Journal of Logic and Computation Advance Access published August 24, 2010

Non-axiomatizability for the linear temporal logic of knowledge with concrete observability

CĂTĂLIN DIMA, *LACL, Université Paris-Est Créteil Val de Marne, 61 av. du Général de Gaulle, 94010 Créteil, France.*
E-mail: dima@univ-paris12.fr

Abstract

We show that propositional linear temporal logic with knowledge modalities but without common knowledge has an undecidable satisfiability problem when interpreted in a 'concrete' semantics with perfect recall or with perfect recall and synchrony. We then conclude that this concrete semantics is not axiomatizable in the semantics, based on local states.

Keywords: Temporal epistemic logics, undecidability.

1 Introduction

Combinations of temporal and epistemic logics have been studied since the mid-80s, starting with [9, 10], identifying 96 different logics, distinguished by semantics and/or the presence of common knowledge operators, and presenting decidability and undecidability results for the satisfiability problem in the presented logics. Such logics have recently proved useful in the formal verification applied to various distributed systems in which the knowledge of the participants is essential for the correctness of the system specification. Examples include the verification of confidentiality [27, 28], authentication [3, 16, 19], mutual agreement [17], various types of anonymity [7, 11, 17, 26] or privacy [1].

The semantics of temporal epistemic logics is based on an observability relation for each agent, expressing the observation capabilities of the agents on which the agenst based their abilities to make deductions on the global system state, and on the other agents' knowledge of it. It is common to consider that observability relations are given by so-called *local states* of each agent, which gives what is called the *interpreted systems* semantics [5]. In this framework, a global state is a tuple of local states, then two global states are identically observable for an agent if the local state of the agent in both global states is the same. Also, in interpreted systems, system evolutions are presented as runs in a transition system—that is, sequences of global states connected by system transitions.

In its most general presentation, e.g. [6, 9], this semantics does not impose any restriction on the valuation of atomic propositions in states that are identically observable for some agent A. More specifically, two global states s, s' might be declared as identically observable for agent A, yet the atomic propositions which hold in s could be disjoint from the atomic propositions which hold in s'. However, in many of the above-mentioned applications of temporal epistemic logics, the observability relations are intimately related with the truth values for atomic propositions. For instance, in [14, 16, 20, 26], the local states for each agent incorporate items describing what agents *sent/received* and what they *possess* during the protocol behaviour, and there are atomic formulas encoding exactly the possession, the reception or the issue of items during the protocol run. It is therefore quite natural to

doi:10.1093/logcom/exq031

consider a 'concrete' observability relation, in which local states for an agent A are characterized by truth values for some fixed subset of atomic propositions Π_A.

We present here such a 'concrete' semantics for temporal epistemic logics, and investigate its relationship with the semantics based on interpreted systems. We focus here on the perfect recall case and the perfect recall and synchronous; formally, two system runs are identically observable by an agent A at some instant i (w.r.t. the synchronous and perfect recall semantics) *if and only if* the sequence, upto instant i, of truth values for atoms in Π_A (with or without stuttering) is the same in both histories. Our logic contains only 'individual knowledge' operators, and no common knowledge, and corresponds to the logic KL_n in [9, 10], also denoted $PTL+S5(n)$ [6], interpreted over the classes of models denoted $\mathcal{C}_{nf}$, resp. $\mathcal{C}_{nf,sync}$, but with a concrete version of the observability relations.

Such a concrete observability relation could be related with the alternative semantics for logics of knowledge and time, known as Kripke semantics. In this semantics, the observability relations are not induced by local states, but are directly defined on global states in the transition system. We may then associate with each equivalence class for the observability relation for some agent A a new atomic proposition which holds exactly in the states of that equivalence class, and does not hold anywhere else.

At a first sight, it may look that the semantics is just a particular case of the semantics based on interpreted systems: on one hand, in the interpreted systems semantics, one may include the local states in the set of atomic propositions and declare the set of atomic propositions corresponding to the local states for agent A as the set of observable atoms for A. For the reverse, one might try to axiomatize the correspondence between local states and validity of Π_A, by considering axioms $\vdash p \rightarrow K_A p$ and $\vdash \neg p \rightarrow K_A \neg p$ for each atomic proposition $p \in \Pi_A$.

Unfortunately, this set of axioms does not completely characterize observability for agent A by means of observability of truth values for Π_A. In fact, as we prove in this article, the concrete semantics is not axiomatizable. We prove this result for both the perfect recall semantics, and for the perfect recall and synchronous case. The reason for this result is that there is no possibility to impose, axiomatically, the identical observability of two histories, on the basis of the observability of truth values for some given subset of atomic propositions. To compare with the classical framework of interpreted systems, note that [8] gives a complete axiom system for both cases, and their satisfiability problem is shown decidable in [9].

There has been some interest in relating the interpreted systems semantics with general Kripke semantics. In [15], for special types of interpreted systems, called *hypercube systems*, the two semantics are shown to be equivalent, and the authors suggest that 'further research could be undertaken […] to have a general methodology for translating interesting classes of interpreted systems into classes of Kripke models'.

These considerations give a first reason why semantics based on concrete observability is worth studying. But there is also a second reason defending this research, related with the possibility to compare the expressive power of temporal epistemic logics with the expressive power of fragments of monadic logics over tree structures with some auxiliary interpreted predicates. As known, epistemic temporal logics are interpreted over transition systems endowed with some additionnal relations. If we have in mind that unfoldings of transition systems are infinite trees, and that MSO, the Monadic Second-Order logic of trees [22], accounts as the reference logic for specifying system behaviours as it is as expressive as the mu-calculus, tree automata or quantified propositional temporal logics, one may ask the question how combinations of temporal and epistemic logics compare, in expressivity, with MSO. The concrete observability semantics gives a natural setting for exploring this question, because the propositional symbols can be interpreted as second-order monadic variables in MSO (i.e. as sets of positions in a tree). Expressing synchrony and perfect recall would then be not very

difficult with the aid of the *equal-level predicate* [21]. In the abstract semantics, translating formulas with temporal and (individual) knowledge operators into MSO with equal-level predicate requires one to encode the observability classes into new propositional variables—which, in fact, accounts to using the concrete observability. The exploration of the the relationship between temporal logics of knowledge and extensions of MSO is not the subject of this article, but it is the main reason for studying this 'concrete' semantics and its relation with the interpreted systems semantics.

Shilov and Garanina [18] have shown how to encode a particular case of the model-checking problem for an epistemic variant of computational tree logic (CTL) into the Chain Logic with equal-level predicate (Chain Logic is a proper subset of MSO, see [21]). Their results do not apply to the full class of model-checking problems, as their translation is dependent on the length of the runs at which the epistemic CTL formulas are to be interpreted. And hence, their technique does not provide a model-independent translation of epistemic CTL formulas into formulas of chain logic with equal-level predicate.

Note also that some of the undecidability results for temporal epistemic logics have been related with undecidability results for temporal logics on bidimensional structures, observing that especially the presence of the common knowledge operator makes it possible to encode grid-like structures [23]. But the connection with MSO logics and the reasons for undecidability that could stem from these connections has never been discussed.

Our non-axiomatizability result is based on a proof of the undecidability of the satisfiability problem for the linear temporal logic of knowledge with both the perfect recall semantics and the perfect recall and synchronous semantics. We actually prove that we may associate to each deterministic Turing machine T which starts with a blank tape, a formula ϕ_T, such that ϕ_T is satisfiable if and only if T has a configuration which is visited infinitely often. The possibility to associate formulas ϕ_T also for Turing machines T which visit all cell tapes gives us the means to show that there exists no recursively enumerable axiomatization for our concrete versions of the semantics of KL_n.

The proof of the undecidability of KL_n satisfiability works by coding the configurations and transitions in the computation of a deterministic Turing machine as runs in a multi-agent system. This proof idea was utilized many times in the literature for proving undecidability of various epistemic temporal logics, starting from [9] where it is proved that CKL_n, which is linear temporal logic (LTL) *with common knowledge operators*, has an undecidable satisfiability problem. But recall that KL_n interpreted over $\mathcal{C}_{nf}$, resp. $\mathcal{C}_{nf,sync}$, is proved to have a decidable satisfiability problem in [9]. We also cite the undecidability result of [24] for LTL *with common knowledge too*, which utilizes the same type of argument. Another paper which utilizes this argument is [23], in which it is shown that several variants of branching-time logics *with common knowledge operators* have an undecidable satisfiability problem. Contrary to these results, our undecidability result holds *without a common knowledge operator*. To complete the figure, recall that the only undecidable cases of temporal logics without common knowledge studied in [9] concern only *non-learning* variants of observability, based on previous results from [13]. But the technique used in this only case without common knowledge heavily relies on asynchrony, and on the non-learning character of the semantics. Our proof uses a different technique, which relies on synchrony and the possibility to manipulate, in the logic, formulas which exactly identify observations (by means of the atomic propositions which are observable for each agent).

