurrency Theory. 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 317–332, Berlin-Heidelberg, 2012. Springer. doi:10.1007/978-3-642-32940-1_23.

[102] Barteld Kooi and Bryan Renne. Generalized arrow update logic. In Krzysztof R. Apt, editor, *TARK XIII: Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 205–211, New York, 2011. ACM. doi:10.1145/2000378.2000403.

[103] Michał Kozak. Distributive Full Lambek Calculus Has the Finite Model Property. *Studia Logica*, 91(2):201–216, 2009. doi:10.1007/s11225-009-9172-7.

[104] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

[105] Dexter Kozen. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Information and Computation*, 110(2):366–390, 1994. doi:10.1006/inco.1994.1037.

[106] Marcus Kracht. Highway to the Danger Zone. *Journal of Logic and Computation*, 5(1):93–109, 1995. doi:10.1093/logcom/5.1.93.

[107] Marcus Kracht. Power and Weakness of the Modal Display Calculus. In Heinrich Wansing, editor, *Proof Theory of Modal Logic*, volume 2 of *Applied Logic Series*, pages 93–121. Springer, Dordrecht, 1996.

[108] Clemens Kupke and Dirk Pattinson. Coalgebraic semantics of modal logics: An overview. *Theoretical Computer Science*, 412(38):5070–5094, 2011. doi:10.1016/j.tcs.2011.04.023.

[109] Natasha Kurtonina. *Frames and Labels. A Modal Analysis of Categorial Inference*. PhD thesis, Utrecht University and ILLC Amsterdam, 1995. URL: https://eprints.illc.uva.nl/id/eprint/1981/.

[110] Natasha Kurtonina and Michael Moortgat. Structural Control. In Patrick Blackburn and Maarten de Rijke, editors, *Specifying Syntactic Structures*. CSLI Publications, Stanford, 1997.

[111] Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, and Valentin Montmirail. On Checking Kripke Models for Modal Logic K. In Pascal Fontaine, Stefan Schulz, and Josef Urban, editors, *PAAR 2016. Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning co-located with International Joint Conference on Automated Reasoning (IJCAR 2016)*, pages 69–81, 2016. URL: https://ceur-ws.org/Vol-1635/paper-07.pdf.

[112] Leslie Lamport. Proving the Correctness of Multiprocess Programs. *IEEE Transactions on Software Engineering*, SE-3(2):125–143, 1977. doi:10.1109/TSE.1977.229904.

[113] Leslie Lamport. Turing Lecture – The Computer Science of Concurrency: The Early Years. *Communications of the ACM*, 58(6):71–76, May 2015. doi:10.1145/2771951.

[114] Björn Lellmann and Francesca Poggiolesi. Nested Sequents or Tree-Hypersequents—A Survey. In Yale Weiss and Romina Birman, editors, *Saul Kripke on Modal Logic*, volume 30 of *Outstanding Contributions to Logic*, pages 243–301. Springer, Cham, 2024.

[115] Zohar Manna and Amir Pnueli. A Hierarchy of Temporal Properties. In *PODC '90: Proceedings of the 9th Annual ACM Symposium in Principles of Distibuted Computing*, pages 377–410, New York, 1990. ACM. invited paper. doi:10.1145/93385.93442.

[116] Ralph N. McKenzie, George F. McNulty, and Walter F. Taylor. *Algebras, Lattices, Varieties: Volume I.* AMS Chelsea Publishing, 1987.

[117] John Charles Chenoweth McKinsey and Alfred Tarski. The Algebra of Topology. *Annals of Mathematics*, 45(1):141–191, 1944. doi:10.2307/1969080.

[118] John Charles Chenoweth McKinsey and Alfred Tarski. On Closed Elements in Closure Algebras. *Annals of Mathematics*, 47(1):122–162, 1946. doi:10.2307/1969038.

