Journal of Logic, Language and Information
https://doi.org/10.1007/s10849-026-09461-3

Bridging concurrency theory and epistemic models: a formal framework for dynamic multi-agent systems

Alessandro Aldini[1] · **Ludovico Fusco**[1]

Accepted: 23 February 2026

Abstract
We present a formal framework encompassing concurrency theoretic and modal logic based approaches to the modeling and verification of dynamic multi-agent systems. We develop a model of computation merging classical labeled transition systems and multi-agent Kripke frames. Based on this model, which we call a Kripke labeled transition system, we provide a modal logic and a process algebra for the specification and analysis of interacting multi-agent systems. Our logic is obtained by combining proof systems for Hennessy-Milner Logic and the normal epistemic logic $\mathsf{S5}_n$ and is sound and complete with respect to its intended models. Further, we show that this logic is decidable and induces a behavioral equivalence combining classical notions of bisimulation. We prove that our process algebra is adequate for specifying the behavior of Kripke labeled transition systems and we show its effectiveness and usability through real-world examples.

Keywords Labeled transition systems · Epistemic models · Dynamic and epistemic logics · Process algebra · Dynamic multi-agent systems

1 Introduction

The study of knowledge-based interactions in dynamic multi-agent systems cuts across several fields, from logic to philosophy and theoretical computer science. In particular, concurrency theory and modal logic are at the base of two strands of research with clear, cross-cutting relations in several application domains, as, e.g., in the case of the formal verification of security properties of communication protocols (see Dechesne and Wang (2010) for a survey and, in particular, Chadha and Delaune (2009); Balliu and Dam (2011); Minami (2020); Bavendiek (2022)). The combination of approaches from these two areas is largely explored in the literature (see, e.g., Knight and Mardare

A. Aldini, L. Fusco: contributed equally to this work

✉ Alessandro Aldini
alessandro.aldini@uniurb.it

[1] Department of Pure and Applied Sciences, University of Urbino Carlo Bo, Piazza della Repubblica 13, 61029 Urbino, Italy

Published online: 11 March 2026

(2012); Knight et al. (2012); Guzman et al. (2017); Dechesne and Mousavi (2007)), but is rarely conducted on a unifying framework supporting the specification and the simultaneous analysis of dynamic and epistemic properties of systems, e.g., in a process algebraic modeling setting.

The goal of this paper is to provide a model encompassing the classical semantics based on labeled transition systems and epistemic models. This unifying framework favors not only the specification of all the ingredients of interest for the modeling of knowledge-based dynamic interacting systems but also for their analysis based on a logical framework joining the Hennessy-Milner Logic HML (Hennessy, 1980) and the standard epistemic system S5 (Ditmarsch et al., 2015). To emphasize the expressiveness and the usability of this model, we define on top of it a process-algebraic specification language for the modeling and verification of real-world multi-agent interacting systems.

In the following, we introduce Kripke labeled transition systems (Section 2), which combine labeled transition systems and (multi-modal) Kripke frames. We then define a logic to describe properties for this model, which includes the standard modal operators of epistemic logic and Hennessy-Milner logic, and provide soundness, completeness, and decidability results. We establish the equivalence relation that is decidable and induces a behavioral equivalence combining classical notions of bisimulation. We also provide an axiomatization of the logic that is sound and complete. We then define a process-algebraic language for modeling multi-agent systems with semantics based on Kripke labeled transition systems (Section 3). The language is structured by layers in order to facilitate the description of agent behavior, network topology, and communication policies at the base of any knowledge transfer. The presentation is accompanied by real-world running examples taken from the FIPA standards for heterogeneous and interacting agents and agent-based systems (Poslad, 2001). An additional case study is proposed that encompasses all the features of the language, in terms of expressiveness of modeling and analysis capabilities (Section 4). Its aim is to emphasize the usability features of the language on top of the underlying unifying framework. A comparison with related work and potential future directions concludes the paper (Section 5).

This paper extends previous work (Aldini, 2024) by proposing:

1. A theory of Kripke labeled transition systems as a general-purpose modeling paradigm for concurrent multi-agent systems. We specialize this framework to the systems of interest in this paper, and introduce the logic EHML, which integrates $\mathsf{S5}_n$ and HML (regarded as a variant of K_n) so as to enable uniform reasoning about both system dynamics and the evolution of agents' knowledge. Although EHML was originally introduced in Aldini (2024), we present here a syntactically refined version of the logic, together with a Hilbert-style proof system, and prove soundness and completeness with respect to its intended semantics. Moreover, leveraging classical decidability results for HML and $\mathsf{S5}_n$, we provide a decision procedure for provability. Finally, we position EHML within the existing literature, with particular attention to its relationship with *combining logics* and *dynamic epistemic logic* frameworks.

2. A process algebra with enriched features for conditional statements and broadcast communications. We present here a set of running examples inspired by the FIPA protocols that emphasize the new features of the specification language and the modeling capabilities of EHML.

2 A dynamic epistemic model

The model we propose derives from the combination of existing models describing the dynamics of systems. On the one hand, in the setting of concurrency theory, two main models are typically used that, basically, rely on directed graphs where the nodes represent system states and the arcs model the transitions between different states. The former is called *Kripke structure* or *Finite State Machine* (Baier & Katoen, 2008), a graph where the nodes are annotated with atomic propositions stating what is true in the system state associated with the node. The latter is called *Labeled Transition System* (LTS), a graph where the arcs are annotated with actions representing the events causing a change of system state. Several approaches are available to encode arc labelings by node labelings and vice versa, by preserving the semantics of the models, e.g., as in the case of the mapping from Kripke structures to Büchi automata. For convenience, when modeling complex systems, combinations of these models are often used.

On the other hand, in the setting of epistemic models and logics, the focus is on reasoning about knowledge from the viewpoint of agents trying to distinguish different scenarios. In this setting, the standard model is called *Kripke model*, a graph where every state (called *possible world*) is annotated with the propositions that hold in it (similarly to Kripke structures) and the transitions describe the so-called *accessibility* relation, which determines, from the viewpoint of each agent of interest, which worlds are compatible (*indistinguishable*) with her knowledge in the current world.

We start by recalling the definition of the two main models that are at the base of our proposal.

Definition 1 (*LTS*) A *labeled transition system* (*LTS*) is a triple $\mathfrak{T} = \langle S, Act, T \rangle$, where S is a countable set of *states*, Act is a non-empty finite set of *actions*, and $T \subseteq S \times Act \times S$ is a *transition relation*. Elements $\langle s, \pi, s' \rangle \in T$ are called the *transitions* of $\mathfrak{T}$. We denote by $\mathbb{LTS}$ the class of all LTSs.

A *rooted* LTS is a quadruple $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$ where $\langle S, Act, T \rangle$ is an LTS and s_0 is a distinguished *initial state*. In the setting of computation modeling, LTSs describe the evolving behavior of discrete systems, where the actions labeling the transitions represent events leading from one configuration of the system to another.

Definition 2 (*Kripke frame*) A (*multi-modal*) *Kripke frame* is a pair $\mathfrak{F} = \langle W, \{R_i\}_{i \in I} \rangle$ where W is a non-empty set of *possible worlds* and $\{R_i\}_{i \in I}$ is a family of binary *accessibility relations* over W, with I a finite set of indices. We denote by $\mathbb{K}$ the class of all Kripke frames.

By a *pointed* Kripke frame we mean a triple $\langle W, \{R_i\}_{i \in I}, w_0 \rangle$, where 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 labelled transition system $\langle S, Act, T \rangle$ can be viewed as a Kripke frame $\langle S, \{T_\pi\}_{\pi \in Act} \rangle$ where each relation T_π 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 $\langle W, \{R\}_{i \in I} \rangle$ correspond to an LTS $\langle W, I, R \rangle$, where R is a transition relation defined from $\{R_i\}_{i \in I}$ and I using the above construction.

In light of the discussion above, the following holds.

Lemma 1 *There is a one-to-one correspondence between* $\mathbb{LTS}$ *and* $\mathbb{K}$.

Remark 1 Kripke frames can be equivalently defined as triples $\mathfrak{F} = \langle W, I, r \rangle$ consisting of a non-empty set of possible worlds W, a non-empty set of indices I, and a function $r : I \to \wp(W \times W)$ assigning to each $i \in I$ an binary relation $r(i)$ over W.

Kripke frames serve as the basis of the semantics for various modal logics and, in the case of epistemic languages, allow us to reason about knowledge in terms of information accessibility. In the latter context, the set I stands for a bunch of rational agents and the relations R_i specify what worlds are compatible with each agent's knowledge base. Since this paper will focus exclusively on this perspective, from now on we will refer to multi-modal Kripke frames as *multi-agent* Kripke frames, writing Ag instead of I to highlight the fact that the indices identifying the relations (and the corresponding modal operators) refer to agents.

2.1 Kripke labeled transition systems

In this section, we introduce and discuss the computational model for the dynamic multi-agent epistemic systems we deal with in the present paper.

Definition 3 (*KLTS*) A *Kripke labeled transition system* (*KLTS*) is a quadruple

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

where:

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

The class of all KLTSs will be denoted by $\mathbb{KLTS}$.

In a KLTS, the function r associates each state s of the underlying LTS $\mathfrak{T}$ with a frame $\mathfrak{F}_s = \langle \wp(At), \{r(i, s)\}_{i \in Ag} \rangle$. It is instructive to exploit the alternative characterization provided in Remark 1: using partial function application, we parameterize r over S and split it into an S-indexed family

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

By doing so, we obtain frames of the form $\mathfrak{F}_s = \langle \wp(At), Ag, r_s \rangle$. When analyzing the behavior of a system modeled by a KLTS, the information generated as the system evolves through different configurations unfolds at two distinct levels:

- An "external" level, which pertains 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 expresses 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 distinguishing power of agent i in s, intended as her capability of distinguishing the possible worlds identified by the values of the propositions. Under the indistinguishability interpretation of epistemic logic, $r_s(i)$ expresses informational indistinguishability between possible worlds. More precisely, $\langle X, Y \rangle \in r_s(i)$ means that in s the agent i has insufficient information to establish whether we are in a state in which all and only the propositions of X hold or in a state in which all and only the propositions of Y hold. Hence, both X and Y are compatible with the knowledge base of the agent i in s. By virtue of this interpretation, in the following, we assume that the accessibility relations are equivalence relations.

Example 1 If $\langle \{p\} \cup X, \{p\} \cup Y \rangle$ belongs to $r_s(i)$ for any choice of $X, Y \in \wp(At)$, then, in s, all the possible worlds in which p holds are mutually indistinguishable from the viewpoint of agent i. If we also have that $\langle \{p\} \cup X, Y \rangle \notin r_s(i)$ whenever $p \notin Y$, we conclude that agent i distinguishes all and only the pairs of worlds differing for the valuation of p. Later, we will realize that this means that, in s, agent i *knows* the truth value of p and is ignorant of any other proposition.

2.2 A logic for KLTSs

The formal characterization of system dynamics, used for describing (and verifying) behavioral properties, relies on different types of modal logics.

In state-based approaches, the goal is to specify *what* happens in states during particular executions. In such a setting, the underlying model is temporal, as the different configurations correspond to "snapshots" of the system at specific instants of time. Depending on the temporal framework of reference (linear- or branching-time), behavioral properties can be expressed in a wide variety of formalisms, ranging from the classic LTL (Pnueli, 1977), CTL (Clarke, 1982), and CTL* (Emerson, 1983) to hyperlogics (Coenen et al., 2019).

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, behavioral properties are most effectively captured using dynamic logics (Harel et al., 2000), 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 Pratt (1976) to provide a semantical counterpart to Hoare Logic (Hoare, 1969). The simplest dynamic logic is HML (Hennessy & Milner, 1985), which extends classical propositional logic with connectives of the forms $\langle \pi \rangle$ and $[\pi]$ specifying conditions satisfied by a system in states reached by executing π. It is important to remark that HML and CTL* (which, as is well known, includes both LTL and CTL) can easily be subsumed under the umbrella of the modal μ-calculus (Kozen, 1983).