One of the sources of inspiration for the concrete semantics is the logic of local propositions from [4]. The atomic propositions in the sets Π_A are interpreted as local propositions for agent A, in the sense of [4]. But, contrary to [4], we do not have quantifications over atomic propositions, and, as such, our logic is strictly less expressive on its pure epistemic part.

Let us finally mention that the undecidability and non-axiomatizability results for the perfect recall (non-synchronous) case reduces to the synchronous case. This holds because, with concrete observability, one may always turn a non-synchronous model into a synchronous one by inserting a 'clock' atom which is observable by everybody. This does not reduce the expressivity of the non-synchronous perfect recall semantics: the model-checking problem for the non-synchronous perfect recall semantics does not reduce to the same problem for the synchronous case.

The article is organized as follows: the next section gives the syntax and semantics of KL_n. We also show how to model the dining cryptographers protocol in our concrete semantics, arguing for the naturality of this type of semantics. We then present the undecidability and the non-axiomatizability result in the third section. We end with a section of conclusions and comments.

2 Syntax and semantics of KL_n

In this section, we recall the syntax and the semantics of KL_n [6, 9] based on interpreted systems. Then, we introduce the 'concrete' semantics, and show that the semantics based on interpreted systems can be expressed in the concrete semantics. We start by giving some of the basic notions and notations used in this article.

2.1 Preliminaries

A *transition system* is a tuple $\mathcal{T}=(S,\rightarrow,S_0)$, where S is a set of *states*, $S_0\subseteq S$ a set of *initial states* and $\rightarrow\subseteq S\times S$ is a set of *transitions*. We denote, as usually, $s\rightarrow s'$ instead of $(s,s')\in\rightarrow$.

Runs of a transition system $\mathcal{T}$ are finite or infinite sequences of states in S, starting with an initial state and connected by the transition relation:

$$\mathsf{Runs}(\mathcal{T})=\left\{(s_i)_{0\leq i<\eta}\,|\,\eta\in\mathbb{N}\cup\{\infty\},\forall i<\eta, s_{i-1}\rightarrow s_i \text{ and } s_0\in S_0\right\}.$$

The item η is called the *length* of the run. Note that this defines both finite runs (when $\eta\in\mathbb{N}$) and infinite runs (when $\eta=\infty$). The set of finite runs is denoted $\mathsf{FinRuns}(\mathcal{T})$, and the set of infinite runs is denoted $\omega\mathsf{Runs}(\mathcal{T})$. The i-th state in the run ρ is denoted $\rho[i]$. Also, given a run $\rho=(s_i)_{0\leq i<k}$ of length k, and some $k'\leq k$, the *prefix of length* k' of ρ is the run denoted $\rho[0..k']=(s_i)_{0\leq i<k'}$.

Given a transition system $\mathcal{T}=(S,\rightarrow,S_0)$ and a surjective mapping $f:S\rightarrow S'$, the set S' can be endowed with a transition system structure as usual, $\mathcal{T}'=(S',\rightarrow_f,S'_0)$ with $S'_0=f(S_0)$ and $u\rightarrow_f u'$ if there exists $s,s'\in S$ with $f(s)=u,f(s')=u$ and $s\rightarrow s'$. We then say that $\mathcal{T}'$ is the *projection* of $\mathcal{T}$ by f. In the same setting, given a run $\rho=\left(s_i\right)_{1\leq i<\eta}\in\mathsf{Runs}(\mathcal{T})$, the *projection* of ρ is the following run of $\mathcal{T}'$: $f(\rho)=\left(f(s_i)\right)_{1\leq i<\eta}$.

In this article, we also have situations in which $\mathcal{T}$ and $\mathcal{T}'$ are such that $S=2^A$ and $U=2^B$ for some set A and $B\subseteq A$, and the projection is defined by $f(X)=X\cap B$. In these situations, the projection f is also denoted $\cdot\big|_B$, notation which is also employed for the projection of a run ρ by f: we denote $\rho\big|_B$ instead of $f(\rho)$.

Given a run $\rho=\left(s_i\right)_{1\leq i<\eta}\subseteq\mathsf{Runs}(\mathcal{T})$, the *removal of stuttering steps from* ρ, denoted $\mathsf{stut}(\rho)$, is the sequence of states s_{i_j} resulting by removing any successive states that are identical in ρ. For example, for $\rho=(s_1s_2s_2s_2s_3s_2s_3s_1s_1s_1)$, then $\mathsf{stut}(\rho)=(s_1s_2s_3s_2s_3s_1)$. Note that $\mathsf{stut}(\rho)$ is also a run of $\mathcal{T}$.

2.2 KL_n syntax

The syntax of KL_n is given by the following grammar:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid \bigcirc\phi \mid \phi \mathcal{U} \phi \mid K_A\phi,$$

where $p \in \Pi$, with Π denoting the set of *atomic propositions*, while $A \in Ag$, with Ag denoting a (finite) set of *agents*. The formula $K_A\phi$ reads 'Agent A knows ϕ'. The set of KL_n formulas over Π is denoted $Form(\Pi)$.

The usual derived operators are listed in the sequel:

$$\Diamond\phi = \mathsf{true}\mathcal{U}\phi \qquad \Box\phi = \neg\Diamond\neg\phi \qquad P_A\phi = \neg K_A\neg\phi.$$

2.3 Interpreted systems semantics for KL_n

The semantics based on interpreted systems for KL_n is based on an 'abstract' notion of state. Several variants of this semantics can be given, based on the possibility of the agents to recall the history of their observations and/or their access to a global clock. We present here only the *perfect recall* semantics, on one side, and the *synchronous and perfect recall* semantics, on the other side.

An Ag-agent transition system (or a multi-agent system, when Ag is understood from the context) is a tuple $\mathcal{M} = \left(S, (L_A)_{A\in\mathsf{Ag}}, \rightarrow, S_0, (\sim_A)_{A\in\mathsf{Ag}}, \Pi, \nu\right)$ whose components satisfy the following properties:

- Ag is a finite set of *agents*.
- L_A is the *local state*, the component of the *global state* that agent A can observe.
- S is the set of *global states*, defined as the cartesian product of local states, $S = \times_{A\in Ag} L_A$. For a global state $s = (l_A)_{A\in Ag}$, we denote $s\big|_A = l_A$, the element indexed A from the tuple s.
- The tuple $(S, \rightarrow, S_0)$ is the *underlying transition system* for $\mathcal{M}$, and hence $\rightarrow \subseteq S \times S$ and $S_0 \subseteq S$.
- $\sim_A$ is the *observability relation* for agent A, and is defined on the set of finite runs of $\mathcal{M}$, $\sim_A \subseteq \mathsf{FinRuns}(\mathcal{M}) \times \mathsf{FinRuns}(\mathcal{M})$.
- ν is the interpretation of atomic propositions, $\nu : S \rightarrow 2^{\Pi}$.

We will be interested here in multi-agent systems in which the observability relation for each agent satisfies some particular properties. The first property is *perfect recall*, and models the situation in which an agent is able to memorize *changes in local states* upto the current moment, and to tell apart two runs which do not have the same history of changes in the local state. The second property is *synchrony and perfect recall*, and models the situation in which an agent is able to tell apart two runs which do not have the same history of local states.

Formally, these two properties are the following

DEFINITION 1

(1) The **perfect recall observability** (*pr*-observability) relation for agent $A \in Ag$ is the relation defined as following:

$$\sim_A^{pr} \subseteq \mathsf{FinRuns}(\mathcal{M}) \times \mathsf{FinRuns}(\mathcal{M}), \quad \rho \sim_A^{pr} \rho' \text{ if } \mathsf{stut}(\rho\big|_{L_A}) = \mathsf{stut}(\rho'\big|_{L_A}).$$

(2) The **perfect recall and synchronous observability** (*prs*-observability) relation for agent $A \in Ag$ is the relation defined as following:

$$\sim_A^{prs} \subseteq \mathsf{FinRuns}(\mathcal{M}) \times \mathsf{FinRuns}(\mathcal{M}), \quad \rho \sim_A^{prs} \rho' \text{ if } \rho\big|_{L_A} = \rho'\big|_{L_A}.$$

Note that agents that have *prs*-observability can always distinguish two finite runs having different lengths.