[119] Matias Menni and Clara Smith. Modes of Adjointness. *Journal of Philosophical Logic*, 43:365–391, 2014. doi:10.1007/s10992-012-9266-y.

[120] George Metcalfe, Nicola Olivetti, and Dov M. Gabbay. *Proof Theory for Fuzzy Logics*, volume 36 of *Applied Logic Series*. Springer Science+Business Media B.V., Dordrecht, 2009.

[121] George Metcalfe, Francesco Paoli, and Constantine Tsinakis. *Residuated Structures in Algebra and Logic*, volume 277 of *Mathematical Surveys and Monographs*. AMS, Providence, 2023.

[122] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, Berlin-Heidelberg, 1980.

[123] Robin Milner. Operational and Algebraic Semantics of Concurrent Processes. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics*, pages 1202–1242. Elsevier/MIT Press, Amsterdam/Cambridge, MA, 1990.

[124] Robin Milner. Elements of interaction: Turing award lecture. *Communications of the ACM*, 36:78–89, 1993. doi:10.1145/151233.151240.

[125] Kiraku Minami. Trace Equivalence and Epistemic Logic to Express Security Properties. In Alexey Gotsman and Ana Sokolova, editors, *Formal Techniques for Distributed Objects, Components, and Systems. 40th IFIP WG 6.1 International Conference, FORTE 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15–19, 2020, Proceedings*, pages 115–132, Cham, 2020. Springer. doi:10.1007/978-3-030-50086-3_7.

[126] Grigori Minc. Cut-elimination theorem for relevant logics. *Journal of Mathematical Sciences*, 6:422–428, 1976. doi:10.1007/BF01084083.

[127] Michael Moortgat. Multimodal Linguistic Inference. *Journal of Logic, Language and Information*, 5(3-4):349–385, 1996. doi:10.1007/BF00159344.

[128] Michael Moortgat. Categorial Type Logics. In Johan Van Benthem and Alice Ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. Elsevier, Amsterdam, 1997.

[129] Richard Moot and Christian Retoré. *The Logic of Categorial Grammars. A deductive account of natural language syntax and semantics*, volume 6850 of *Lecture Notes in Computer Science*. Springer, Berlin-Heidelberg, 2012.

[130] Peter D. Mosses. Denotational Semantics. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics*, pages 576–631. Elsevier/MIT Press, Amsterdam/Cambridge, MA, 1990.

[131] Mohammad Reza Mousavi and Mahsa Varshosaz. Telling Lies in Process Algebra. In *2018 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 116–123. IEEE, 2018. doi:10.1109/TASE.2018.00023.

[132] Renato Neves, Alexandre Madeira, Manuel A. Martins, and Luis S. Barbosa. Proof theory for hybrid(ised) logics. *Science of Computer Programming*, 126:73–93, 2016. doi:10.1016/j.scico.2016.03.001.

[133] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part i. *Theoretical Computer Science*, 13(1):85–108, 1981. Special Issue Semantics of Concurrent Computation. doi:10.1016/0304-3975(81)90112-2.

[134] Joël Ouaknine and James Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007. doi:10.2168/LMCS-3(1:8)2007.

[135] Francesco Paoli. *Substructural Logics: A Primer*, volume 13 of *Trends in Logic*. Springer Science+Business Media, Dordrecht, 2002.

[136] Michele Pasqua and Isabella Mastroeni. On Topologies for (Hyper)Properties. In Dario Della Monica, Aniello Murano, Sasha Rubin, and Luigi Sauro, editors, *Joint Proceedings of the 18th Italian Conference on Theoretical Computer Science (ICTCS 2017) and the 32nd Italian Conference on Computational Logic (CILC 2017) co-located with the 2017 IEEE International Workshop on Measurements and Networking (2017 IEEE M&N)*, pages 150–161, 2017. URL: https://ceur-ws.org/Vol-1949/ICTCSpaper13.pdf.