In both of the frameworks mentioned above, the modeling of computational infrastructures is performed 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. Examples include Alternating-time Temporal Logic (ATL) (Alur et al., 2002), Strategic Logic (SL) (Chatterjee et al., 2010), Alternating-time Temporal Epistemic Logic (ATEL) (Hoek, 2002) and many extensions of standard temporal logics with agent-based modalities (e.g., Rybakov (2009)). On the other hand, extensions of action-based logics have been successfully applied to the modeling and verification of multi-agent systems, starting from the rich framework of Dynamic Epistemic Logic (DEL) (Van Ditmarsch & Van Der Hoek, 2008).

In this section, we adopt a multi-agent perspective and present a logic for the analysis of KLTSs, which we call EHML (for Epistemic Hennessy-Milner Logic). Our logic combines HML with the normal multimodal logic $\mathsf{S5}_n$ so as to model the evolution of the knowledge of a fixed group of agents over an underlying LTS structure. However, unlike previous work in this direction–most notably (Knight & Mardare, 2012)–our formalism prevents agents from having complete knowledge of the external behavior of a KLTS, while still granting them epistemic access to a portion of the information generated by the system. Indeed, for the kind of non-deterministic systems we intend

to model, it is not intuitive to allow agents to know how the system will evolve, thus it only makes sense to "access" the information contained in the frames associated with the states when it is necessary to evaluate formulas concerning the agents' knowledge. In light of these observations, we base the syntax of EHML on an initial layer of $\mathsf{S5}_n$-formulas, which we refer to as *internal*.

Definition 4 (*Internal formula*) 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 following 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 (EHML-*formula*) 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 4 and 5, 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 $\mathrm{sub}_+(\varphi) := \{\lfloor\psi\rfloor \in \lfloor Int\rfloor \mid \lfloor\psi\rfloor \in \mathrm{sub}(\varphi)\}$. 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 $\varphi_1, \ldots, \xi_n$ 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 6 (*Model over a KLTS*) Let $\mathfrak{S} = \langle\mathfrak{T}, At, Ag, r\rangle$ be a KLTS with underlying LTS $\mathfrak{T} = \langle S, Act, T, s_0\rangle$. A *model* over $\mathfrak{S}$ is a pair $\mathfrak{M} = \langle\mathfrak{S}, v\rangle$, where $v : S \rightarrow \wp(At)$ is a valuation function.

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

Definition 7 (*s-model*) Let $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ be a model over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$. For a state $s \in S$, the *s-model* relative to $\mathfrak{M}$ is the Kripke model $\mathfrak{M}_s = \langle \mathfrak{F}_s, \mathrm{id} \rangle$, where $\mathfrak{F}_s = \langle \wp(At), Ag, r_s \rangle$ 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 8 (EHML-*model*) Let $\mathfrak{M}$ be a model over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with underlying LTS $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$. We say that $\mathfrak{M}$ is an EHML-*model* if the direct image $r[Ag \times S]$ of r is a family of equivalence relations on $\wp(At)$. Lastly, we define:

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

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

Definition 9 (*Satisfaction of internal formulas*) Let $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ be an EHML-model over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$. For a state $s \in S$, the *satisfaction* of an internal formula generated over At with respect to the s-model $\mathfrak{M}_s = \langle \mathfrak{F}_s, \mathrm{id} \rangle$ and a world $X \in \wp(At)$ is recursively defined as follows:

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

Definition 10 (*Satisfaction of* EHML-*formulas*) Let $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ be an EHML-model over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$. We define the *satisfaction* of an EHML-formula generated over At_+ with respect to $\mathfrak{M}$ and a state $s \in S$ as follows:

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

Definition 11 (*Truth and validity*) Let φ be an EHML-formula.

1. φ is *true* in an EHML-model $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ (written $\mathfrak{M} \Vdash \varphi$) if $\mathfrak{M}, s \Vdash \varphi$, for each state s of $\mathfrak{S}$;
2. φ is *valid in* $\mathfrak{S} \in \mathbb{EHML}$ (written $\mathfrak{S} \Vdash \varphi$) if it is true in every EHML-model over $\mathfrak{S}$;
3. φ is *valid in* $\mathbb{EHML}$ (written $\mathbb{EHML} \Vdash \varphi$) if $\mathfrak{S} \Vdash \varphi$, for all $\mathfrak{S} \in \mathbb{EHML}$.

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

Definition 12 (*Semantic consequence and equivalence in* $\mathbb{EHML}$) Let $\Gamma \cup \{\varphi\} \subseteq Fm$. The formula φ is a (*local*) *semantic consequence* of Γ over $\mathbb{EHML}$ (henceforth, $\mathbb{EHML}$-*consequence*) if $\mathfrak{M}, s \Vdash \Gamma$ implies $\mathfrak{M}, s \Vdash \varphi$, for every EHML-model $\mathfrak{M}$ and every state s of $\mathfrak{M}$. Two EHML-formulas φ_1 and φ_2 are $\mathbb{EHML}$-*equivalent* if and only if each is an $\mathbb{EHML}$-consequence of the other.

Throughout this paper, we shall also make use of the standard notions of validity and semantic consequence for $\mathsf{S5}_n$- and HML-formulas. Let $\mathbb{S}5$ denote the class of reflexive, transitive, and symmetric Kripke frames. For $\psi \in Int$, we write $\mathbb{S}5 \Vdash \psi$ to indicate that ψ is valid in $\mathbb{S}5$. Given a pure HML-formula φ, we write $\mathbb{LTS} \Vdash \varphi$ (resp. $\mathbb{K} \Vdash \varphi$) to mean that φ is valid in $\mathbb{LTS}$ (resp. $\mathbb{K}$). Local semantic consequence over $\mathbb{S}5$ and $\mathbb{LTS}/\mathbb{K}$ will be expressed using the same notation as in Definition 12, with the reference class indicated explicitly each time.

We now analyze in greater detail the semantic consequences of allowing delimited formulas in the syntax of EHML. First, we show that such formulas can be equivalently satisfied at a state s of an EHML-model or at the state $v(s)$ of the corresponding s-model, provided that $\psi \in At \cup \{\top\}$, or its principal connective is a Boolean operator. In other words, satisfaction of atomic formulas is invariant under delimitation, and the application of delimiters "commutes" with negation and conjunction.

Lemma 2 *Let $p \in At$ and let $\psi, \psi_1, \psi_2 \in Int$. Then, for any* EHML-*model $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$, and any state $s \in S$:*

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

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

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

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

Proof The proof of (INV_p) follows immediately from condition (a) in Definition 9 together with conditions (1) and (1_+) in Definition 10. 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 13 (*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 1 *Every $\varphi \in Fm$ can be transformed into an equivalent $\varphi^\dagger \in DeNF$.*

Proof Immediate from Lemma 2. □

Definition 14 Let φ be an EHML-formula. If $\mathrm{sub}_+(\varphi) = \{\xi_1, \ldots, \xi_n\}$, we denote by φ^{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 φ^{p} as an alternative for φ.

Remark 2 The superscript $^{\mathrm{p}}$ 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 Γ^{p} the set of all formulas φ^{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 the same substitution is performed in φ'.

Proposition 2 *Let $\langle \mathfrak{S}, v\rangle$ be an* EHML*-model over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r\rangle$ with $\mathfrak{T} = \langle S, Act, T, s_0\rangle$. Then, for $\varphi \in Fm$ and $s \in S$, there exists $v' \colon S \to \wp(At)$ such that $\langle \mathfrak{S}, v\rangle, s \Vdash \varphi$ iff $\langle \mathfrak{S}, v'\rangle, s \Vdash \varphi^{\mathrm{p}}$.*

Proof Left to the reader. □

The transformation of EHML-formulas into pure HML-formulas introduced in Definition 14 allows for a clearer understanding of the notion of validity in $\mathbb{EHML}$. Trivially, any formula $\varphi \in Fm$ whose pure HML counterpart is valid in $\mathbb{LTS}$ is also valid in $\mathbb{EHML}$. 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, $\mathbb{EHML} \Vdash \varphi \wedge \lfloor\psi\rfloor$, as $\mathbb{LTS} \Vdash \varphi^{\mathrm{p}}$ and $\mathbb{S}5 \Vdash \psi$. However, $(\varphi \wedge \lfloor\psi\rfloor)^{\mathrm{p}}$ has the form $\varphi \wedge r$, which is plainly not valid in $\mathbb{LTS}$.

Lemma 3 *Let $\varphi \in Fm$. The following conditions hold:*

(a) if $\mathrm{sub}_+(\varphi) = \emptyset$, then $\mathbb{EHML} \Vdash \varphi$ iff $\mathbb{LTS} \Vdash \varphi$;
(b) if $\mathrm{sub}_+(\varphi) \neq \emptyset$, then $\mathbb{LTS} \Vdash \varphi^{\mathrm{p}}$ implies $\mathbb{EHML} \Vdash \varphi$;
(c) if $\mathbb{EHML} \Vdash \varphi$ and $\mathbb{LTS} \nVdash \varphi^{\mathrm{p}}$, then there exists $\lfloor\Delta\rfloor \subseteq \lfloor Int\rfloor$ such that $\mathbb{S}5 \Vdash \Delta$, and φ is an $\mathbb{EHML}$-consequence of $\lfloor\Delta\rfloor$.

Proof The proof of (a) and (b) is immediate. As for (c), suppose that $\mathbb{EHML} \Vdash \varphi$ while $\mathbb{LTS} \nVdash \varphi^{\mathrm{p}}$. Then the validity of φ in $\mathbb{EHML}$ cannot (at least entirely) be accounted for by the LTS component of $\mathbb{EHML}$-models. It follows that $\mathrm{sub}_+(\varphi) \neq \emptyset$: otherwise, by (a) and Definition 14, φ^{p}, or equivalently φ, would be valid over $\mathbb{LTS}$, contradicting the initial assumption. Three cases can then be distinguished. If $\varphi = \lfloor\psi\rfloor$ with $\mathbb{S}5 \Vdash \psi$, then we take $\lfloor\Delta\rfloor = \emptyset$, and the claim follows. If, by applying the delimiter-manipulation rules of Lemma 2, φ is $\mathbb{EHML}$-equivalent to $\lfloor\psi\rfloor$ with $\mathbb{S}5 \Vdash \psi$, then we take $\lfloor\Delta\rfloor = \{\lfloor\psi\rfloor\}$, and the claim follows. Finally, if φ contains dynamic subformulas and $\mathbb{LTS} \nVdash \varphi^{\mathrm{p}}$, then, by applying some (possibly none) of the delimiter-manipulation rules of Lemma 2, φ can be transformed into an $\mathbb{EHML}$-equivalent formula φ' such that either $\mathbb{LTS} \Vdash (\varphi')^{\mathrm{p}}$, or the $\mathbb{EHML}$-validity of φ' is determined by the $\mathbb{S}5$-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 $\lfloor\Delta\rfloor = \lfloor\Delta'\rfloor$. □

HML axioms

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

$\mathsf{S5}_n^d$ 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

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

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

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

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

Fig. 1 Hilbert-style proof system for EHML

We now proceed to introduce a Hilbert-style proof system for EHML, whose axioms and inference rules are listed in Figure 1. To analyze KLTS dynamics, the system includes an HML component, which is defined by the axioms (CL), (K_π), and (DUAL$_\pi$), together with the rules (MP), (NEC$_\pi$), and (SUB). Note that, in light of Lemma 1, 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$\neg$), (D$\wedge$), 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 2. 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 axiomatized 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 4 (MP^d) *is derivable in* EHML.