The semantics of KL_n , parameterized by the type of observability relation $rel \in \{pr, prs\}$ is given by the following rules, in which $\rho \in \omega\mathsf{Runs}$ and $i \in \mathbb{N}$:

$$
\begin{array}{ll}
(\mathcal{M},\rho,i) \models_{rel} p & \text{if } p \in \rho[i] \\
(\mathcal{M},\rho,i) \models_{rel} \phi_1 \wedge \phi_2 & \text{if } (\mathcal{M},\rho,i) \models_{rel} \phi_1 \text{ and } (\mathcal{M},\rho,i) \models_{rel} \phi_2 \\
(\mathcal{M},\rho,i) \models_{rel} \neg\phi & \text{if } (\mathcal{M},\rho,i) \not\models_{rel} \phi \\
(\mathcal{M},\rho,i) \models_{rel} \bigcirc\phi & \text{if } (\mathcal{M},\rho,i+1) \models_{rel} \phi \\
(\mathcal{M},\rho,i) \models_{rel} \phi_1 \mathcal{U} \phi_2 & \text{if } \exists j \geq i \text{ with } (\mathcal{M},\rho,j) \models_{rel} \phi_2 \text{ and } \forall i \leq k < j, (\mathcal{M},\rho,i) \models_{rel} \phi_1 \\
(\mathcal{M},\rho,i) \models_{rel} K_A\phi & \text{if } \forall \rho' \in \omega\mathsf{Runs}(\mathcal{M}) \text{ and } \forall j \in \mathbb{N} \text{ with } \rho[0..i] \sim_A^{rel} \rho'[0..j] \\
 & \quad \text{we have } (\mathcal{M},\rho',j) \models_{rel} \phi
\end{array}
$$

Given a formula $\phi \in Form(\Pi)$ and a multi-agent system $\mathcal{M}$, we say that ϕ *is satisfied in* $\mathcal{M}$ *w.r.t. pr-observability* (resp. w.r.t. *prs*-observability) if there exists a run ρ and a position i in the run such that $(\mathcal{M},\rho,i) \models_{pr} \phi$ – respectively $(\mathcal{M},\rho,i) \models_{prs} \phi$. A formula is *satisfiable* if there exists a model in which it is satisfiable. A formula ϕ is *valid* if it is satisfied in any system $\mathcal{M}$.

2.4 *A concrete semantics for* KL_n

The classical semantics presented in the previous subsection does not impose any restriction on the interpretation of atomic propositions in two global states $s, s' \in S$ that may occur on distinct runs ρ, ρ', that are identically observable for agent A. For instance, it is perfectly possible to have two runs $\rho = (s_i)_{0 \leq i < k}$ and $\rho' = (s'_i)_{0 \leq i < k}$ with $\rho \sim_A^{prs} \rho'$, (hence $s_i\big|_A = s'_i\big|_A$ for all $0 \leq i < k$) and such that $\nu(s_k) \cap \nu(s'_k) = \emptyset$, that is, no atomic proposition that holds at s_k also holds at s'_k.

However, it is natural to consider that the truth value of some subset of atomic propositions is observable by an agent A, that is, to have a fixed set of atomic propositions Π_A that have the same truth value at both s_k and s'_k above. It is also natural to consider that the set of atomic propositions that are observable by agent A characterizes *uniquely* agent A's observability relation. This can be formalized as follows:

DEFINITION 2
A multi-agent system $\mathcal{M}$ **has concrete observability** for agent $A \in Ag$ if there exists a subset of atomic propositions $\Pi_A \subseteq \Pi$ such that for each $s, s' \in S$, $s\big|_A = s'\big|_A$ if and only if $\nu(s) \cap \Pi_A = \nu(s') \cap \Pi_A$.

The set Π_A is the set of atomic propositions whose truth values can be observed by the agent A *in any state of the system*. It is essential to note the equivalence, required by this definition, between the state-based observability and the observability of a specific set of atomic propositions, Π_A.

In such contexts, we may redefine the two special observability relations discussed in the previous section as follows:

DEFINITION 3
A multi-agent system in which all agents have **concrete perfect recall observability** (*cpr*-observability) is a system $\mathcal{M} = \left(S, \rightarrow, S_0, (\sim_A^{cpr}), \Pi, (\Pi_A)_{A \in Ag}, \nu\right)$ in which $\Pi_A \subseteq \Pi$ for all agents $A \in Ag$ and

$$\rho \sim_A^{cpr} \rho' \text{ if and only if } \mathsf{stut}(\nu(\rho)\big|_{\Pi_A}) = \mathsf{stut}(\nu(\rho')\big|_{\Pi_A}). \tag{1}$$

A multi-agent system in which all agens have **concrete perfect recall and synchronous observability** (*cprs*-observability) is a system $\mathcal{M}=\big(S,\rightarrow,S_0,(\sim_A^{cprs}),\Pi,(\Pi_A)_{A\in Ag},\nu\big)$ with

$$\rho\sim_A^{cprs}\rho' \text{ if and only if } \nu(\rho)\big|_{\Pi_A}=\nu(\rho')\big|_{\Pi_A}. \tag{2}$$

In other words, a multi-agent system with concrete observability is just a multi-agent system in which the observability relation on states is the equivalence relation defined by the mapping $\nu_A:S\rightarrow 2^{\Pi_A},\nu_A(q)=\nu(q)\cap\Pi_A$, in the sense that two runs ρ and ρ' are identically observable by an agent which has an observability relation which is concrete, synchronous and has perfect recall, if the set of atoms from Π_A have the same truth values along both runs.

We then have two new semantics, $\models_{cpr}$ and $\models_{cprs}$, introduced by these two observability relations along the rules listed at the end of the previous subsection.

We may show that the two (sets of) concrete observability relations $\sim_A^{cpr}$ and $\sim_A^{cprs}$ are at least as expressive as their classical variants. The idea for proving this expressivity result is to add $\bigcup_{A\in Ag}L_A$ to the set of propositional symbols, and put $\Pi_A=L_A$ for all agents $A\in\mathsf{Ag}$. Formally, given a multi-agent system $\mathcal{M}=\big(S,(L_A)_{A\in\mathsf{Ag}},\rightarrow,S_0,(\sim_A^{pr})_{A\in\mathsf{Ag}},\Pi,\nu\big)$ with *pr*-observability relations for each agent, we build the following multi-agent system with *cpr*-observability :

$$\mathcal{M}'=\big(S,\rightarrow,S_0,(\simeq_A^{cpr})_{A\in\mathsf{Ag}},\Pi',(\Pi'_A)_{A\in Ag},\nu'\big),$$

where $\Pi'=\Pi\cup\bigcup_{A\in Ag}L_A$, $\Pi'_A=L_A$ and $\nu'(s)=\nu(s)\cup\{s\big|_A\}$.

Then it is easy to observe that for any formula $\phi\in Form(\Pi)$, any run $\rho\in\omega\mathsf{Runs}(\mathcal{M})$ and any time point $i\in\mathbb{N}$,

$$(\mathcal{M},\rho,i)\models_{pr}\phi \text{ iff } (\mathcal{M}',\rho,i)\models_{cpr}\phi.$$

The main problem that we address in the rest of this article is the following:

PROBLEM 1
Are the two concrete observability relations, the *cpr*-observability and the *cprs*-observability, axiomatizable into the abstract semantics?

It may look that the answer to this question is to consider the following set of axioms

$$\vdash p\rightarrow K_Ap \qquad\qquad \vdash\neg p\rightarrow K_A\neg p$$

for each $p\in\Pi_A$, and each $A\in Ag$.

In a multi-agent system which satisfies this set of axioms, we do not necessary have that whenever $\nu(s)\big|_A=\nu(s')\big|_A$ for two states $s,s'\in S$, it also holds that $s\big|_A=s'\big|_A$. Had we have asked only the 'only if' part of the Definition 2, that is, $\forall s,s'\in S,s\big|_A=s'\big|_A\Rightarrow\nu(s)\cap\Pi_A=\nu(s')\cap\Pi_A$, then this axiomatization would be sufficient.

Section 3 is dedicated to proving that the expressivity Problem 1 has a negative answer.

2.5 An example

We show here how to model the Dining Cryptographers protocol in our concrete semantics. We first recall the protocol:

EXAMPLE 1
Three cryptographers are having dinner together, and the dinner is to be paid anonymously, by either one of them or by the National Security Agency (NSA). The cryptographers would like to know if it is NSA that paid for the dinner or not.

Each cryptographer flips an unbiased coin, so that only him and his neighbor at the right can see the outcome. The cryptographers that did not pay tell whether the two coins they see both fell on the same side. The cryptographer that pays (if any) tells the opposite (that is 'both heads' if he sees a head and a tail).