[137] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Fakultät für Mathematik und Physik, Technischen Hochschule Darmstadt, 1962. URL: https://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/pdf/diss_petri_d.pdf.

[138] Paolo Petta, Andrea Omicini, Terry Payne, and Peter McBurney. Introduction to the special issue: The AgentLink III technical forums. *ACM Transactions on Autonomous and Adaptive Systems*, 2(4):Article 12, 2007. doi:10.1145/1293731.1293732.

[139] Amir Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.

[140] Stefan Poslad. Specifying protocols for multi-agent systems interaction. *ACM Transactions on Autonomous and Adaptive Systems*, 2(4):Article 15, 2007. doi:10.1145/1293731.1293735.

[141] Stefan Poslad and Patricia Charlton. Standardizing Agent Interoperability: The FIPA Approach. In *Multi-Agent Systems and Applications. 9th ECCAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, July 2-13, 2001. Selected Tutorial Papers*, volume 2086 of *Lecture Notes in Computer Science*, pages 98–117, Berlin-Heidelberg, 2001. Springer. doi:10.1007/3-540-47745-4_5.

[142] Vaughan Pratt. Action Logic and Pure Induction. In Jan van Eijck, editor, *Logics in AI. Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA 1990)*, volume 478 of *Lecture Notes in Computer Science*, pages 97–120, 1991. doi:10.1007/BFb0018436.

[143] Vaughan R. Pratt. Semantical Considerations On Floyd-Hoare Logic. In *17th Annual Symposium on Foundations of Computer Science (SFCS 1976)*, pages 109–121, 1976. doi:10.1109/SFCS.1976.27.

[144] Arthur Norman Prior. *Time and Modality*. Clarendon Press, Oxford, 1957.

[145] Bryan Renne, Joshua Sack, and Audrey Yap. Logics of temporal-epistemic actions. *Synthese*, 193(3):813–849, 2016. doi:10.1007/s11229-015-0773-6.

[146] Greg Restall. A Useful Substructural Logic. *Bulletin of the IGPL*, 2(2):137–148, 1994. doi:10.1093/jigpal/2.2.137.

[147] Greg Restall. Displaying and Deciding Substructural Logics 1: Logics with Contraposition. *Journal of Philosophical Logic*, 27(2):179–216, 1998. doi:10.1023/A:1017998605966.

[148] Frigyes Riesz. Sur la décomposition des opérations fonctionnelles linéaires. In *Atti del Congresso Internazionale dei Matematici. Bologna 3–10 Settembre 1928. Tomo III*, pages 143–148, Bologna, 1929. Zanichelli. URL: https://www.mathunion.org/fileadmin/ICM/Proceedings/ICM1928.3/ICM1928.3.ocr.pdf.

[149] Grigore Roşu. On Safety Properties and Their Monitoring. *Scientific Annals of Computer Science*, 22(2):327–365, 2012. doi:10.7561/SACS.2012.2.327.

[150] Grigore Roşu. Finite-trace linear temporal logic: coinductive completeness. *Formal Methods in System Design*, 53:138–163, 2018. doi:10.1007/s10703-018-0321-3.

[151] Vladimir Rybakov. Linear Temporal Logic $\mathscr{LTL}_K$ extended by Multi-Agent Logic K_n with Interacting Agents. *Journal of Logic and Computation*, 19(6):989–1017, 2009. doi:10.1093/logcom/exp027.

[152] Mehrnoosh Sadrzadeh and Roy Dyckhoff. Positive logic with adjoint modalities: Proof theory, semantics and reasoning about information. *Electronic Notes in Theoretical Computer Science*, 249:451–470, 2009. doi:10.1016/j.entcs.2009.07.102.

[153] Sanchit Saraf and Sumit Sourabh. Characterizing successful formulas: the multi-agent case. *CoRR*, abs/1209.0935, 2012.