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

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

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

1.	$\lfloor\psi_1\rfloor$	d_1
2.	$\lfloor\psi_1 \rightarrow \psi_2\rfloor$	d_2
3.	$\lfloor\psi_1\rfloor \rightarrow \lfloor\psi_2\rfloor$	(D$\rightarrow$), 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 $\lfloor\psi_1 \rightarrow \psi_2\rfloor$, 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. It remains, however, to account for the hybrid rules (D$\neg$), (D$\wedge$), and (INV). We begin by observing that:

Proposition 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, $\vdash_{\mathsf{EHML}} \lfloor\psi\rfloor^{\dagger}$.*

Proposition 4 *Let $\Gamma \cup \{\varphi\} \subseteq Fm$ and $\Delta \cup \{\psi\} \subseteq Int$. Then:*

1. *$\Gamma \vdash_{\mathsf{HML}} \varphi$ iff $\Gamma^p \vdash_{\mathsf{K}_n} \varphi^p$, for $(\Gamma \cup \{\varphi\})^p$ constructed as in Remark 2;*
2. *$\lfloor\Delta\rfloor \vdash_{\mathsf{S5}_n^d} \lfloor\psi\rfloor$ iff $\Delta \vdash_{\mathsf{S5}_n} \psi$.*

Proof Left to the reader. □

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 *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 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 *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 6 (Completeness theorem for EHML) *Let φ be an* EHML*-formula. Then*

$$\vdash_{\mathsf{EHML}} \varphi \quad \textit{iff} \quad \mathbb{EHML} \Vdash \varphi.$$

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 9, 10, 11, and Lemmas 2 and 5. As for the converse, suppose that $\nvdash_{\mathsf{EHML}} \varphi$. By Lemma 5, 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:

$$\begin{array}{rll} \nvdash_{\mathsf{S5}_n^d} \lfloor \psi \rfloor & \Leftrightarrow \nvdash_{\mathsf{S5}_n} & \text{by Proposition 4(2)} \\ & \Leftrightarrow \mathbb{S}5 \nVdash \psi & \text{by weak completeness of } \mathsf{S5}_n \text{ Halpern and Moses (1992)} \\ & \Leftrightarrow \mathbb{EHML} \nVdash \lfloor \psi \rfloor & \text{by Definitions 10 and 11} \end{array}$$

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

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

$$\begin{array}{rll} \nvdash_{\mathsf{HML}} \varphi & \Leftrightarrow \nvdash_{\mathsf{K}_n} \varphi^{\mathrm{p}} & \text{by Proposition 4(1)} \\ & \Leftrightarrow \mathbb{K} \nVdash \varphi^{\mathrm{p}} & \text{by weak completeness of } \mathsf{K}_n \text{ Halpern and Moses (1992)} \\ & \Leftrightarrow \mathbb{LTS} \nVdash \varphi^{\mathrm{p}} & \text{by Lemma 1} \\ & \Leftrightarrow \mathbb{EHML} \nVdash \varphi & \text{by Proposition 2} \end{array}$$

   (B) Let $\Phi \neq \emptyset$. We are thus in the situation where, for every nonempty 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 $\mathbb{EHML}$, we have $\mathbb{EHML} \Vdash \alpha$ for every $\alpha \in \Phi$. Assuming that $(\Phi \cup \{\varphi\})^{\mathrm{p}}$ is constructed following Remark 2, we have:

$$\begin{array}{rll} \Phi \nvdash_{\mathsf{HML}} \varphi & \Leftrightarrow \Phi^{\mathrm{p}} \nvdash_{\mathsf{K}_n} \varphi^{\mathrm{p}} & \text{by Proposition 4(1)} \\ & \Leftrightarrow \Phi^{\mathrm{p}} \nVdash \varphi^{\mathrm{p}} \text{ in } \mathbb{K} & \text{by strong completeness of } \mathsf{K}_n \\ & & \text{Van Ditmarsch et al. (2008)} \\ & \Leftrightarrow \exists \mathfrak{F} \in \mathbb{K} : \mathfrak{F} \Vdash \Phi^{\mathrm{p}} \text{ and } \mathfrak{F} \nVdash \varphi^{\mathrm{p}} & \text{by def. of } \mathbb{K}\text{-consequence} \\ & \Leftrightarrow \exists \mathfrak{T} \in \mathbb{LTS} : \mathfrak{T} \Vdash \Phi^{\mathrm{p}} \text{ and } \mathfrak{T} \nVdash \varphi^{\mathrm{p}} & \text{by Lemma 1} \end{array}$$

   We conclude that $\mathbb{LTS} \nVdash \varphi^{\mathrm{p}}$. Since φ^{p} is a pure HML-formula, by Lemma 3(a) we have that $\mathbb{EHML} \Vdash \varphi^{\mathrm{p}}$ iff $\mathbb{LTS} \Vdash \varphi^{\mathrm{p}}$. Hence, $\mathbb{EHML} \nVdash \varphi^{\mathrm{p}}$. We need to show that $\mathbb{EHML} \nVdash \varphi$. Suppose, towards a contradiction, that $\mathbb{EHML} \Vdash \varphi$. Then the antecedent of Lemma 3(c) is satisfied. It follows that there exists a set $\lfloor \Delta \rfloor$ of delimited $\mathbb{S}5$-valid formulas such that φ is an $\mathbb{EHML}$-consequence of $\lfloor \Delta \rfloor$. We therefore reconsider the three cases distinguished in Lemma 3(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 $\mathbb{EHML} \nVdash \varphi$;
   (B2) by applying the delimiter-manipulation rules of Lemma 2, φ is $\mathbb{EHML}$-equivalent to $\lfloor \psi \rfloor$ with $\mathbb{S}5 \Vdash \psi$. Again, φ would be a theorem of E, and

we would have $\{\varphi\} \vdash_{\mathsf{HML}} \varphi$, contradicting the initial assumption. Hence, $\mathbb{EHML} \nVdash \varphi$;

(B3) φ contains dynamic subformulas and, by applying some (possibly none) delimiter-manipulation rules, φ can be transformed into an $\mathbb{EHML}$-equivalent formula φ' such that either $\mathbb{LTS} \Vdash (\varphi')^{\mathrm{p}}$, or the $\mathbb{EHML}$-validity of φ' is determined by the $\mathbb{S}5$-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 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 $\mathbb{EHML} \nVdash \varphi$.

□

Example 2 We provide a simple example to clarify item (B3) in the above proof. Let

$$\alpha_1 := [\pi](p \wedge \lfloor\mathbf{K}_j q\rfloor) \rightarrow [\pi]p \qquad \psi := \mathbf{K}_i p \rightarrow \neg\mathbf{K}_i\neg p \qquad \alpha_2 := \lfloor\mathbf{K}_i p\rfloor \rightarrow \lfloor\neg\mathbf{K}_i\neg p\rfloor$$

Clearly $\mathbb{S}5 \Vdash \psi$, so we have $\mathbb{EHML} \Vdash \lfloor\psi\rfloor$, and by Lemma 2, $\lfloor\psi\rfloor$ and α_2 are $\mathbb{EHML}$-equivalent. It is immediate to give an HML-proof of α_1:

1. $(p \wedge \lfloor\mathbf{K}_j q\rfloor) \rightarrow p$	CL	
2. $[\pi]((p \wedge \lfloor\mathbf{K}_j q\rfloor) \rightarrow p) \rightarrow ([\pi](p \wedge \lfloor\mathbf{K}_j q\rfloor) \rightarrow [\pi]p)$	(K_π)	(d_{α_1})
3. $[\pi]((p \wedge \lfloor\mathbf{K}_j q\rfloor) \rightarrow p)$	NEC_π, 1	
4. $[\pi](p \wedge \lfloor\mathbf{K}_j q\rfloor) \rightarrow [\pi]p$	MP, 2, 4	

Therefore $\mathbb{LTS} \Vdash \alpha_1^{\mathrm{p}}$ and, by Lemma 3(b), $\mathbb{EHML} \Vdash \alpha_1$. Let now $\varphi := \alpha_1 \wedge \alpha_2$. Clearly, $\mathbb{EHML} \Vdash \varphi$ whereas $\mathbb{LTS} \nVdash \varphi^{\mathrm{p}}$. Letting $\varphi' := \alpha_1 \wedge \lfloor\psi\rfloor$, one readily verifies that φ and φ' are $\mathbb{EHML}$-equivalent. Note that $\mathbb{EHML} \Vdash \varphi'$ *because* $\mathbb{S}5 \Vdash \psi$, 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. $\alpha_1 \rightarrow (\lfloor\psi\rfloor \rightarrow (\alpha_1 \wedge \lfloor\psi\rfloor))$	(CL)	$(d_{\varphi'})$
4. $\lfloor\psi\rfloor \rightarrow (\alpha_1 \wedge \lfloor\psi\rfloor)$	(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 $\varepsilon := \lfloor\psi\rfloor \leftrightarrow \alpha_2$:

$$\begin{array}{lll} 1.\ \lfloor(\mathbf{K}_i\, p \rightarrow \neg\mathbf{K}_i\neg p) \leftrightarrow (\mathbf{K}_i\, p \rightarrow \neg\mathbf{K}_i\neg p)\rfloor & \text{CL}^d & \\ 2.\ \lfloor\mathbf{K}_i\, p \rightarrow \neg\mathbf{K}_i\neg p\rfloor \leftrightarrow \lfloor\mathbf{K}_i\, p \rightarrow \neg\mathbf{K}_i\neg p\rfloor & (\text{D}\leftrightarrow),\ 1 & (d_\varepsilon) \\ 3.\ \lfloor\mathbf{K}_i\, p \rightarrow \neg\mathbf{K}_i\neg p\rfloor \leftrightarrow (\lfloor\mathbf{K}_i\, p\rfloor \rightarrow \lfloor\neg\mathbf{K}_i\neg p\rfloor) & (\text{D}\rightarrow),\ 1 & \end{array}$$

We can now obtain a derivation of φ from $\lfloor\psi\rfloor$ and ε:

$$\begin{array}{lll} 1.\ \alpha_1 \wedge \lfloor\psi\rfloor & (d_{\varphi'}) & \\ 2.\ \lfloor\psi\rfloor \leftrightarrow \alpha_2 & (d_\varepsilon) & (d_\varphi) \\ 3.\ \alpha_1 \wedge \lfloor\psi\rfloor & \text{Proposition 5, 1, 2} & \end{array}$$

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 Halpern and Moses (1992) for further details).

Theorem 7 EHML *is decidable.*

Proof Let φ be an EHML-formula. If $\varphi = \lfloor\psi\rfloor$, then, by Lemma 5 and Proposition 4, $\vdash_{\mathsf{EHML}} \lfloor\psi\rfloor$ 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 $\vdash_{\mathsf{K}_n} \varphi^{\mathrm{p}}$, then, by Proposition 4, $\vdash_{\mathsf{EHML}} \varphi$. Otherwise, we proceed to Step 2.
2. We construct the formula $\varphi_{\min}$ (which is $\mathbb{EHML}$-equivalent to φ by Lemma 2) 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 $\mathbb{EHML}$-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 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 $\mathbb{EHML}$-equivalent to α by Lemma 2) 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 $(\alpha')^{\mathrm{p}}$. If

$\vdash_{\mathsf{K}_n} (\alpha')^{\mathfrak{p}}$, then $\vdash_{\mathsf{EHML}} \alpha'$, therefore, by Theorem 6, $\vdash_{\mathsf{EHML}} \varphi_{\min}$, whence $\vdash_{\mathsf{EHML}} \varphi$; otherwise, $\nvdash_{\mathsf{EHML}} (\alpha')^{\mathfrak{p}}$ 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. □

EHML brings about the following notion of behavioral equivalence.

Definition 15 (*Bisimulation*) Let $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ be a model over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, Ag, r \rangle$ with $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$. An equivalence relation $\mathcal{B} \subseteq S \times S$ is a *bisimulation* iff whenever $\langle s, t \rangle \in \mathcal{B}$ 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 $\langle s', t' \rangle \in \mathcal{B}$;
3. there exists an equivalence relation $\mathcal{B}_{st}$ between the worlds of the Kripke models $\mathfrak{M}_s = \langle \mathfrak{F}_s, \mathrm{id} \rangle$ (with $\mathfrak{F}_s$ pointed at $v(s)$) and $\mathfrak{M}_t = \langle \mathfrak{F}_t, \mathrm{id} \rangle$ (with $\mathfrak{F}_t$ pointed at $v(t)$) such that $\langle v(s), v(t) \rangle \in \mathcal{B}_{st}$ and for any $X, Y \in \wp(At)$, whenever $\langle X, Y \rangle \in \mathcal{B}_{st}$ 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 $\langle X', Y' \rangle \in \mathcal{B}_{st}$.