Show that each cryptographer is able to tell whether it's the NSA or one of them who paid the dinner.

The 3-agent model with concrete observability for this protocol is built as follows: consider first the three sets

$$Paid = \{p_0, p_1, p_2, p_{NSA}\}, Coin = \{h_0, h_1, h_2\}, Says = \{s_0, s_1, s_2\}$$

with $p_i \in \{0,1\}$ being an atomic proposition specifying whether it is the cryptographer i who paid the dinner, p_{NSA} whether its the NSA who paid, h_i whether the coin thrown by cryptographer i shows a head, and s_i whether the two coins that the cryptographer i sees show the same side. We will abuse notation and identify each subset $S \subseteq Paid$ with its characteristic function, and hence if $p_2 \in S$ we denote $S(p_2) = 1$, otherwise $S(p_2) = 0$. Denote also $\oplus_k$ the summation modulo k, that is, $2 \oplus_3 2 = 1$.

The states of the multi-agent systems are then

$$\begin{aligned} Q = &\{P \subseteq Paid \mid card(S) \leq 1\} \cup \{P \cup C \mid P \subseteq Paid, card(P) \leq 1, C \subseteq Coin\} \\ &\cup \{P \cup C \cup S \mid P \subseteq Paid, card(P) \leq 1, C \subseteq Coin, S \subseteq Says \text{ with} \\ &\qquad S(s_i) = C(h_i) \oplus_2 C(h_{i \oplus_3 1}) \oplus P(p_i)\}. \end{aligned} \tag{3}$$

Note that the condition 3 above encorporates exactly the requirement that the cryptographers that did not pay say the truth of what they see, and the one who paid says the opposite. Also, the set of initial states is $\{P \subseteq Paid \mid card(P) \leq 1\}$.

The transition relation is then:

$$\begin{aligned} \delta = &\{P \to P \cup C \mid P \subseteq Paid, card(P) \leq 1\} \\ &\cup \{P \cup C \to P \cup C \cup S \mid P \subseteq Paid, card(P) \leq 1, C \subseteq Coin, S \subseteq Says \text{ with} \\ &\qquad S(s_i) = C(h_i) \oplus_2 C(h_{i \oplus_3 1}) \oplus P(p_i). \end{aligned}$$

The concrete observability sets of atomic propositions are then defined as follows:

$$\Pi_i = \{p_i, h_i, h_{i \oplus_3 1}, s_1, s_2, s_3\}.$$

If we follow [12], the abstract observability relations would have to be defined as follows: first, the system needs to be decomposed into 'local transition systems', which have to be composed (with an asynchronous semantics of composition) to construct the whole system. This is needed for the identification of local states for each agent, and requires the definition of 10 automata. We do not

reproduce this process in detail here, pointing the interested reader to the paper [12]. Note that, though the semantics from [12] is not a 'history-based' semantics, it incorporates naturally synchrony and perfect recall in the case of the Dining Cryptographers.

Summarizing the process for constructing the abstract semantics, one defines the local state for agent i, $0 \leq i \leq 2$, as the intersection of the global state with $\{p_i, h_i, h_{i\oplus_3 1}, s_1, s_2, s_3\}$.

We believe that this example provides the essence of our argument that this semantics is simpler than the interpreted systems semantics. Note that we specify with an atomic proposition *what is seen* by each agent (is he the one who paid? what coins he observes? what is he hearing when everybody tell the others about the 'parity' they see?). Note also that there is no need to define, with a new relation/formula, the observability relation. This also means that the concrete semantics is very appropriate for symbolic verification.

Let us denote the resulting model $\mathcal{M}$. It is then easy to observe that

$$(\mathcal{M}, \rho, 0) \models \bigcirc\bigcirc \bigwedge_{i=1}^{3} \Big(K_i \big(\bigvee_{j=1}^{3} p_j \big) \vee K_i p_{NSA} \Big) \wedge \Big(\neg p_{NSA} \rightarrow \bigcirc\bigcirc \bigwedge_{j=1}^{3} P_i p_j \wedge P_i \neg p_j \Big),$$

which ensures both the anonymity (by the right subformula involving the P_i operators) and the knowledge whether it is the NSA who paid or not.

3 Undecidability of the satisfiability problem and non-axiomatizability for the concrete semantics

In this section, we give first an undecidability result for the satisfiability problem in KL_n with respect to *cpr*-observability and *cprs*-observability, and then, as a corollary, we show that the concrete semantics is not axiomatizable in the abstract semantics.

THEOREM 1
Satisfiability of KL_n formulas w.r.t. the *cprs*-observability semantics is undecidable.

PROOF. The main idea is to simulate an undecidable problem for Turing machines with the satisfiability problem of a recursive set of formulas. The undecidable problem that will be simulated is the following:

PROBLEM 2 (Looping Problem)
Given a deterministic Turing machine T, decide whether T, once it starts with a blank tape, has a reachable configuration which is reached infinitely often.

Hence, we simulate the Looping Problem 2 by the satisfiability of a KL_n formula ϕ_T. Similarly to [24], in each model $\mathcal{M}_T$ in which ϕ_T would be satisfied we code each of the configurations and transitions of the looping computation of T as a particular run in $\mathcal{M}_T$, and use three agents $1, 2, 3$ and their observability relations $\sim_i^{cprs}$ $(1 \leq i \leq 3)$ to connect configurations. The third agent 3 will be used for encoding the initial configuration of T, $(q_0, \varepsilon, 0)$. The synchronous and perfect recall characteristics of the semantics are needed for specifying that all configurations are simulated on a finite amount of space, which is 'copied', by synchrony and perfect recall, from one run encoding a configuration to the next. In the rest of the proof, we will utilize the notation $\sim_i$ instead of $\sim_i^{cprs}$ $(1 \leq i \leq 3)$.

So take a deterministic Turing machine $T = (Q, \Sigma, \delta, q_0)$ with $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}$. We will speak of elements of Q as *locations*, to avoid confusion with the states of $\mathcal{M}_T$, the models in which ϕ_T

is satisfied. Assume, without loss of generality, that the transitions in δ always changes location and tape cell, that is, if $\delta(q,a)=(r,b,dir)$ then $q \neq r$ and $dir \in \{L,R\}$. Assume also that δ is total—hence T may only visit all its tape cells, or cycle through some configuration, or halt because trying to move the head to the left when it points on the first tape cell.

The formula ϕ_T will be constructed over the set of atomic propositions consisting of:

(1) Four copies of Q, denoted Q, Q', $\overline{Q}$ and $\overline{Q}'$.
(2) Two copies of Σ, denoted Σ and Σ'.
(3) Two extra symbols *init* and $\perp$.

The meaning of the atomic proposition *init* is that it holds only in 'initial' states, that is, states which lie at the beginning of the runs that encode the looping computation of the Turing machine. The symbol $\perp$ will be used as a marker of the right end of the available space on the input tape on which T will be simulated, and its position will be 'guessed' at the beginning of the simulation. The utility of the copies of Q and Σ is explained in the following, along with the way the computation of T is simulated.

The computation steps of T are simulated by runs in the system $\mathcal{M}_T$ satisfying ϕ_T, runs which can be of two types:

(1) Type 1 runs, representing instantaneous configurations.
(2) Type 2 runs, representing transitions between configurations.

In a type 1 run ρ representing an instantaneous configuration (q,w,i)—where w is the contents of the tape and i is the head position—the configuration is coded using atomic propositions from Q and Σ in the following way:

- The contents of tape cell j, i.e., symbol $w_j = a \in \Sigma$, is an atomic proposition that holds in the jth state of ρ, and no other symbol holds in that state. That is, satisfiability of symbols from Σ is mutually exclusive.
- i is the unique state on the run in which the atomic proposition q holds, and no other symbol from Q holds along the whole run ρ.
- At each state on the run ρ where a symbol in $Q \cup \Sigma$ holds, the corresponding primed symbol in $Q' \cup \Sigma'$ holds too.
- The run ρ contains a state at which $\perp$ holds and from there on it holds forever. Also, no symbol from $Q \cup \Sigma$ holds when $\perp$ holds. This codes the finite amount of cell tapes used during simulation.
- At the state where $\perp$ holds, the symbols $\overline{q}$ and $\overline{q}'$ also hold on the run encoding (q,w,i). This is needed for coding the connection between type 1 runs and type 2 runs.
- The symbol *init* holds at a position on the tape before $\perp$, and marks the left bound of the tape. the 'tape left marker'.