[154] Vladimiro Sassone, Mogens Nielsen, and Glynn Winskel. Models for concurrency: towards a classification. *Theoretical Computer Science*, 170(1):297–348, 1996. doi:10.1016/S0304-3975(96)80710-9.

[155] Klaus Schneider. *Verification of Reactive Systems. Formal Methods and Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin-Heidelberg, 2004.

[156] Philippe Schnoebelen. The Complexity of Temporal Logic Model Checking. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logic, Volume 4: Papers From the Fourth AiML Conference, Held in Toulouse, October 2002*, pages 393–436, Rickmansworth, 2003. College Publications. URL: https://lsv.ens-paris-saclay.fr/Publis/PAPERS/PDF/Sch-aiml02.pdf.

[157] Michael B. Smyth. Topology. In Samson Abramski, Dov M. Gabbay, and Thomas Stephen Edward Maibaum, editors, *Handbook of Logic in Computer Science. Volume 1. Background: Mathematical Structures*, pages 641–761. Oxford University Press, Oxford, 1993.

[158] Bayu Surarso and Hiroakira Ono. Cut Elimination in Noncommutative Substructural Logics. *Reports on Mathematical Logic*, 30:13–29, 1996. URL: https://rml.tcs.uj.edu.pl/rml-30/ono.pdf.

[159] Mirko Tagliaferri and Alessandro Aldini. From belief to trust: A quantitative framework based on modal logic. *Journal of Logic and Computation*, 32(6):1017–1047, 2022. doi:10.1093/logcom/exac016.

[160] Alfred Tarski. A Remark on Functionally Free Algebras. *Annals of Mathematics*, 47(1):163–166, 1946. doi:10.2307/1969039.

[161] Maurice H. ter Beek, Alessandro Fantechi, Stefania Gnesi, and Franco Mazzanti. A state/event-based model-checking approach for the analysis of abstract system properties. *Science of Computer Programming*, 76(2):119–135, 2011. doi:10.1016/j.scico.2010.07.002.

[162] Johan van Benthem, Jelle Gerbrandy, and Eric Pacuit. Merging frameworks for interaction: DEL and ETL. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '07, pages 72–81, New York, 2007. ACM.

[163] Wiebe van der Hoek and Michael Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, AAMAS '02, pages 1167–1174, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/545056.545095.

[164] Hans van Ditmarsch and Barteld Kooi. The Secret of My Success. *Synthese*, 151:201–232, 2006. doi:10.1007/s11229-005-3384-9.

[165] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*, volume 337 of *Syntese Library*. Springer, Dordrecht, 2008. doi:10.1007/978-1-4020-5839-4.

[166] Jan van Eijck. DEMO-S5. https://staff.fnwi.uva.nl/d.j.n.vaneijck2/software/demo_s5/DEMO_S5.pdf.

[167] Sieuwert van Otterloo and Geert Jonker. On epistemic temporal strategic logic. *Electronic Notes in Theoretical Computer Science*, 126:77–92, 2005. 2nd Workshop on Logic and Communication in Multi-Agent Systems (2004). doi:10.1016/j.entcs.2004.11.014.

[168] Heinrich Wansing. Modal Tableaux Based on Residuation . *Journal of Logic and Computation*, 7(6):719–731, 1997. doi:10.1093/logcom/7.6.719.

[169] Heinrich Wansing. *Displaying Modal Logic*, volume 3 of *Trends in Logic*. Springer Science+Business Media, Dordrecht, 1998.

[170] Friedrich Wehrung. Tensor products of structures with interpolation. *Pacific Journal of Mathematics*, 176(1):267–285, 1996. doi:10.2140/pjm.1996.176.267.

[171] Glynn Winskel and Mogens Nielsen. Models for Concurrency. In Samson Abramski, Dov M. Gabbay, and Thomas Stephen Edward Maibaum, editors, *Handbook of Logic in Computer Science. Volume 4. Semantic Modelling*, pages 1–148. Oxford University Press, Oxford, 1995.