Conditions 1. and 3. resemble the definition of modal bisimulation for Kripke models (Blackburn & De Rijke, 2002), while condition 2. characterizes the strong bisimulation for LTSs (Hennessy, 1980). Two states s and t of a model $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ are bisimilar, written $s \sim_{\mathfrak{M}} t$, if and only if there exists a bisimulation $\mathcal{B}$ such that $\langle s, t \rangle \in \mathcal{B}$. 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 16 (*Modal equivalence*) Let $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ be an EHML-model. Two states s, s' of $\mathfrak{S}$ are *modal equivalent* (denoted $s \equiv_{\mathfrak{M}} s'$) if, for any EHML-formula φ:

$$\mathfrak{M}, s \Vdash \varphi \quad \text{iff} \quad \mathfrak{M}, s' \Vdash \varphi.$$

It is easy to check that $\equiv_{\mathfrak{M}}$ is an equivalence relation.

Theorem 8 *For any two states s, t of a image-finite model $\mathfrak{M}$, $s \sim_{\mathfrak{M}} t$ iff $s \equiv_{\mathfrak{M}} t$.*

Proof The proof is an adaptation of standard approaches (Blackburn & De Rijke, 2002; Hennessy, 1980).

Case ($\Rightarrow$): $s \sim_{\mathfrak{M}} t$ implies $s \equiv_{\mathfrak{M}} t$.

Let us assume that there exists a bisimulation $\mathcal{B}$ including the pair $\langle s, t \rangle$. We will show the result through an induction on the structure of the formulas.

The base case refers to the atomic formulas and trivially holds by definition of bisimulation since $v(s) = v(t)$. We leave to the reader the proof that no classical formula can distinguish bisimilar states.

In the case of a formula $\langle \pi \rangle \varphi$, suppose that $\mathfrak{M}, s \Vdash \langle \pi \rangle \varphi$. Hence there exists $s' \in S$ such that $\langle s, \pi, s' \rangle \in T$ and $\mathfrak{M}, s' \Vdash \varphi$. By Definition 15, there exists $t' \in S$ such that

$\langle t, \pi, t' \rangle \in T$ and $\langle s', t' \rangle \in \mathcal{B}$. By the induction hypothesis, it holds that $\mathfrak{M}, t' \Vdash \varphi$ whence $\mathfrak{M}, t \Vdash \langle \pi \rangle \varphi$, and therefore s and t cannot be distinguished by EHML-formulas prefixed by dynamic modalities.

As for the case of epistemic formulas, the only interesting case is $\mathfrak{M}, s \Vdash \lfloor \mathbf{K}_i \psi \rfloor$. Thus, by Definition 9, $\mathfrak{M}_s, v(s) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i \psi$. By Definition 15, there exists an equivalence relation $\mathcal{B}_{st}$ between the worlds of $\mathfrak{M}_s = \langle \mathfrak{F}_s, \mathrm{id} \rangle$ (with $\mathfrak{F}_s$ pointed at $v(s)$) and $\mathfrak{M}_t = \langle \mathfrak{F}_t, \mathrm{id} \rangle$ (with $\mathfrak{F}_t$ pointed at $v(t)$) such that $\langle v(s), v(t) \rangle \in \mathcal{B}_{st}$.

We now show that $\mathfrak{M}_s$ and $\mathfrak{M}_t$ satisfy the same internal formulas and, therefore, $\mathfrak{M}_t, v(t) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i \psi$, from which we derive $\mathfrak{M}, t \Vdash \lfloor \mathbf{K}_i \psi \rfloor$.

- Base: Since $v(s) = v(t)$, $\mathfrak{M}_s, v(s) \Vdash_{\mathsf{S5}_n} p$ iff $\mathfrak{M}_t, v(t) \Vdash_{\mathsf{S5}_n} p$, for all $p \in At$.
- Induction: Trivially, no formula constructed using only Boolean connectives can distinguish $v(s)$ from $v(t)$. So, let us consider $\mathfrak{M}_s, v(s) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i \psi$. Now, let X be such that $\langle v(s), X \rangle \in r_s(i)$. Hence we have $\mathfrak{M}_s, X \Vdash_{\mathsf{S5}_n} \psi$. By Definition 15, there exists Y such that $\langle v(t), Y \rangle \in r_t(i)$ with $\langle X, Y \rangle \in \mathcal{B}_{st}$ and, by induction hypothesis, $\mathfrak{M}_t, Y \Vdash_{\mathsf{S5}_n} \psi$. As a consequence, again by Definition 15, $\mathfrak{M}_t, v(t) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i \psi$, as expected.

Therefore, having a bisimulation relating two states is sufficient for the two states to verify the same EHML-formulas.

Case ($\Leftarrow$): $s \equiv_{\mathfrak{M}} t$ implies $s \sim_{\mathfrak{M}} t$.

We will show (by contradiction) that $\equiv_{\mathfrak{M}}$ is a bisimulation itself.

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 $s' \equiv_{\mathfrak{M}} t'$. 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 $\mathfrak{M}, s' \Vdash \varphi_j$ and $\mathfrak{M}, t' \nVdash \varphi_j$. Hence, it holds that $\mathfrak{M}, s \Vdash \langle \pi \rangle \bigwedge_j \varphi_j$ (since there is s' such that $\langle s, \pi, s' \rangle \in T$ and satisfying $\bigwedge_j \varphi_j$), and $\mathfrak{M}, t \nVdash \langle \pi \rangle \bigwedge_j \varphi_j$, thus contradicting the hypothesis. The same kind of reasoning applies to the symmetric case.

Third, assume that the worlds of the Kripke models $\mathfrak{M}_s = \langle \mathfrak{F}_s, \mathrm{id} \rangle$ (with $\mathfrak{F}_s$ pointed at $v(s)$) and $\mathfrak{M}_t = \langle \mathfrak{F}_t, \mathrm{id} \rangle$ (with $\mathfrak{F}_t$ pointed at $v(t)$) are not related by a binary equivalence relation $\mathcal{B}_{st}$ including $\langle v(s), v(t) \rangle$ as given in Definition 15. 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 $\langle X, Y \rangle \in \mathcal{B}_{st}$. 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 $\mathfrak{M}_s, X \Vdash_{\mathsf{S5}_n} \psi_j$ and $\mathfrak{M}_t, Y \nVdash_{\mathsf{S5}_n} \psi_j$. But then $\mathfrak{M}_s, X \Vdash_{\mathsf{S5}_n} \neg \mathbf{K}_i \neg \bigwedge_j \psi_j$ (since there is X, accessible from $v(s)$, satisfying $\bigwedge_j \psi_j$) and $\mathfrak{M}_t, v(t) \nVdash_{\mathsf{S5}_n} \neg \mathbf{K}_i \neg \bigwedge_j \psi_j$ (since by assumption and the semantics of the epistemic operator, $\mathfrak{M}_t, v(t) \Vdash_{\mathsf{S5}_n} \mathbf{K}_i \neg \bigwedge_j \psi_j$), thus contradicting the hypothesis. The same kind of reasoning applies to the symmetric case.

Since the modal equivalence $\equiv_{\mathfrak{M}}$ satisfies the three conditions of Definition 15, it is a bisimulation and the proof is completed. □

3 A process algebra with KLTS semantics

Our approach to defining a KLTS-based language for modeling multi-agent systems is incremental. The first step consists of defining a process calculus inspired by classical CCS-like calculi with value passing (Fokkink, 2007; Gorrieri & Versari, 2015; Hennessy, 1991; Huang et al., 2012), which we use as the basis for the description of sequential process terms in isolation, the semantics of which is expressed as LTSs. Secondly, we will introduce the notion of an agent, which possesses a unique identity and is characterized by a behavior defined as a process term. Thirdly, we will define networks (named pools) of communicating epistemic agents, the semantics of which will be given in terms of KLTSs. The various ingredients and phases of our specification methodology will be illustrated through a running example.

3.1 Running example: FIPA interaction protocols

The FIPA (Foundation for Intelligent Physical Agents) standard specifications (Poslad, 2001) describe basic interaction protocols for multi-agent systems that support interoperability, information exchange, and open service interaction (Poslad, 2007). 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 behavior of agents for two FIPA protocol standards:

- Query interaction protocol: 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: 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.

3.2 Modeling agents

We start by presenting a calculus 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 (ranging over $a, b, \ldots$), including the special name τ representing an unobservable internal action. We will use A to denote $A_\tau \backslash \{\tau\}$. Moreover, let *set* and *in* be two additional special action names. To model value passing (Hennessy, 1991; Huang et al., 2012), we will use variables ($x, y, \ldots, f, g, \ldots$),

values (v, v', ...) from fixed domains, and expressions (e, e', ...) that usually represent simple values.

The calculus for sequential processes is based on the operators of action prefix, nondeterministic choice, and recursion.

Definition 17 (*Syntax of Process Terms*) For At a fixed set of atomic propositions and Ag a set of agent variables, the set $\mathcal{L}$ of process terms of the calculus for sequential processes is generated through the following syntax:

$$P ::= \underline{0} \mid \textstyle\sum_{k \in I} \pi_k \,.\, P_k \mid C(e_1, \ldots, e_n)$$
$$\pi ::= [\psi]b \mid a(y, f) \mid \bar{a}(J, \psi) \mid set(p, w)$$

where I is any finite indexing set, C is a constant name with the natural $n \geq 0$ being the arity of C, $b \in A_\tau$, ψ is any internal EHML-formula, $a \in A$, $J \subseteq Ag$, $p \in At$, 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 $k \in I$, 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 we will see formally in the next Section, the reading of $[\psi]b$ will be "the action b is executed if ψ is known by the agent". This notation is not to be confused with the inverse notation $[\pi]\varphi$ expressing that executing action π must make φ hold. In the following, we will use the abbreviation b to stand for $[\top]b$.
- The input action $a(y, f)$ specifies that when executing the input 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 output action named a the internal 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 occurrence of any variable in a process term P is bound by either an input action or a constant definition. 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.

Table 1 Semantics rules for sequential processes

$$\pi \,.\, P \xrightarrow{\pi} P \;\; \pi \text{ any of } \{[\psi]b, \bar{a}(i,\psi), set(p,w)\}$$

$$a(y,f) \,.\, P \xrightarrow{a(i,\psi)} P[i/y, \psi/f] \;\; \textit{For any } i \in Ag \text{ and internal formula } \psi$$

$$\frac{\pi \,.\, P \xrightarrow{\pi'} P}{\pi \,.\, P + E \xrightarrow{\pi'} P}$$

$$\frac{P[v_1/x_1, \ldots, v_n/x_n] \xrightarrow{\pi} P'}{C(e_1, \ldots, e_n) \xrightarrow{\pi} P'} \; C(x_1, \ldots, x_n) := P \text{ and each } e_i \text{ evaluates to } v_i$$

Definition 18 (*Semantics of Process Terms*) The behavior of a process term $P \in \mathcal{L}$ 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 1.

Let us illustrate the semantic rules of Table 1. The first rule formalizes the behavior of the prefix operator whenever the action is different from an input action. Note that the rule states that the action is enabled without considering any pre- and post-conditions concerned with the action parameters. Pre- and post-conditions will be introduced only when defining the semantics of the parallel composition of agents and the structures characterizing the knowledge of these agents.

The second rule describes the behavior 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 premise), 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 (Gorrieri & Versari, 2015; Huang et al., 2012).

3.2.1 From process terms to agents

Process terms of the calculus defined above represent behavioral patterns. An *agent* is a specific instance of a process term with a unique identity.

Definition 19 (*Agent*) An agent is described by a pair $\langle i, P \rangle$, where $i \in Ag$ and $P \in \mathcal{L}$. Given $\langle i, P \rangle$, we mean that P is the local behavior of agent i.

We will use $\mathcal{I}, \mathcal{J}, \ldots$ to range over the set of agent pairs and sometimes we will adopt the notation $\mathcal{I}_i$ to denote a pair $\langle i, P \rangle$.