In a type 2 run representing a transition between two configurations, say, $(q,w,i) \vdash (r,z,j)$, the unprimed symbols (from $Q \cup \Sigma$) along the run represent the configuration *before* the transition, while the primed symbols (from $Q' \cup \Sigma'$) represent the configuration *after* the transition, in a way completely similar to the above description. Hence, we will have some state on the run where location symbol q holds, and another state (immediately before or after) where symbol r' holds. Also, *init* marks the left bound of the tape and $\perp$ the limit of the available tape space, and $\overline{q}$ and $\overline{r}'$ hold wherever $\perp$ holds.

It then remains to connect type 1 runs (representing instantaneous configurations) with type 2 runs (representing transitions), by means of the observability relations. This connection will be implemented using three agents and their *cprs*-observability relations, defined by the following

subsets of atomic propositions:

$$\Pi_1 = Q \cup \overline{Q} \cup \Sigma \cup \{init, \bot\}$$
$$\Pi_2 = Q' \cup \overline{Q}' \cup \Sigma' \cup \{init, \bot\}$$
$$\Pi_3 = \{init, \bot\}$$
$$\Pi = \Pi_1 \cup \Pi_2 \cup \Pi_3.$$

REMARK 1
Note that for any two runs, $\rho \sim_1 \rho'$ implies $\rho \sim_3 \rho'$, and $\rho \sim_2 \rho'$ implies $\rho \sim_3 \rho'$.

We will connect, by means of $\sim_1$, each type 1 run ρ representing some configuration (q, w, i) with a type 2 run ρ' representing a transition $(q, w, i) \vdash (r, z, j)$. Then, by means of $\sim_2$, we code the connection between ρ' and another type 1 run ρ'', which represents the configuration (r, z, j). The first type of connection is imposed by means of the operator P_1, whereas the second type of connection is ensured by the employment of P_2. Finally, the operators K_3 will be used to impose the same constraints to all the runs that pass through a state labelled with *init*.

We give in Figure 1, an example of the association between a part of the computation of a Turing machine and a set of runs. The set of runs presented in Figure 1 is associated with the following computation of the Turing machine:

$$(q_0, ab, 1) \vdash^{\delta_1} (q_1, cb, 2) \vdash^{\delta_2} (q_2, cBB, 3) \vdash^{\delta_3} (q_3, cBa, 2), \tag{4}$$

where the transitions applied at each step are the following:

$$\delta_1(q_0, a) = (q_1, c, R), \qquad \delta_2(q_1, b) = (q_2, B, R), \qquad \delta_1(q_2, B) = (q_3, a, L).$$

The set of runs simulates a 'guess' that the Turing machine will utilize strictly less than 5 tape cells: on each run there are at most 4 tape cells simulated before the $\bot$ symbol. B denotes the blank tape symbol, whereas R, resp. L denote the commands 'move head to the right', respectively, 'to the left'. The run which ends with a state labelled with $\bot, \overline{q}_0, \overline{q}'_0$ is $\sim_1$-similar with the run ending with a state labelled $\bot, \overline{q}_0, \overline{q}'_1$, which is $\sim_2$-similar with a run ending in a state labelled $\bot, \overline{q}_1, \overline{q}'_1$. This encodes the first step in the computation in Identity 4 above.

Formally, ϕ_T is

$$\phi_T = \bigwedge_{i=0}^{10} \phi_i,$$

where $\phi_0, \ldots, \phi_{10}$ are the following formulas:

(0) ϕ_0, essential for imposing that ϕ_T must be satisfied on all the runs of type 1 or of type 2 which are built during the simulation, and specifying also that ϕ_T must be satisfied in a state modeling the left bound of the tape:

$$\phi_0 = init \wedge K_3 \bigcirc \Box \neg init \wedge \bigwedge_{z \in \Sigma \cup Q} (\neg z \wedge \neg z')$$

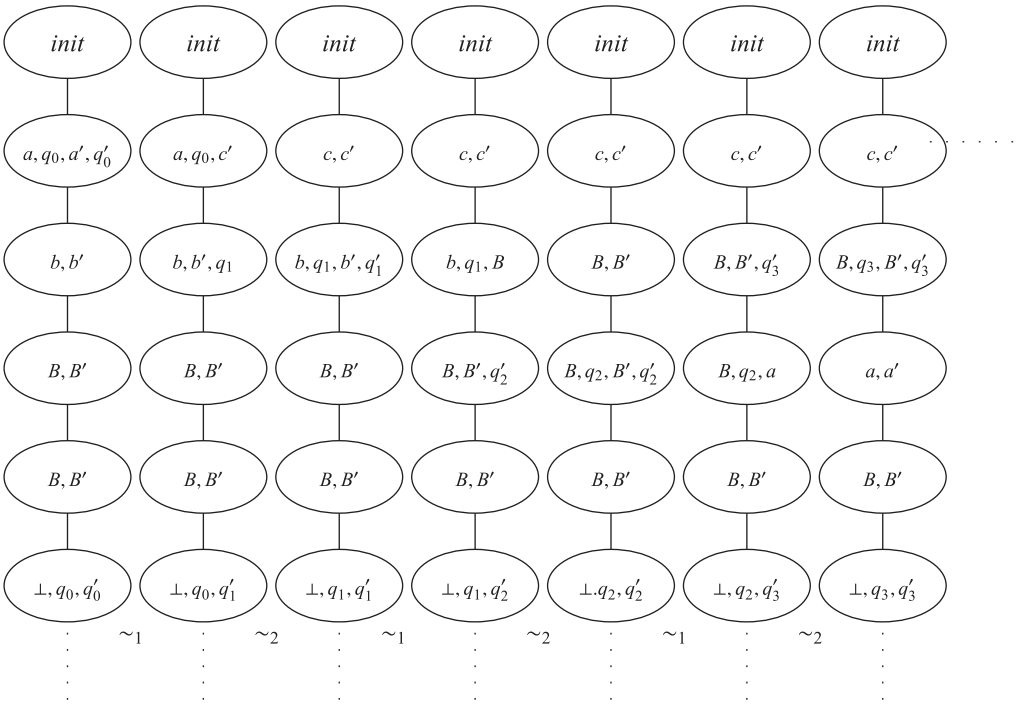

FIGURE 1. Simulating a sequence of transitions of a Turing machine within a set of runs. Here, $\Pi_1 = \{a, b, c, B, q_0, q_1, q_2, q_3, \bot, init\}$ and $\Pi_2 = \{a', b', c.B', q'_0, q'_1, q'_2, q'_3, \bot, init\}$. Note that all finite runs of length k are in relation $\sim_3$ with prefixes of length k of the first run on the left.

(1) ϕ_1, specifying that symbols in the same unprimed/primed/overlined set are mutually exclusive:

$$\phi_1 : K_3 \Box \bigwedge_{q,r \in Q, q \neq r} \Big(\neg(q \wedge r) \wedge \neg(q' \wedge r') \wedge \neg(\overline{q} \wedge \overline{r}) \wedge \neg(\overline{q}' \wedge \overline{r}') \Big)$$
$$\wedge K_3 \Box \bigwedge_{a,b \in \Sigma, a \neq b} \Big(\neg(a \wedge b) \wedge \neg(a' \wedge b') \Big)$$

(2) ϕ_2, specifying that on each run there exists a single occurrence of a location symbol, which marks the position of the R/W head.

$$\phi_2 : K_3 \Big(\big(\Diamond \bigvee_{q \in Q} q \big) \wedge \big(\Diamond \bigvee_{q \in Q} q' \big) \Big) \wedge K_3 \Box \Big((q \rightarrow \bigcirc \Box \neg q) \wedge (q' \rightarrow \bigcirc \Box \neg q') \Big)$$

(3) ϕ_3, which, in combination with ϕ_7 and ϕ_9 below, is used to encode the fact that the simulation of T is done on a finite tape, whose end is marked with $\bot$:

$$\phi_3 : K_3 \Diamond \bot \wedge K_3 \Box \Big(\bot \rightarrow \Box \big(\bot \wedge \bigwedge_{z \in \Sigma \cup Q} (\neg z \wedge \neg z') \big) \Big)$$

(4) ϕ_4 which copies both q and q' into the state of the run where the end marker $\bot$ occurs – here q is the location of the simulated configuration (q, w, i); the two copies are $\overline{q}$, $\overline{q}'$. This formula

is useful for connecting configurations with the aid of $\sim_1$ and $\sim_2$:

$$\phi_4 : K_3 \Box \bigwedge_{q \in Q} \Big(\big(q \to \Box(\bot \to \overline{q})\big) \wedge \big(q' \to \Box(\bot \to \overline{q}')\big) \Big)$$

(5) ϕ_5, specifying that on a run of type 1 the primed and unprimed symbols are the same:

$$\phi_5 : K_3 \Box \bigwedge_{q \in Q} \Big(q \wedge q' \to \Box \bigwedge_{a \in \Sigma} a \leftrightarrow a' \Big) \wedge K_3 \Box \bigwedge_{q \in Q} \Big(\Diamond(q \wedge q') \to \bigwedge_{a \in \Sigma} a \leftrightarrow a' \Big)$$

(6) ϕ_6, specifying that on a run of type 2, the primed and unprimed symbols are almost everywhere the same, excepting the current position of the R/W head:

$$\phi_6 : K_3 \Box \bigwedge_{q \in Q} \Big(q \wedge \neg q' \to \Box \bigcirc \bigwedge_{c \in \Sigma} c \leftrightarrow c' \Big) \wedge$$
$$K_3 \Box \bigwedge_{q \in Q} \Big(\Diamond \bigcirc (q \wedge \neg q') \to \bigwedge_{c \in \Sigma} c \leftrightarrow c' \Big) \wedge$$

(7) ϕ_7, specifying that if a run ρ is of type 1 and encodes a configuration in which a certain transition can be applied, then there exists a run ρ' of type 2 which encodes the respective transition. (The format of the unique transition which will be applied is the subject of ϕ_8.) ϕ_7 also specifies that ρ' carries the end marker at the same position as ρ:

$$\phi_7 : K_3 \Box \bigwedge_{(q,a) \in \mathsf{supp}(\delta)} \Big(q \wedge a \wedge q' \to \Box \big(\bot \to P_1 \neg \overline{q}' \big) \Big)$$

(8) ϕ_8, specifying that each transition which can be applied in a certain configuration must be applied in that configuration:

$$\phi_8 : K_3 \Box \bigwedge_{q \in Q, a \in \Sigma, \delta(q,a) = (r,b,L)} \Big(\bigcirc (q \wedge a \wedge \neg q') \to (r' \wedge \bigcirc b') \Big) \wedge$$
$$K_3 \Box \bigwedge_{q \in Q, a \in \Sigma, \delta(q,a) = (r,b,R)} \Big(q \wedge a \wedge \neg q' \to (\bigcirc r' \wedge b') \Big)$$

Recall that T is deterministic, and hence in each configuration at most one transition can be applied. Note also that the $\bigcirc$ operator is needed on the first line above, in order to code the situations when the head is on the first tape cell and tries to move left—in such situations the machine halts, no next configuration exists, and therefore our formula ϕ_T is unsatisfiable.

(9) ϕ_9, specifying that the outcome of a transition, as coded in a type 2 run ρ, is copied, via $\sim_2$, into a type 1 run ρ'. ϕ_9 also specifies that ρ' carries the end marker $\bot$ at the same position as ρ:

$$\phi_9 : K_3 \bigwedge_{q \in Q} \Box \Big(q' \wedge \neg q \to \Box \big(\bot \to P_2 \overline{q} \big) \Big)$$

(10) ϕ_{10}, encoding the initial configuration of T – here q_0 is the initial location of T:

$$\phi_{10} : \bigcirc (q_0 \wedge q_0') \wedge \bigcirc (B \wedge B') \mathcal{U} \bot \big).$$

Here, B represents the blank symbol from Σ.

Hence the satisfiability of the conjunct ϕ_{10} of formula ϕ_T in a model $\mathcal{M}_T$ ensures the existence of a run coding the initial configuration of T, together with a guess of the amount of tape space needed for simulating the whole behaviour of T. Then, by means of ϕ_7, ϕ_8 and ϕ_9, we have that once a type 1 run ρ encoding a configuration (q,w,i) exists in $\omega\mathsf{Runs}(\mathcal{M}_T)$, and some transition $(q,w,i)\vdash(r,z,j)$ may be fired, then a type 2 run ρ' encoding this transition exists, and the resulting configuration (r,z,j) is also encoded in another type 1 run ρ'. It then follows that $\mathcal{M}_T \models \phi_T$ if and only if there exists a finite subset $Z \subseteq \omega\mathsf{Runs}(\mathcal{M})$ representing the evolution of the Turing machine T, starting with a blank tape. Note that T cannot then halt by trying move its head past the left end of the tape. Also note that there is an inital guess of a finite amount of tape that T uses during its computation, and all runs in Z simulate configurations that do not use more than this guessed amount of space.

Proof of the correctness of the construction: In the following, given a Turing machine $T = (Q, \Sigma, \delta, q_0)$, we write its configurations as words in the language $\Sigma^* \cdot Q \cdot \Sigma^*$. Hence, the word $a_1 a_2 q B a_3$ denotes the configuration in which the tape contains symbols a_1, a_2, B, a_3 (in this order), and the head points to the 3rd cell. Note that the same configuration is represented by the word $a_1 a_2 q B a_3 BBBBB$.

Suppose T has a looping computation, denote it $\left(c_{i-1} \vdash c_i\right)_{1\leq i\leq n}$, with $c_n = c_k$ for some $1 \leq k \leq n-1$. Suppose that the maximal number of cell tapes that are used during this computation is m. The word representing i-th configuration in this computation is denoted $c_i = a^i_1 a^i_2 \ldots q_i a^i_{p_i} \ldots a^i_m$—hence all words have the same length.

We construct a model for the formula ϕ_T by encoding the configurations $c_1, \ldots, c_n$ into n type 1 runs, and by creating also $n-1$ type 2 runs which encode each computation step $c_{i-1} \vdash c_i$. The model is the following: $\mathcal{M} = \left(S, \rightarrow, S_0, (\sim^{cpr}_A), \Pi, (\Pi_A)_{A\in Ag}, \nu\right)$, where

- $S = \left\{(c_i, j) \mid 1 \leq i \leq n, 0 \leq j \leq m+1\right\} \cup \left\{(c_{i-1}, c_i, j) \mid 2 \leq i \leq n, 0 \leq j \leq m+1\right\}$.
- $S_0 = \left\{(c_i, 0) \mid 1 \leq i \leq n\right\} \cup \left\{(c_{i-1}, c_i, 0) \mid 2 \leq i \leq n\right\}$.
- The sets of atomic propositions Π, Π_1, Π_2, Π_3 are as listed above.
- The state labelling function is defined as follows:

$$\begin{aligned}
\nu((c_i, j)) &= \{a^i_j, (a^i_j)'\} \cup \{q_i, (q_i)' \mid j = p_i\} \\
\nu((c_{i-1}, c_i, j)) &= \{a^{i-1}_j, (a^i_j)'\} \cup \{q_{i-1} \mid j = p_{i-1}\} \cup \{(q_i)' \mid j = p_i\} \\
\nu((c_i, 0)) &= \nu((c_i, 0)) = \{init\} \\
\nu((c_i, m+1)) &= \{\bot\} \cup \{\overline{q_i}, \overline{q_i}' \mid j = p_i\} \\
\nu((c_{i-1}, c_i, m+1)) &= \{\bot\} \cup \{\overline{q_{i-1}} \mid j = p_{i-1}\} \cup \{\overline{q_i}' \mid j = p_i\}.
\end{aligned}$$

- The transition relation is the following (the domain for each index is omitted):

$$\begin{aligned}
&(c_i, j) \rightarrow (c_i, j+1) && (c_i, m+1) \rightarrow (c_i, m+1) \\
&(c_{i-1}, c_i, j) \rightarrow (c_{i-1}, c_i, j+1) && (c_{i-1}, c_i, m+1) \rightarrow (c_{i-1}, c_i, m+1).
\end{aligned}$$

Clearly, $(\mathcal{M}, \rho, 0) \models \phi_T$ for each run which starts in $(c_1, 0)$.

For the reverse implication, suppose that $\mathcal{M}$ is a system, ρ an infinite run in $\mathcal{M}$ and i is an index on the run such that $(\mathcal{M}, \rho, i) \models \phi_T$. We will show that, for any integer k, we may construct a sequence of runs, $\mathcal{R}_k = (\rho_i)_{1\leq i\leq k}$, all having the same length, such that ρ_{2i-1} encodes the configuration c_i in the *unique computation* of the deterministic Turing machine T, (which means that ρ_{2i-1} is a type 1 run) and ρ_{2i} encodes the computation step $c_i \vdash c_{i+1}$ (which means that ρ_{2i} is a type 2 run). Since

there can only be a finite number of distinct runs, two of them must be identical, which will imply that the encoded configurations must be identical – that is, that T has a looping computation.