The semantics of $\langle i, P \rangle$ is given by the LTS expressing the behavior of P, up to the renaming of the actions as defined by the 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 behavioral 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 3 (*Specification of the FIPA query interaction protocol*) 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:

$$\begin{aligned} \mathit{Participant} &:= \textstyle\sum_{p \in Pr} \mathit{query\text{-}if}_p(x, \_).(\\ &\quad \mathit{set}(acc, 0).\overline{\mathit{notify}}(x, \neg acc).\mathit{Participant} + \\ &\quad \mathit{set}(acc, 1).\overline{\mathit{notify}}(x, acc).\mathit{Query\_Exec}_p(x)) \\ \mathit{Query\_Exec}_p(x) &:= \overline{\mathit{failure}}(x, \top).\mathit{Participant} + \\ &\quad \overline{\mathit{inform\text{-}t/f}}(x, p).\mathit{Participant} + \overline{\mathit{inform\text{-}t/f}}(x, \neg p).\mathit{Participant} \end{aligned}$$

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 Initiator agent behaves as follows:

$$\begin{aligned} \mathit{Initiator} &:= \textstyle\sum_{p \in Pr} \overline{\mathit{query\text{-}if}_p}(\mathit{MySQL\_Server}, \top).\mathit{notify}(\_, g_1).\mathit{Init\_Wait} \\ \mathit{Init\_Wait} &:= [\neg acc]\mathit{is\_refuse}.\mathit{Initiator} + [acc]\mathit{is\_agree}. \\ &\quad (\mathit{failure}(\_, \_).\mathit{Initiator} + \mathit{inform\text{-}t/f}(\_, g_2).\mathit{ok}.\mathit{Initiator}) \end{aligned}$$

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 this FIPA protocol, consider a very simple scenario with an agent *Web_App* of type Initiator and an agent *MySQL_Server* of type Participant. Hence, we have the two agents:

$$\langle \mathit{Web\_App}, \mathit{Initiator} \rangle \text{ and } \langle \mathit{MySQL\_Server}, \mathit{Participant} \rangle.$$

Example 4 (*Specification of the FIPA contract net interaction protocol*) We assume a task offered by the participant agents belonging to a set J. Hence, the Initiator model is as follows:

$$\begin{aligned}
Initiator(x, f_{req}) &:= \overline{call}(J, \top).Init_Wait(x, f_{req}) \\
Init_Wait(x, f_{req}) &:= propose(y, f_{pre}).(\\
&\quad [acc_y]verify.\overline{accept}(y, \top).Init_Exec(x, f_{req}) + \\
&\quad [\neg \mathbf{K}_x acc_y]verify.\overline{reject}(y, \top).Init_Wait(x, f_{req})) + \\
&\quad [\psi_t]timeout.Initiator(x, f_{req}) \\
Init_Exec(x, f_{req}) &:= fail(_, _).Init_Wait(x, f_{req}) + result(_, g).ok.Initiator(x, f_{req})
\end{aligned}$$

Process *Initiator* is parameterized by the identity x of the agent and by the formula f_{req} expressing the task requirements. The call for proposals is modeled 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 (modeled by a formula ψ_t guarding action *timeout*). A proposal from a participant agent y comes with task preconditions modeled 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), the proposal can be accepted (which is modeled by proposition acc_y). In such a case, the guard $[acc_y]$ enables the branch leading to acceptance. Contrariwise, if it is not possible to derive acc_y, i.e., $\neg \mathbf{K}_x acc_y$, 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:

$$\begin{aligned}
Participant(f_{pre}) &:= call(x, _).(\\
&\quad \overline{propose}(x, f_{pre}).Part_Wait(f_{pre}) + \\
&\quad [\psi_t]timeout.Participant(f_{pre})) \\
Part_Wait(f_{pre}) &:= reject(_, _).Participant(f_{pre}) + \\
&\quad accept(x, _).(\\
&\quad \overline{fail}(x, _).Participant(f_{pre}) + \\
&\quad \overline{result}(x, \psi_r).Participant(f_{pre}))
\end{aligned}$$

A participant agent is parameterized by a formula f_{pre} modeling 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 agents of type *Participant*, with preconditions p_1 and p_2, respectively. Moreover, we have one *Initiator* agent C, with task requirements modeled by the formula $\neg r_1 \wedge r_2$. Therefore, we have the set of agents:

$$T := \{\langle C, Initiator(C, \neg r_1 \wedge r_2)\rangle,\ \langle S_1, Participant(p_1)\rangle,\ \langle S_2, Participant(p_2)\rangle\}.$$

3.3 Modeling concurrent agents

To model concurrent systems, agents must be able to communicate with each other to form a network of interacting agents. Moreover, while so far, we described agents only in terms of their local behavior, 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 behavior of several agents by integrating the notion of knowledge, which will allow us to specify how they can interact.

Definition 20 (*Pool of agents*) Let $I \subseteq Ag$ be a finite set of cardinality n. A *pool of agents* is a tuple $\mathcal{P} = \langle \{\mathcal{I}_i\}_{i\in I}, \{R_i\}_{i\in I}, X\rangle$, where $\{\mathcal{I}_i\}_{i\in I}$ is a set of agent pairs, $\{R_i\}_{i\in I}$ is a set 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 behavior of the set of agents forming the pool depends on the local behavior of each $\mathcal{I}_i = \langle i, P_i\rangle$. Each equivalence relation R_i represents the knowledge structure governing the capability of agent i to distinguish the possible worlds based on the values that can be attributed to the propositions of At. Finally, X denotes the set of atomic propositions that are taken to be true with respect to $\{\mathcal{I}_i\}_{i\in I}$.

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 (i) be conditioned by the knowledge of some formula and (ii) imply some change in the truth assignments and knowledge structures.

Formally, such a joint knowledge-based and action-based behavior is represented by a model over a KLTS describing the evolution of the pool of agents.

Definition 21 (*Semantics for pools of agents*) Let $\mathcal{P} = \langle \{\mathcal{I}_i\}_{i\in I}, \{R_i\}_{i\in I}, X\rangle$ be a pool of agents of cardinality n. The semantics of $\mathcal{P}$ is given by a model $\mathfrak{M} = \langle \mathfrak{S}, v\rangle$ over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, I, r\rangle$ with underlying LTS $\mathfrak{T} = \langle S, Act, T, s_0\rangle$ such that:

- the states in S represent pools of agents, where $\mathcal{P}$ defines the initial state s_0, such that $v(s_0) = X$ and 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 2;
- for each $s \in S$ represented by a tuple of the form $\langle \{\mathcal{I}'_i\}_{i\in I}, \{R'_i\}_{i\in I}, X'\rangle$, we define $v(s) = X'$ and for each $i \in I$, $r_s(i) = R'_i$.

Intuitively, the pool tuple $\mathcal{P}$ defines the initial state of a KLTS model that is built by applying the rules of Table 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 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 premise 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 premise of the rule. In particular, the knowledge of ψ by j is determined with respect to the Kripke model associated with

Table 2 Semantics rules for a pool of agents

Let:

- $\mathfrak{M}_s = \langle\langle \wp(At), I, \{R_i\}_{i\in I}\rangle, \mathrm{id}\rangle$ for $s = \langle \_, \{R_i\}_{i\in I}, \_\rangle$ denoting a pool of agents $\mathcal{P}$
- $diff(\mathfrak{M}_s, X, Y, \psi) = (\mathfrak{M}_s, X \Vdash_{\mathsf{S5}_n} \psi \wedge \mathfrak{M}_s, Y \nVdash_{\mathsf{S5}_n} \psi) \vee (\mathfrak{M}_s, X \nVdash_{\mathsf{S5}_n} \psi \wedge \mathfrak{M}_s, Y \Vdash_{\mathsf{S5}_n} \psi)$
- $closure(R_k) = R_k \cup \{\langle X, Y\rangle \mid \exists Z.\, \langle X, Z\rangle \in R_k \wedge \langle Z, Y\rangle \in R_k\}$

Rules:

$$(pool) \quad \frac{\mathfrak{M}_s, X \Vdash_{\mathsf{S5}_n} \mathbf{K}_j\psi \quad \mathcal{J} \xrightarrow{j.[\psi]b} \mathcal{J}'}{\mathcal{P} \xrightarrow{j.b} \mathcal{P}'}$$

where:

- $\mathcal{P} = \langle\{\mathcal{J}\} \cup \{\mathcal{I}_i\}_{i\in I\setminus\{j\}}, \{R_i\}_{i\in I}, X\rangle$
- $\mathcal{P}' = \langle\{\mathcal{J}'\} \cup \{\mathcal{I}_i\}_{i\in I\setminus\{j\}}, \{R_i\}_{i\in I}, X\rangle$

$$(set) \quad \frac{\mathcal{J} \xrightarrow{j.set(p,w)} \mathcal{J}'}{\mathcal{P} \xrightarrow{\tau} \mathcal{P}'}$$

where:

- $\mathcal{P} = \langle\{\mathcal{J}\} \cup \{\mathcal{I}_i\}_{i\in I\setminus\{j\}}, \{R_j\} \cup \{R_i\}_{i\in I\setminus\{j\}}, X\rangle$
- $\mathcal{P}' = \langle\{\mathcal{J}'\} \cup \{\mathcal{I}_i\}_{i\in I\setminus\{j\}}, \{R'_j\} \cup \{R'_i\}_{i\in I\setminus\{j\}}, X'\rangle$
- $R'_j = R_j \setminus \{\langle Y, Y'\rangle \mid diff(\mathfrak{M}_s, Y, Y', p)\}$
- $R'_i = closure(R_i \cup \{\langle\{p\} \cup Y, Y\rangle \mid p \notin Y\} \cup \{\langle Y, \{p\} \cup Y\rangle \mid p \notin Y\})$
- $X' = \begin{cases} X\setminus\{p\} & \text{if } w = 0 \\ X \cup \{p\} & \text{if } w = 1 \end{cases}$

$$(com) \quad \frac{\mathfrak{M}_s, X \Vdash_{\mathsf{S5}_n} \mathbf{K}_j\psi \quad \mathcal{I} \xrightarrow{j.\bar{a}(J,\psi)} \mathcal{I}' \quad \mathcal{J}_{i_1} \xrightarrow{i_1.a(j,\psi)} \mathcal{J}'_{i_1} \;\ldots\; \mathcal{J}_{i_h} \xrightarrow{i_h.a(j,\psi)} \mathcal{J}'_{i_h} \quad j \notin J}{\mathcal{P} \xrightarrow{\tau} \mathcal{P}'}$$

where:

- $\mathcal{P} = \langle\{\mathcal{I}, \mathcal{J}_{i_1}, \ldots, \mathcal{J}_{i_h}\} \cup \{\mathcal{I}_k\}_{k\in I\setminus(\{j\}\cup J)}, \{R_{i_1}, \ldots, R_{i_h}\} \cup \{R_k\}_{k\in I\setminus J}, X\rangle$
- $\mathcal{P}' = \langle\{\mathcal{I}', \mathcal{J}'_{i_1}, \ldots, \mathcal{J}'_{i_h}\} \cup \{\mathcal{I}_k\}_{k\in I\setminus(\{j\}\cup J)}, \{R'_{i_1}, \ldots, R'_{i_h}\} \cup \{R_k\}_{k\in I\setminus J}, X\rangle$
- $J = \{i_1, \ldots, i_h\} \subseteq I$
- $\forall i \in J.R'_i = R_i \setminus \{(Y, Y') \mid diff(\mathfrak{M}_s, Y, Y', \psi)\}$

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 $i \in I\setminus\{j\}$). 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 in 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 emphasized 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 in 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 (Aldini, 2018) – see the premises 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 premise 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 synchronization 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 *Let $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ be a model for the semantics of a pool of agents. Then $\mathfrak{S}$ is image-finite.*

This result immediately follows Definition 21 and the semantics of Table 2. As anticipated, another important result is that the semantics of Table 2 preserves the indistinguishability interpretation of the accessibility relations.