Before going to the proof of this claim, we formalize the encoding of a configuration by a run. Consider the mapping $\mu : S \to \Sigma \cup \Sigma \cdot Q$ (with $\Sigma \cdot Q$ denoting the formal concatenation of the two sets), defined by

$$\mu(s) = \begin{cases} q \cdot a & \text{if } q, a \in \nu(s) \\ a & \text{if } a \in \nu(s) \text{ and } \nu(s) \cap Q = \emptyset. \\ \bot & \text{if } \bot \in \nu(s) \\ \varepsilon & \text{otherwise} \end{cases}$$

Similarly, we define the mapping $\mu' : S \to \Sigma \cup \Sigma \cdot Q$, with

$$\mu'(s) = \begin{cases} q \cdot a & \text{if } q', a' \in \nu(s) \\ a & \text{if } a' \in \nu(s) \text{ and } \nu(s) \cap Q' = \emptyset. \\ \bot & \text{if } \bot \in \nu(s) \\ \varepsilon & \text{otherwise} \end{cases}$$

Given an infinite run $\rho = (s_{i-1} \to s_i)_{i \geq 1}$ for which there exist $i_0, i_1 \in \mathbb{N}$ with $init \in \nu(s_{i_0})$ and $\bot \in \nu(s_{i_1})$, we say that:

- *ρ encodes, between indices i_0 and i_1, the configuration* $\mu(s_{i_0+1}) \ldots \mu(s_{i_1-1})$ if there exist $q \in Q, a \in Q$ and a unique index $i_0 < i_2 < i_1$ with $\mu(s_{i_2}) = qa = \mu'(s_{i_2})$.
- *ρ encodes, between indices i_0 and i_1, the computation step:*

$$\mu(s_{i_0}+1) \ldots \mu(s_{i_1}-1) \vdash \mu'(s_{i_0+1}) \ldots \mu'(s_{i_1}-1)$$

  if there exists $i_0 < i_2 < i_1$ with $\mu(s_{i_2}) = qa$, $\mu'(s_{i_2}) = b$, and one of the following properties hold:
  - Either $\mu'(s_{i_2+1}) = rc$, with $\delta(q,a) = (r,b,R)$.
  - Or $\mu'(s_{i_2-1}) = rc$, with $\delta(q,a) = (r,b,L)$.

  In both cases, we denote $\mu(\rho) = \mu(s_{i_0+1}) \ldots \mu(s_{i_1-1})$ and $\mu'(\rho) = \mu'(s_{i_0+1}) \ldots \mu'(s_{i_1-1})$.

The formalization of our claim is then the following:

CLAIM

For any $k \in \mathbb{N}$, there exists a sequence of infinite runs $\mathcal{R}_k = (\rho_i)_{1 \leq i \leq k}$ and two indices $i_0, i_1 \in \mathbb{N}$ satisfying the following properties:

(1) ρ_{2i-1} encodes the configuration c_i between i_0 and i_1.
(2) ρ_{2i} encodes the computation step $c_i \vdash c_{i+1}$ also between i_0 and i_1.
(3) $\rho_{2i-1}[0..i_1] \sim_1 \rho_{2i}[0..i_1]$ and $\rho_{2i}[0..i_1] \sim_2 \rho_{2i+1}[0..i_1]$.
(4) $(\mathcal{M}, \rho_i, i_0) \models \phi_T$.

We prove the claim by induction on k. The base step can be proved as follows: since $(\mathcal{M}, \rho, i) \models \phi_T$, we must have $(\mathcal{M}, \rho, i) \models \phi_{10}$, which means that $q_0, q_0' \in \nu(\rho[i])$ and there exists $N \geq i$ with $\bot \in \nu(\rho[N])$. Then clearly, the run ρ gives an encoding of the initial configuration of the Turing machine, between indices i and N.

For the induction step, two cases may occur, depending on whether the last run in the constructed sequence $\mathcal{R}_k$ is a type 1 run or a type 2 run.

Suppose that k is odd, $k=2l-1$, hence ρ_{2l-1} is a type 1 run which encodes the configuration c_l between i_0 and i_1 and with $(\mathcal{M},\rho_{2l-1},i_0)\models\phi_T$. This means several things:

- $\rho[i_0]$ is labelled with *init* and $\rho[i_1]$ is labelled with $\perp$.
- $\mu(\rho_{2l-1})=\mu'(\rho_{2l-1})$, that is, the sequence of primed and unprimed symbols in ρ_{2l-1} is the same.
- $(\mathcal{M},\rho,i_0)\models\bigwedge_{i=0}^{10}\phi_i$.

Assume that the Turing state in configuration c_l is q, and the machine head points to tape cell j where we see a. Since $(\mathcal{M},\rho_{2l-1},i_0)\models\phi_4$, we must have that ρ_{2l-1} is labelled with $\overline{q}$ and with $\overline{q}'$.

On the other hand, $(\mathcal{M},\rho_{2l-1},i_0)\models\phi_7$ implies that $(\mathcal{M},\rho_{2l-1},i_1)\models\overline{q}\wedge P_1\neg\overline{q}'$. Since ρ_{2l-1} does not satisfy $\overline{q}\wedge\neg\overline{q}'$, there must exist another run ρ' with $(\mathcal{M},\rho',i_1)\models\neg\overline{q}'$, and with $\rho'[0..i_1]\sim_1\rho_{2l-1}[0..i_1]$.

Two essential facts can be deduced:

- The relation between ρ' and ρ_{2l-1} also implies that $\rho'[0..i_0]\sim_3\rho_{2l-1}[0..i_0]\sim_3\rho_1[0..i_0]$, as noted in Remark 1. This means that we also have that
$$(\mathcal{M},\rho',i_0)\models\bigwedge_{i=0}^{10}\phi_i.$$
- $\mu(\rho')=\mu(\rho_{2l-1})$, that is, ρ' encodes, on the unprimed symbols, the configuration c_l.

Three situations have to be studied:

(1) The position of the tape cell j satisfies $i_1+1<j<i_2-1$.
(2) $j=i_0+1$ and the transition is $\delta(q,a)=(r,b,L)$.
(3) $j=i_1-1$ and the transition is $\delta(q,a)=(r,b,R)$.

In the first case, ρ' must encode the unique computation step $c_l\vdash c_{l+1}$ in the (deterministic!) Turing machine, by virtue of the combined constraints imposed by the subformulas $\phi_1,\ldots,\phi_9$. Hence ρ' can be appended to $\mathcal{R}_k$ such that the resulting sequence of runs satisfies the conclusion of our claim.

We will prove that the second and the third case cannot occur if ϕ_T is satisfied in $\mathcal{M}$. To this end, we have to observe that, since $(\mathcal{M},\rho',i_0)\models\phi_8$, we should also have that $(\mathcal{M},\rho',i_0)\models r'$. But this is in contradiction with ϕ_0, which says that at position i_0 no atom from Q' must hold true. That is, the Turing machine does not halt in the configuration c_l. A similar proof can be given for the thid case above, which ends the proof when k is odd.

The situation where k is even can be treated along the same ideas. This ends our proof of Theorem 1. ■

We may accomodate the same proof for the case of *cpr*-observability semantics with the following trick: we insert an atomic proposition *tick* that is supposed to be observed by all agents, and its value is supposed to flip, at each state, from true to false, or vice-versa. Basically, this means that, synchrony can always be obtained by the observability of a 'clock' variable like *tick*.

Formally, and consider the formula

$$Ticks:=init\rightarrow K_I\Box(tick\rightarrow\bigcirc\neg tick)\wedge K_I\Box(\neg tick\rightarrow\bigcirc tick),$$

where I plays the role of the agent 3 in the above proof, agent which is only able to see the atomic proposition *init*. This formula encodes the fact that *tick* must hold on each second position of a run that starts in an initial state, and must be false in between.

Take a multi-agent system with *cprs*-observability, $\mathcal{M} = \left(S, \rightarrow, S_0, (\sim_A^{cprs})_{A \in \mathsf{Ag}}, \Pi, (\Pi_A)_{A \in Ag}, \nu\right)$ and put $\Pi'_A = \Pi_A \cup \{tick, init\}$ – hence each agent can observe *tick* and *init*.

We may associate with $\mathcal{M}$ the following multi-agent system with *cpr*-observability: $\mathcal{M}_{tick} = \left(S, \rightarrow, S_0, (\sim_A^{cpr})_{A \in \mathsf{Ag}'}, \Pi', (\Pi_A)_{A \in Ag'}, \nu\right)$, in which $Ag' = Ag \cup \{I\}$ and $\Pi_I = \{init\}$. It is then easy to observe the following property:

PROPOSITION 1
For any formula ψ of the form $\psi = init \wedge K_I \bigcirc \Box \neg init \wedge K_I \phi$, and for any infinite run ρ and position i in ρ, $(\mathcal{M}, \rho, i) \models_{cprs} \psi$ if and only if $(\mathcal{M}_{tick}, \rho, i) \models_{cpr} \psi \wedge Ticks$.