Theorem 9 *Let $\mathcal{P} = \langle \{\mathcal{I}_i\}_{i \in I}, \{R_i\}_{i \in I}, X \rangle$ be a pool of agents of cardinality n, with semantics given by a model $\mathfrak{M} = \langle \mathfrak{S}, v \rangle$ over a KLTS $\mathfrak{S} = \langle \mathfrak{T}, At, I, r \rangle$ with underlying LTS $\mathfrak{T} = \langle S, Act, T, s_0 \rangle$. 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 $s_0 = \mathcal{P}$, $\{R_i\}_{i \in I}$ is a set of equivalence relations.

By following the rules of Table 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. □

As an immediate consequence of this theorem, we have the following corollary.

Corollary 1 *The semantics of a pool of agents is an* EHML*-model.*

Based on our running examples, we now describe the behaviors of the FIPA scenarios illustrated in Examples 3 and 4.

Example 5 Let us consider the query interaction protocol 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:

$$\mathcal{P}_0 = \langle \{\langle Web\_App, Initiator \rangle, \langle MySQL\_Server, Participant \rangle\}, \{R_{WA}, R_{SQL}\}, X \rangle$$

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 2 shows a portion of the model underlying the behavior 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-if*$_p$ and yields the same relations and valuation of state $\mathcal{P}_0$. Afterward, state $\mathcal{P}_1$ (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

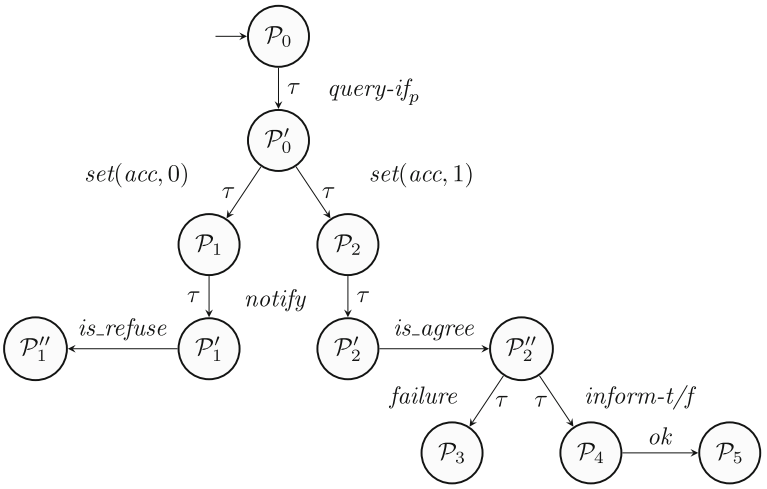

Fig. 2 Portion of the KLTS underlying the pool of agents of the query interaction protocol

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 behavior, states $\mathcal{P}_0$, $\mathcal{P}''_1$, $\mathcal{P}_3$, and $\mathcal{P}_5$ 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 that:

- 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 emphasized by the path leading to $\mathcal{P}_4$, 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.
- $\mathcal{P}_0 \not\equiv \mathcal{P}_5$, because $\mathbf{K}_{WA}(p)$ holds in the latter state but not in the former state.
- The difference among $\mathcal{P}_0$, $\mathcal{P}''_1$, and $\mathcal{P}_3$ 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 Building on the topology presented at the end of Example 4, the initial pool tuple for the contract net interaction protocol with three agents is:

$$\mathcal{P}_0 = \langle T, \{R_C, R_{S_1}, R_{S_2}\}, X\rangle.$$

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 in such a way that 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 $r_2 \wedge p_2 \rightarrow acc_2$; 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 $\mathcal{P}_0$ can be followed in Figure 4. In particular, regarding the behavior 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 synchronizes simultaneously with the two agents S_1 and S_2 (state $\mathcal{P}'_0$). Afterwards, agent C can receive the two proposals. Let us consider the case in which the proposal from S_1 arrives first (see state $\mathcal{P}_1$ 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 $\mathcal{P}'_1$ and $\mathcal{P}''_1$. Then, it is the turn of the proposal from S_2 (see state $\mathcal{P}_{12}$), 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 $\mathcal{P}'_{12}$ and $\mathcal{P}''_{12}$), with subsequent failure (state $\mathcal{P}_C$) or success of the task delivery (state $\mathcal{P}$). Going back to the race between the two proposals (see state $\mathcal{P}'_0$), let us consider the case in which agent C receives the proposal from S_2 first (see state $\mathcal{P}_2$ of the right-hand branch). For the same motivations surveyed above, the proposal is accepted (see states $\mathcal{P}'_2$ and $\mathcal{P}''_2$). At this stage, the task either is completed successfully (state $\mathcal{P}_{S_1}$), or fails (state $\mathcal{P}_{21}$). 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 $\mathcal{P}'_{21}$, $\mathcal{P}''_{21}$, and $\mathcal{P}_C$. Now, it is worth observing and comparing the three states that we have not expanded yet, i.e., $\mathcal{P}$, $\mathcal{P}_C$, and $\mathcal{P}_{S_1}$. State $\mathcal{P}$ 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 $\mathcal{P}_C$, 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 $\mathcal{P}_{S_1}$, 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 emphasize 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 $\langle\tau\rangle\langle\tau\rangle\langle verify\rangle\langle\tau\rangle\lfloor \mathbf{K}_C acc_2 \rfloor$ is satisfied in the initial state of the model, while the EHML formula $\langle\tau\rangle\langle\tau\rangle\langle verify\rangle\langle\tau\rangle\lfloor \mathbf{K}_C acc_1 \rfloor$ is false in the model, as emphasized by the alternative paths of Figure 3.

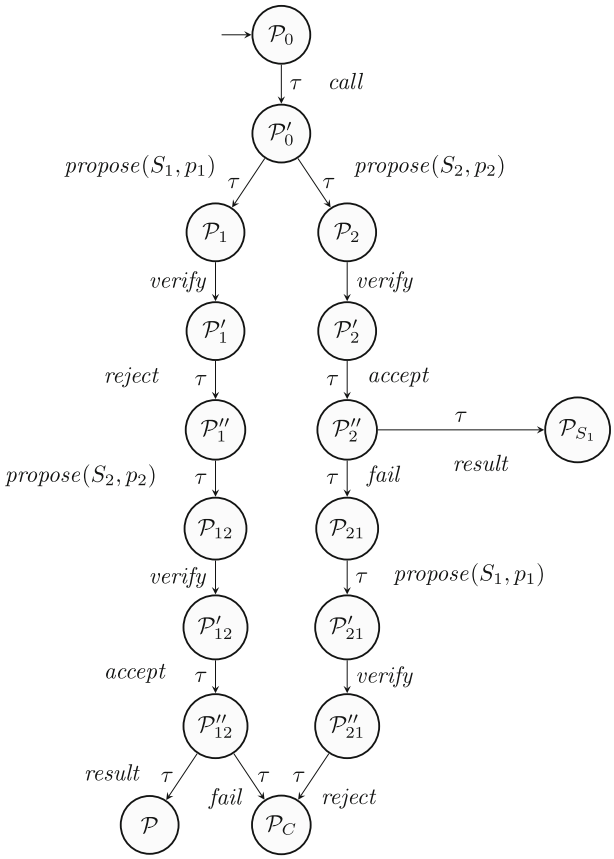

Fig. 3 Portion of the KLTS underlying the pool of agents of the contract net interaction protocol

- The EHML formula $\lfloor \neg \mathbf{K}_C \psi_r \rfloor \rightarrow \neg \langle ok \rangle \top$ 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.

4 Case study: playing Cluedo

The present case study is designed to highlight the modeling features and analysis opportunities of our framework. Despite its simplicity, this use case encompasses many of the features of real-world applications, including strategic thinking, private and public communications, and knowledge transfer. Moreover, it represents an application of the FIPA query interaction protocol.

For the sake of brevity, instead of the full Cluedo game[1] we model a simplified version. Let us consider a game set with 3 players, a dealer, and 8 cards, numbered

[1] The reader interested in reviewing the rules of the game can refer to the official instructions by Hasbro.

from 1 to 8. At the beginning of the game, the dealer samples secretly and puts aside two cards, shuffles the remaining cards together, making sure none of the cards are seen by any of the players, and then deals two cards per player. Then, the game starts and proceeds by sequential turns. On her turn, each player makes publicly a suggestion of the form: *I suggest that the two secret cards of the dealer are i and j*. There are no constraints about the specific choice of i and j. Then, if the player on the right of the one making the suggestion has at least one of the cards mentioned, she must show one of these cards secretly to her. Then, the inquiry passes to the player on the left with the same rule. At the end of her turn, the player wins the game if she has learned and can correctly declare what the dealer's cards are. Otherwise, the game proceeds with the following turns until one of the players wins.

Formally, we model the game set through the propositions p_i^j and q_i, for $0 \leq j \leq 2$ and $1 \leq i \leq 8$, where p_i^j means that player j has card i and q_i means that card i is one of the two secret cards of the dealer. The pool includes one dealer and three players and, initially, is defined as the tuple:

$$\langle \{\langle \text{Mr. Black}, \mathit{Dealer}\rangle, \langle 0, \mathit{Player}(0)\rangle, \langle 1, \mathit{Player}(1)\rangle, \langle 2, \mathit{Player}(2)\rangle\}, R, X\rangle$$

where R represents the family of accessibility relations and X the truth assignment. The three players have the same behavioral pattern, given by the process term $\mathit{Player}(\_)$, which is fed with a parameter representing the player identity. Set X is empty because the cards have yet to be shuffled by the dealer Mr. Black – hence, all the propositions are set to 0. The accessibility relation of the dealer, $R_{Mr.Black}$, contains only the reflexive pairs because, by assumption, the dealer is like an oracle and can distinguish any possible scenario. As we will see, $R_{Mr.Black}$ is immutable. The accessibility relation for each player j, denoted R_j, is such that two possible worlds are related if and only if they coincide for the values of the propositions p_i^j, $1 \leq i \leq 8$. The intuition is that, at least, a player is able to distinguish two possible worlds differing in the values of the cards she receives. All such accessibility relations are equivalence relations but are not immutable, as the knowledge of the players will change as the game proceeds.

Initially, the dealer shuffles the cards and chooses nondeterministically the two secret cards and the assignments for the players (see actions *set*):

$$\begin{aligned}
\mathit{Dealer} &:= \textstyle\sum_{k_1,k_2} \mathit{set}(q_{k_1}, 1).\mathit{set}(q_{k_2}, 1).\mathit{Deal}(k_1, k_2)\\
\mathit{Deal}(x, y) &:= \textstyle\sum_{i_1,i_2 \notin \{x,y\}} \mathit{set}(p_{i_1}^0, 1).\mathit{set}(p_{i_2}^0, 1).\overline{\mathit{deal}}(0, p_{i_1}^0 \wedge p_{i_2}^0).(\\
&\quad \textstyle\sum_{i_3,i_4 \notin \{i_1,i_2,x,y\}} \mathit{set}(p_{i_3}^1, 1).\mathit{set}(p_{i_4}^1, 1).\overline{\mathit{deal}}(1, p_{i_3}^1 \wedge p_{i_4}^1).(\\
&\quad \textstyle\sum_{i_5,i_6 \notin \{i_1,\ldots,i_4,x,y\}} \mathit{set}(p_{i_5}^2, 1).\mathit{set}(p_{i_6}^2, 1).\overline{\mathit{deal}}(2, p_{i_5}^2 \wedge p_{i_6}^2).\mathit{Play}(0)))
\end{aligned}$$

Whenever clear from the context, the bounds of a summation are not specified – in general, $\sum_{i,j}$ expresses a choice over all the possible unordered pairs of different values (i, j), each one ranging from 1 to 8. Process term *Dealer* models the random sampling of the two secret cards, and then the invocation of process term $\mathit{Deal}(k_1, k_2)$ describes the following behavior of the dealer whenever k_1 and k_2 have been chosen. The sampling for each player is modeled analogously through a pair of subsequent actions *set*. The output action *deal* is used to communicate the assignments to the

players. Then, the dealer coordinates the game rounds:

$$Play(x) := \overline{start\_turn}(x, \top).(end\_turn(\_, \_).Play((x + 1)\bmod 3) + win(\_, \_).\underline{0})$$

by assigning each turn (through the output action *start_turn*) to a different player, sequentially. Then, the dealer waits for a response: either the player turn terminates (input action *end_turn*) or the player wins the game by learning the secret pair during her turn (input action *win*).