As a consequence, the formula ϕ_T associated with some Turing machine T (with the restrictions from Theorem 1) is satisfiable in some model $\mathcal{M}$ with *cprs*-observability if and only if $\phi_T \wedge tick$ is satisfiable in $\mathcal{M}_{tick}$ which has *cpr*-observability.

3.1 *Non-axiomatizability of the cprs-observability and the cpr-observability semantics*

We start this subsection with the following restatement of the conclusion of Theorem 1:

REMARK 2
Theorem 1 provides us with a non-recursively enumerable set of formulas in KL_n: it is the set of formulas corresponding to Turing machines that start with a blank tape and do not have a repeating configuration, denote it

$$InfVisit = \left\{\phi_T \mid T \text{ visits all the cells of its tape or has a blocking computation}\right\}.$$

We may use this to prove the main result[1] of this article:

THEOREM 2
The semantics of KL_n based on *cprs*-observability or on *cpr*-observability do not have a recursively enumerable axiomatization.

PROOF. Suppose that such a r.e. axiomatization existed. Then the set $ValidKL_n$ of all formulas which are valid w.r.t. to *cprs*-observability or *cpr*-observability would also be r.e.

On the other hand, the set of formulas of the form $\neg\phi_T$ produced in the proof of Theorem 1, is a recursive set—denote this set *TurForm*. Since r.e. sets are closed under intersection, $ValidKL_n \cap TurForm = InfVisit$ would have to be r.e. too. But this contradicts the conclusion of Remark 2, which ends the proof of our result. ■

4 Conclusions

We have presented a 'concrete' semantics for KL_n , with two variants, a perfect recall variant and a variant with perfect recall and synchrony in observations. We have shown that the classical semantics can be encoded in this semantics, but the reverse does not hold, in the sense that the concrete semantics is not axiomatizable in the classical semantics.

[1] Which is a remark due to Dimitar Guelev, see the acknowledgements section.

To summarize, we have a temporal epistemic logic, the KL_n or $(PTL+S5(n))$ with two semantics:

(1) The interpreted systems semantics from [9] for which the logic is finitely axiomatizable and has a decidable satisfiability problem.
(2) The concrete semantics presented here, which is a variant of the Kripke semantics, for which the logic has no r.e. axiomatization and an undecidable satisfiability problem.

We believe that, in spite of these problems, the concrete semantics is more natural for expressing system models, as it does not require explicitely presenting, with the aid of some extra relation or formula, the observability relation. The appropriate approach would then be the *model-checking* approach. We have already shown that, for the case of CTL with individual knowledge operators (that is, for KB_n) with the concrete semantics, the model-checking problem is decidable [2]. The same result could be proved for KL_n with the concrete semantics, and the complexity would be the same as in the case of the interpreted systems semantics [25].

One of the aims of introducing this concrete semantics is to relate epistemic logics with logics over linear or branching orders. In [18], they have shown that the model-checking problem for a variant of KL_n can be solved using chain logic with equal-level predicate. We conjecture that both KL_n and KB_n (the branching-time temporal epistemic logic from [9]) with the concrete pr. or prs. semantics are expressively equivalent with fragments of the chain logic with equal-level predicate [21].

Acknowledgements

The author acknowledges the many discussions with Dimitar Guelev on temporal logics with knowledge, during his visits at the LACL in October–November 2007 and in May 2009. The remark that Theorem 1 actually provides a non-r.e. set of valid formulas, and, as such, proves that the concrete semantics is non-axiomatizable, is due to him.

References

[1] R. Chadha, S. Delaune, and S. Kremer. Epistemic logic for the applied pi calculus. In *Proceedings of FMOODS/FORTE'09*, Vol. 5522 of *Lecture Notes in Computer Science*, pp. 182–197. Springer, 2009.
[2] C. Dima. Revisiting satisfiability and model-checking for CTLK with synchrony and perfect recall. In *Proceedings of the 9th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA IX)*, Vol. 5405 of *Lecture Notes in Artificial Intelligence*, pp. 117–131. Springer, 2008.
[3] C. Dixon, M. Fernández Gago, M. Fisher, and W. van der Hoek. Temporal logics of knowledge and their applications in security. *Electronic Notes in Theoretical Computer Science*, **186**, 27–42, 2007.
[4] K. Engelhard, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In *Proceedings of TARK'98*, pp. 29–41. Morgan Kaufman, 1998.
[5] J. Halpern and R. Fagin. A formal model of knowledge, action, and communication in distributed systems: preliminary report. In *Proceedings of PODC'85*, pp. 224–236. ACM, 1985.
[6] J. Halpern, R. Fagin, Y. Moses, and M. Vardi. *Reasoning about knowledge*. The MIT Press, 2004.
[7] J. Halpern and K. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, **13**, 483–512, 2005.

[8] J. Halpern, R. van der Meyden, and M. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal of Computing*, **33**, 674–703, 2004.
[9] J. Halpern and M. Vardi. The complexity of reasoning about knowledge and time: extended abstract. In *Proceedings of STOC'86*, pp. 304–315, 1986.
[10] J. Halpern and M. Vardi. The complexity of reasoning about knowledge and time. I. Lower bounds. *Journal of Computer System Sciences*, **38**, 195–237, 1989.
[11] H. Jonker and W. Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Proceedings of IAVoSS Workshop On Trustworthy Elections (WOTE2006)*, Robinson College, 2006.
[12] M. Kacprzak, A. Lomuscio, A. Niewiadomski, W. Penczek, F. Raimondi, and M. Szreter. Comparing BDD and SAT based techniques for model checking Chaum's Dining Cryptographers Protocol. *Fundamenta Informaticae*, **72**, 215–234, 2006.
[13] R. Ladner and J. Reif. The logic of distributed protocols. In *Proceedings of TARK'86*, pp. 207–222. Morgan Kaufmann, 1986.
[14] A. Lomuscio and W. Penczek. Ldyis: A framework for model checking security protocols. *Fundamenta Informaticae*, **85**, 359–375, 2008.
[15] A. Lomuscio and M. Ryan. On the relation between interpreted systems and kripke models. In *Proceedings of AAMAS'97*, Vol. 1441 of *Lecture Notes in Computer Science*, pp. 46–59. Springer, 1998.
[16] A. Lomuscio and B. Wozna. A complete and decidable security-specialised logic and its application to the tesla protocol. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pp. 145–152. ACM, 2006.
[17] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. *Journal of Applied Logic*, **5**, 235–251, 2005.
[18] N.V. Shilov and N.O. Garanina. Model checking knowledge and fixpoints. In *Proceedings of FICS'02*, pp. 25–39, Extended version available as Preprint 98, Ershov Institute of Informatics, Novosibirsk, 2002.
[19] K. Su, Q. Chen, A. Sattar, W. Yue, Gu. Lv, and X. Zheng. Verification of authentication protocols for epistemic goals via sat compilation. *Journal of Computer Science and Technology*, **21**, 932–943, 2006.
[20] K. Su, Gu. Lv, and Q. Chen. Knowledge theoretic approach to formal verification of authentication protocols. *Science in China, Series F: Information Science*, **48**, 513–532, 2006.
[21] W. Thomas. Infinite trees and automation-definable relations over omega-words. *Theoretical Computer Science*, **103**, 143–159, 1992.
[22] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, Vol. 3, Beyond Words, G. Rozenberg and A. Salomaa, eds, pp. 389–455. Springer, 1997.
[23] J. van Benthem and E. Pacuit. The tree of knowledge in action: towards a common perspective. In *Proceedings of AiML'06*, Guido Governatori, Ian M. Hodkinson, and Yde Venema, eds, pp. 87–106. College Publications, 2006.
[24] R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, **140**, 115–157, 1998.
[25] R. van der Meyden and N.V. Shilov. Model checking knowledge and time in systems with perfect recall (extended abstract). In *Proceedings of FSTTCS'99*, Vol. 1738 of *Lecture Notes in Computer Science*, pp. 432–445, 1999.
[26] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004)*, pp. 280–291. IEEE Computer Society, 2004.

[27] H. van Ditmarsch, W. van der Hoek, R. van der Meyden, and Ji Ruan. Model checking russian cards. *Electronic Notes in Theoretical Computer Science*, **149**, 105–123, 2006.
[28] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Model checking a knowledge exchange scenario. *Applied Artificial Intelligence*, **18**, 937–952, 2004.

Received 22 October 2009