After receiving the cards through the input action *deal*, each player is available to start her turn (input action *start_turn*) or to manage inputs from the other players. The process term $Player(x)$ is defined as follows:

$$\begin{aligned}
Player(x) := {} & deal(y, _). \\
& (start_turn(_, _). \textstyle\sum_{i_1,i_2} \overline{ask_{i_1,i_2}}((x + 1)\bmod 3, \top).show(_, _). \\
& \qquad \overline{ask_{i_1,i_2}}((x + 2)\bmod 3, \top).show(_, _). \\
& \qquad (\overline{end_turn}(y, \neg\beta_x).Player(x) + \overline{win}(y, \beta_x).\underline{0})) \\
& + \textstyle\sum_{i_1,i_2} ask_{i_1,i_2}(z, _). \\
& \qquad (\overline{show}(3 - x - z, p^x_{i_1} \vee p^x_{i_2}).(\overline{show}(z, p^x_{i_1}).Player(x) + \\
& \qquad\qquad \overline{show}(z, p^x_{i_2}).Player(x)) + \\
& \qquad \overline{show}(3 - x - z, \neg p^x_{i_1} \wedge \neg p^x_{i_2}).\overline{show}(z, \neg p^x_{i_1} \wedge \neg p^x_{i_2}).Player(x)) \\
& + show(_, _).Player(x))
\end{aligned}$$

When initiating a new turn, the player chooses nondeterministically two cards to be asked to each other player (output action *ask*) and then waits for the related answer (input action *show*). At the end of the turn, either the player learns the secret and wins the game (output action *win*) or passes the hand (output action *end_turn*). The winning condition for player x determining which output is executed is given by the knowledge of the formula $\beta_x = \bigvee_{(k,k')} \mathbf{K}_x(q_k \wedge q_{k'})$, i.e., the player knows the secret pair. Then, players respond to incoming requests through the input action *ask*. If player x receives from player z a request about cards i_1 and i_2, then we distinguish two cases. Firstly, x may have at least one of the two cards ($p^x_{i_1} \vee p^x_{i_2}$). In this case, x reveals one of the possessed cards to z, by choosing the card nondeterministically if necessary. Indirectly, even the third, silent player (identified by $3 - x - z$) learns something, i.e., the fact that x has one of the two cards. We model this indirect transfer of knowledge through an explicit output directed to player $3 - x - z$. Secondly, x may have none of the two cards ($\neg p^x_{i_1} \wedge \neg p^x_{i_2}$). In this case, the information is shared with both the other players. Finally, due to the outputs directed to player $3 - x - z$, players must also be available to learn some information during the turns of the other players (through the input action *show*).

It is worth noting that the management of the players' knowledge structures is left to the semantics of the underlying Kripke models. At the specification level, only the initial setting for these structures is modeled explicitly. This is particularly significant from the viewpoint of usability, as an analogous model based on, e.g., classical finite-state machines, would be much more challenging. To appreciate this aspect, the same use case has been modeled in the software tool NuSMV (Cimatti et

al., 2002), the specifications of which result in finite-state machines that turn out to be Kripke structures.[2] Since there are 2520 ways of dealing the 8 cards to the three players and the dealer—the computing formula is $\binom{8}{2}\cdot\binom{6}{2}\cdot\binom{4}{2}$—the NuSMV specification refers to one of these, chosen deterministically through external parameters that initialize the system configuration. Moreover, the NuSMV specification describes only a very simplistic version of the players' knowledge, in which each player does not deduce any information when observing the interactions between the other two players. In fact, the additional information needed to model the full deduction capabilities of the players should be represented explicitly by the designer and would make the model much more complicated and error-prone. By the way, despite these simplifications, the NuSMV specification is made out of about 200 code lines and 58 variables.[3]

To show an example of properties that can be model checked, we consider the derived *eventually* modality $\Diamond$, such that $\mathfrak{M}, s \Vdash \Diamond\varphi$ if and only if $\mathfrak{M}, s \Vdash \varphi$ or $\exists\pi.\, \mathfrak{M}, s \Vdash \langle\pi\rangle\Diamond\varphi$, and the derived *globally* modality $\Box$, such that $\mathfrak{M}, s \Vdash \Box\varphi$ if and only if $\mathfrak{M}, s \Vdash \varphi$ and $\exists\pi.\, \mathfrak{M}, s \Vdash \langle\pi\rangle\Box\varphi$. Then, the reachability property $\Diamond(\bigvee_x \lfloor\beta_x\rfloor)$ is satisfied, i.e., the *winning* state is reachable by some players. However, even the unreachability property $\Box(\bigwedge_x \neg\lfloor\beta_x\rfloor)$ holds. The reason is that the simple, nondeterministic strategy followed by the players when choosing their suggestion does not guarantee that the game can always be won.

5 Related work

Computational modeling and temporal logics The closest approach to our framework is proposed in Knight and Mardare (2012), 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 Dechesne and Mousavi (2007) 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-agent systems (Guzman et al., 2017; Knight et al., 2012).

In the field of concurrency theory, some of the ideas presented in this paper can be found in the study of temporal logics encompassing features from HML and modal μ-calculus (De Nicola, 1990). As an example, a variant of CTL is defined in Ter Beek et al. (2011) to check properties over expressive models called L^2TS, which are defined as an extension of a Kripke structure with a labeling 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

[2] The specification can be found on github.

[3] The underlying Kripke structure has about 2^{20} states.

to the verification of state-based propositional logic formulas and the model checking of temporal formulas.

Combining logics Within the broad framework of combining logics (Carnielli & Coniglio, 2025), EHML can be viewed as a kind of *parameterization* (in the sense of Caleiro and Sernadas (1999)) 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 temporalizing a logic introduced by Finger and Gabbay (Finger & Gabbay, 1992). In that work, starting from a propositional temporal logic TL and an arbitrary propositional logic S (with a classical base), the system TL(S) is obtained by internalizing 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. The syntax of EHML in Definition 5 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., Van Ditmarsch and Van Der Hoek (2008)). The effect of dynamic behaviors on Kripke models is represented, e.g., through a modal operator of the form $[condition]F$. In particular, this operator is used to express that F 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, F is evaluated to true in the new version of the model rather than the current one. This dynamic mechanism characterizes, e.g., the Public Announcement Logic (PAL) (Baltag & Moss, 2004) 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 G, so that $[G]F$ is true in a state of a Kripke model if, whenever G is true in the state, F is true in the state after we eliminate all not-G states and related arrows. The intuition is that G represents a public announcement that enriches the knowledge possessed by all agents, thus motivating the elimination

of all non-G possibilities from consideration. This behavior is similar to the broadcast mechanism that we implemented at the level of our process algebra.

The logic DEL (Baltag & Moss, 2016; Benthem & Gerbrandy, 2007; Bolander, 2017) 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 Renne et al. (2016) presents a framework encompassing explicitly a timekeeping relation modeling 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 (Benthem & Gerbrandy, 2007) 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 Van Otterloo (2005) mixes the epistemic operator $\mathbf{K}$ 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, 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, 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., Kooi (2011), or in a more general way, in Girard and Seligman (2012). 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 Baltag and Moss (2004), 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 modeling 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.

In general, the main novelty of our approach is to combine the low-level semantic model of KLTSs, which joins dynamic, action-based, epistemic features, with a high-level process algebraic language for the modeling of multi-agent, concurrent systems, which can be model-checked against properties expressed in a logic encompassing dynamic and epistemic ingredients.

6 Conclusions

In this paper, we have proposed a formal framework merging an LTS-based semantics for the representation of the dynamics of multi-agent systems with knowledge-based structures taken from the setting of epistemic logic.

Summarizing, by following suggestions deriving from works on dynamic and temporal epistemic logics (Parikh & Ramanujam, 2003), we associated a Kripke frame to each state of an LTS, the actions of which act as frame-transforming operations from the internal viewpoint. These transitions naturally model the evolution of the system, while the Kripke models linked to the visited states represent the way in which the knowledge of every agent evolves during the system execution. The process algebraic language that we introduced emphasizes these effects and allows for a compact and elegant description of multi-agent systems, where the details of the knowledge evolution are left to the underlying epistemic model.

Starting from this point, 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 the belief modality and the treatment of contradictions resulting from the communication between agents. Interestingly, this would open to the modeling of malicious agents sharing false information and, therefore, a theory of fake news (Aldini, 2022; Mousavi, 2018), trust, and reputation (Aldini, 2020; Aldini et al., 2021, 2024; Tagliaferri & Aldini, 2022). 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 (Van Ditmarsch & Kooi, 2006). Several studies investigate the syntactic form of potential unsuccessful formulas, in particular in the setting of public announcements for multi-agent systems (Saraf, 2012). Obviously, even in our framework, such forms can be recognized and their formal investigation will be the subject of future work.

We further aim 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* (Goguen & Burstall, 1992) (see, for instance, Neves et al. (2016)). Finally, given the high generality of the proposed framework, it would be interesting to investigate the mapping of DEL/ETL models to our models (e.g., along the same line of Benthem and Gerbrandy (2007)) as well as the relation with other abstract models, such as coalgebraic modal logics (Kupke & Pattinson, 2011), to better guide the comparison with the literature.

Data Availability No public or private datasets have been used during the current study. The files discussed in Section 4 about the NuSMV representation of the case study are publicly available at https://github.com/aldinia/cluedo.

Declarations

Conflicts of Interest All authors have no conflicts of interest.

References

Aldini, A. (2024). A process algebraic framework for multi-agent dynamic epistemic systems. In: de'Liguoro, U., Palazzo, M., & Roversi, L. (eds.) *25th Italian Conference on Theoretical Computer Science (ICTCS)*, vol. 3811, pp. 269–283. CEUR-WS, Aachen, Germany. https://ceur-ws.org/Vol-3811/.

Aldini, A., & Tagliaferri, M. (2020). Logics to reason formally about trust computation and manipulation. In: Saracino, A., & Mori, P. (eds.) *Emerging Technologies for Authorization and Authentication*. LNCS, vol. 11967, pp. 1–15. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-030-39749-4_1.

Aldini, A., Curzi, G., Graziani, P., & Tagliaferri, M. (2021). Trust evidence logic. In: Vejnarová, J., Wilson, N. (eds.) *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 16th European Conference (ECSQARU). LNAI*, vol. 12897, pp. 575–589. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-030-86772-0_41.

Aldini, A., Curzi, G., Graziani, P., & Tagliaferri, M. (2024). A probabilistic modal logic for context-aware trust based on evidence. *International Journal of Approximate Reasoning, 169*, Article 109167. https://doi.org/10.1016/j.ijar.2024.109167

Aldini, A. (2018). Design and verification of trusted collective adaptive systems. *Transactions on Modeling and Computer Simulation (TOMACS), 28*(2), 1–27. https://doi.org/10.1145/3155337

Aldini, A. (2022). On the modeling and verification of the spread of fake news, algebraically. *Journal of Logic and Computation, 32*(6), 1272–1291. https://doi.org/10.1093/logcom/exac015

Alur, R., Henzinger, T. A., & Kupferman, O. (2002). Alternating-time temporal logic. *Journal of the ACM, 49*(5), 672–713. https://doi.org/10.1145/585265.585270

Baier, C., & Katoen, J.-P. (2008). *Principles of Model Checking (Representation and Mind Series)*. Cambridge, MA, USA: The MIT Press.

Balliu, M., Dam, M., & Le Guernic, G. (2011). Epistemic temporal logic for information flow security. In: *PLAS'11: Proceedings of the ACM SIGPLAN 6th Workshop on Programming Languages and Analysis for Security*. ACM, New York. https://doi.org/10.1145/2166956.2166962.

Baltag, A., Moss, L. S., & Solecki, S. (2016). The logic of public announcements, common knowledge, and private suspicions. In: Arló-Costa, H., Hendricks, V.F., Benthem, J. (eds.) *Readings in Formal Epistemology: Sourcebook*, pp. 773–812. Springer, Cham. https://doi.org/10.1007/978-3-319-20451-2_38.

Baltag, A., & Moss, L. S. (2004). Logics for epistemic programs. *Synthese, 139*, 165–224. https://doi.org/10.1023/B:SYNT.0000024912.56773.5e

Bavendiek, K., & Schupp, S. (2022). A process calculus for privacy-preserving protocols in location-based service systems. *Journal of Logical and Algebraic Methods in Programming, 125*, Article 100735. https://doi.org/10.1016/j.jlamp.2021.100735

Benthem, J., Gerbrandy, J., & Pacuit, E. (2007). Merging frameworks for interaction: DEL and ETL. In: *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'07)*, pp. 72–81. ACM, New York.

Blackburn, P., De Rijke, M., & Venema, Y. (2002). Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge, UK. https://doi.org/10.1017/CBO9781107050884

Bolander, T. (2017). A gentle introduction to epistemic planning: The DEL approach. *Electronic Proceedings in Theoretical Computer Science, 243*, 1–22. https://doi.org/10.4204/EPTCS.243.1

Caleiro, C., Sernadas, C., & Sernada, A. (1999). Parameterisation of Logics. In: Fiadeiro, J.L. (ed.) *Recent Trends in Algebraic Development Techniques. 13th International Workshop, WADT'98, 1998, Selected Papers*, pp. 48–63. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48483-3_4.

Carnielli, W., & Coniglio, M. E. (2025). Combining Logics. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford Encyclopedia of Philosophy.* Metaphysics Research Lab: Stanford University.

Chadha, R., Delaune, S., & Kremer, S. (2009). Epistemic Logic for the Applied Pi Calculus. In: Lee, D., Lopes, A., Poetzsch-Heffter, A. (eds.) *Formal Techniques for Distributed Systems. Joint 11th IFIP WG 6.1 International Conference FMOODS and 29th IFIP WG 6.1 International Conference FORTE*, pp. 182–197. Springer, Berlin, Heidelberg. doi: https://doi.org/10.1007/978-3-642-02138-1_12.

Chatterjee, K., Henzinger, T. A., & Piterman, N. (2010). Strategy logic. *Information and Computation, 208*(6), 677–693. https://doi.org/10.1016/j.ic.2009.07.004

Cimatti, A., Clarke, E. M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., & Tacchella, A. (2002). NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In: *Computer Aided Verification. 14th International Conference, CAV 2002*, pp. 359–364. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45657-0_29.

Clarke, E. M., & Emerson, E. A. (1982). Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) *Logics of Programs,* pp. 52–71. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0025774.

Coenen, N., Finkbeiner, B., Hahn, C., & Hofmann, J. (2019) The Hierarchy of Hyperlogics. In: *Proc. of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'19),* Article No.: 39, pp. 1–13. IEEE, Vancouver. doi: https://doi.org/10.1109/LICS.2019.8785713.

De Nicola, R., & Vaandrager, F. (1990). Action versus state based logics for transition systems. In: Guessarian, I. (ed.) *Semantics of Systems of Concurrent Processes. LITP Spring School on Theoretical Computer Science 1990, Proceedings. LNCS*, vol. 469, pp. 407–419. Springer, Berlin, Heidelberg. doi: https://doi.org/10.1007/3-540-53479-2_17.

Dechesne, F., Mousavi, M. R., & Orzan, S. (2007). Operational and Epistemic Approaches to Protocol Analysis: Bridging the Gap. In: *Dershowitz, N., Voronkov, A. (eds.) Logic for Programming, Artificial Intelligence, and Reasoning. 14th International Conference, LPAR. LNAI*, vol. 4790, pp. 226–241. Springer, Berlin, Heidelberg. doi: https://doi.org/10.1007/978-3-540-75560-9_18.

Dechesne, F., & Wang, Y. (2010). To know or not to know: epistemic approaches to security protocol verification. *Synthese, 177*, 51–76. https://doi.org/10.1007/s11229-010-9765-8

Ditmarsch, H., Halpern, J. Y., Hoek, W., & Kooi, B. (Eds.). (2015). *Handbook of Epistemic Logic*. College Publications, Rickmansworth, UK.

Emerson, E. A., & Halpern, J. Y. (1983). "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*, pp. 127–140. ACM, New York, NY, USA. doi: https://doi.org/10.1145/567067.567081.

Finger, M., & Gabbay, D. M. (1992). Adding a Temporal Dimension to a Logic System. *Journal of Logic, Language, and Information, 1*(3), 203–233. https://doi.org/10.1007/BF00156915

Fokkink, W. (2007). Introduction to Process Algebra. *Springer, Berlin, Heidelberg.* https://doi.org/10.1007/978-3-662-04293-9

Girard, P., Seligman, J., & Liu, F. (2012). General dynamic dynamic logic. In: Bolander, T., Braüner, T., Ghilardi, S., & Moss, L. (eds.) *9th Conf. on Advances in Modal Logic. Advances in Modal Logic*, vol. 9, pp. 239–260. College Publications, Rickmansworth, UK.

Goguen, J. A., & Burstall, R. M. (1992). Institutions: abstract model theory for specification and programming. *Journal of the ACM, 39*(1), 95–146. https://doi.org/10.1145/147508.147524

Gorrieri, R., & Versari, C. (2015). Introduction to Concurrency Theory - Transition Systems and CCS. *Springer, Cham.* https://doi.org/10.1007/978-3-319-21491-7

Guzman, M., Haar, S., Perchy, S., Rueda, C., & Valencia, F. D. (2017). 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. https://doi.org/10.1016/j.jlamp.2016.09.001

Halpern, J. Y., & Moses, Y. (1992). A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence, 54*(3), 319–379. https://doi.org/10.1016/0004-3702(92)90049-4

Harel, D., Kozen, D., & Tiuryn, J. (2000). *Dynamic Logic*. Cambridge, MA: MIT Press.

Hennessy, M., & Milner, R. (1980). On observing nondeterminism and concurrency. In: Bakker, J.W., & Leeuwen, J. (eds.) *Automata, Languages and Programming. Seventh Colloquium. LNCS*, vol. 85, pp. 299–309. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-10003-2_79.

Hennessy, M. (1991). A proof system for communicating processes with value-passing. *Formal Aspects of Computing, 3*, 346–366. https://doi.org/10.1007/BF01642508

Hennessy, M., & Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. *Journal of the ACM, 32*(1), 137–161. https://doi.org/10.1145/2455.2460

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM, 12*(10), 576–580. https://doi.org/10.1145/363235.363259

Hoek, W., & Wooldridge, M. (2002). Tractable multiagent planning for epistemic goals. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3. AAMAS'02*, pp. 1167–1174. ACM, New York, NY, USA. https://doi.org/10.1145/545056.545095.

Huang, S., Cao, Y., Wang, H., & Qu, W. (2012). Value-passing CCS with noisy channels. *Theoretical Computer Science, 433*, 43–59. https://doi.org/10.1016/j.tcs.2012.03.002

Knight, S., Mardare, R., & Panangaden, P. (2012). Combining Epistemic Logic and Hennessy-Milner Logic. In: Constable, R.L., & Silva, A. (eds.) *Logic and Program Semantics. Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday. LNCS,* vol. 7230, pp. 219–243. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29485-3_14.

Knight, S., Palamidessi, C., Panangaden, P., & Valencia, F. D. (2012). Spatial and epistemic modalities in constraint-based process calculi. In: Koutny, M.,& Ulidowski, I. (eds.) *CONCUR - Concurrency Theory, pp. 317–332. Springer, Berlin, Heidelberg*. https://doi.org/10.1007/978-3-642-32940-1_23.

Kooi, B., & Renne, B. (2011). Generalized arrow update logic. In: *13th Conf. on Theoretical Aspects of Rationality and Knowledge (TARK'11),* pp. 205–211. ACM, New York, NY, USA. https://doi.org/10.1145/2000378.2000403.

Kozen, D. (1983). Results on the propositional μ-calculus. *Theoretical Computer Science, 27*(3), 333–354. https://doi.org/10.1016/0304-3975(82)90125-6

Kupke, C., & Pattinson, D. (2011). Coalgebraic semantics of modal logics: An overview. *Theoretical Computer Science, 412*(38), 5070–5094. https://doi.org/10.1016/j.tcs.2011.04.023

Minami, K. (2020). Trace Equivalence and Epistemic Logic to Express Security Properties. In: Gotsman, A., Sokolova, A. (eds.) Formal Techniques for Distributed Objects, Components, and Systems. 40th IFIP WG 6.1 International Conference, FORTE, pp. 115–132. Springer, Cham. https://doi.org/10.1007/978-3-030-50086-3_7.

Mousavi, M. R., & Varshosaz, M. (2018). Telling lies in process algebra. In: *2018 Symposium on Theoretical Aspects of Software Engineering (TASE),* pp. 116–123. IEEE, New York, NY, USA. https://doi.org/10.1109/TASE.2018.00023.

Neves, R., Madeira, A., Martins, M. A., & Barbosa, L. S. (2016). Proof theory for hybrid(ised) logics. *Science of Computer Programming, 126*, 73–93. https://doi.org/10.1016/j.scico.2016.03.001

Parikh, R., & Ramanujam, R. (2003). A knowledge based semantics of messages. *Journal of Logic, Language and Information, 12*(4), 453–467. https://doi.org/10.1023/A:1025007018583

Pnueli, A. (1977). The temporal logic of programs. In: *Proc. of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977),* pp. 46–57. https://doi.org/10.1109/SFCS.1977.32.

Poslad, S., & Charlton, P. (2001). 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. Selected Tutorial Papers. LNCS*, vol. 2086, pp. 98–117. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-47745-4_5.

Poslad, S. (2007). Specifying protocols for multi-agent systems interaction. *ACM Transactions on Autonomous and Adaptive Systems, 2*(4), 15. https://doi.org/10.1145/1293731.1293735

Pratt, V. R. (1976) Semantical Considerations On Floyd-Hoare Logic. In: 17th Annual Symposium on Foundations of Computer Science (SFCS 1976), pp. 109–121. https://doi.org/10.1109/SFCS.1976.27.

Renne, B., Sack, J., & Yap, A. (2016). Logics of temporal-epistemic actions. *Synthese, 193*(3), 813–849. https://doi.org/10.1007/s11229-015-0773-6

Rybakov, V. (2009). Linear Temporal Logic $\mathcal{LTLK}$ extended by Multi-Agent Logic K_n with Interacting Agents. *Journal of Logic and Computation, 19*(6), 989–1017. https://doi.org/10.1093/logcom/exp027

Saraf, S., & Sourabh, S. (2012). Characterizing successful formulas: the multi-agent case. CoRR arXiv:1209.0935.

Tagliaferri, M., & Aldini, A. (2022). From belief to trust: A quantitative framework based on modal logic. *Journal of Logic and Computation, 32*(6), 1017–1047. https://doi.org/10.1093/logcom/exac016

Ter Beek, M. H., Fantechi, A., Gnesi, S., & Mazzanti, F. (2011). A state/event-based model-checking approach for the analysis of abstract system properties. *Science of Computer Programming, 76*(2), 119–135. https://doi.org/10.1016/j.scico.2010.07.002

van Ditmarsch, H., van der Hoek, W., & Kooi, B. (2008). Dynamic Epistemic Logic. Syntese Library, vol. 337. Springer, Dordrecht. https://doi.org/10.1007/978-1-4020-5839-4.

Van Ditmarsch, H., & Kooi, B. (2006). The secret of my success. *Synthese, 151*, 201–232. https://doi.org/10.1007/s11229-005-3384-9

van Otterloo, S., & Jonker, G. (2005). On epistemic temporal strategic logic. Electronic Notes in Theoretical Computer Science, 126, 77–92. 2nd Workshop on Logic and Communication in Multi-Agent Systems (2004). https://doi.org/10.1016/j.entcs.2004.11.014.