An Epistemic Strategy Logic

XIAOWEI HUANG, University of Liverpool
RON VAN DER MEYDEN, UNSW Sydney

This article presents an extension of temporal epistemic logic with operators that can express quantification over agent strategies. Unlike previous work on alternating temporal epistemic logic, the semantics works with systems whose states explicitly encode the strategy being used by each of the agents. This provides a natural way to express what agents would know were they to be aware of some of the strategies being used by other agents. A number of examples that rely on the ability to express an agent's knowledge about the strategies being used by other agents are presented to motivate the framework, including reasoning about game-theoretic equilibria, knowledge-based programs, and information-theoretic computer security policies. Relationships to several variants of alternating temporal epistemic logic are discussed. The computational complexity of model checking the logic and several of its fragments are also characterized.

26

Q1 CCS Concepts:

Q2 Additional Key Words and Phrases:

ACM Reference format:
Xiaowei Huang and Ron van der Meyden. 2018. An Epistemic Strategy Logic. *ACM Trans. Comput. Logic* 19, 4, Article 26 (September 2018), 45 pages.
https://doi.org/10.1145/3233769

1 INTRODUCTION

In distributed and multi-agent systems, agents typically have a choice of actions to perform and have individual and possibly conflicting goals. This leads agents to act strategically, attempting to select their actions over time so as to guarantee achievement of their goals even in the face of other agents' adversarial behaviour. The choice of actions generally needs to be made on the basis of *imperfect* information concerning the state of the system.

These concerns have motivated the development of a variety of modal logics that aim to capture aspects of such settings. One of the earliest, dating from the 1980s, was multi-agent *epistemic logic* [23, 44], which introduced modal operators that deal with imperfect information by providing a way to state what agents *know*. Combining such constructs with temporal logic constructs [46] gives *temporal epistemic logics*, which support reasoning about how agents' knowledge

This article combines results from Reference [31], An epistemic strategy logic, X. Huang and R. van der Meyden, in *Proceedings of the 2nd International Workshop on Strategic Reasoning*, and Reference [33] A temporal logic of strategic knowledge, X. Huang and R. van der Meyden, In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. It extends these works by including full proofs for all results.
Most of Huang's work was performed when he was at UNSW Sydney, Australia.
Q3 Authors' addresses:
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
1529-3785/2018/09-ART26 $15.00
https://doi.org/10.1145/3233769

changes over time. Temporal-epistemic logic is an area about which a significant amount is now understood [19].

Logics dealing with reasoning about strategies, which started to be developed in the same period [43], had a slower initial start but have in recent years become the focus of intense study [2, 30, 45]. *Alternating temporal logic* (ATL) [2], which generalizes branching-time temporal logic to encompass reasoning about the temporal effects of strategic choices by one group of agents against all possible responses by their adversaries, has become a popular basis for work in this area.

One of the ways in which recent work has extended ATL is to add epistemic operators, yielding an *alternating temporal epistemic logic*, e.g., ATEL [29]. Many subtle issues arise concerning what agents know in settings where multiple agents act strategically. In the process of understanding these issues, there has been a proliferation of epistemic extensions of ATL [34, 35, 39, 55]. Some of the modal operators introduced in this literature are complex, interweaving ideas about the knowledge of a group of agents, the strategies available to them, the effect of playing these strategies against strategies available to agents not in the group, and the knowledge that other groups of agents may have about these effects.

Our contribution in this article is to develop a logic that extends the expressive power of previous work on logics for knowledge and strategies, while at the same time simplifying the syntactic basis by identifying a small set of primitives that can be composed to represent the more complex constructs for reasoning about strategies and knowledge from prior literature. We present examples to show that the logic is useful for a range of applications, including expressing notions of information flow security (such as strategic notions of noninterference and erasure policies), reasoning about implementations of knowledge-based programs, and reasoning about game-theoretic equilibria. We also conduct a detailed analysis of the complexity of model checking a number of fragments of the logic. Our semantic framework is able to model a range of semantics for knowledge and strategies including a "perfect recall" interpretation, but since we are interested in model-checking complexity results at the lower end of the complexity spectrum, we concentrate on an "imperfect recall" or "observational" semantics of knowledge. (We note that model checking just ATL, even without knowledge operators, under an imperfect information and perfect recall semantics is already undecidable [16].)

At the semantic level, the key way in which our logic extends prior work on alternating temporal epistemic logic is by treating agents' strategies as first-class citizens in the semantics, represented as components of the global state of the system at any moment of time in a run of the system. This is in contrast to most prior work in the area, where strategies are used to generate runs of a system, but the runs themselves contain no explicit information about the specific strategies used by the players to produce them. Our approach provides a referent for the notion "the strategy being used by player i," which cannot be expressed in most prior works on alternating temporal epistemic logic.

We reflect this additional referent at the syntactic level by introducing a syntactic notation $\sigma(i)$, which refers to the strategy of agent i. Since the strategy of agent i is modelled semantically as a component of the global state, just like the local state of agent i, we allow this construct to be used in the same contexts where the agent name i can be used—in particular, in operators for knowledge (including distributed and common knowledge). An example of what can be expressed with this extension is $D_{\{i,\sigma(i)\}}\phi$, which says that the truth of ϕ in all possible futures can be deduced from knowledge of agent i's local state plus the strategy being applied by agent i. Intuitively, the construct $D_{\{i,\sigma(i)\}}$ captures what agent i knows when it takes into account the strategy it is running.

We show that this extension of temporal epistemic logic gives a logical approach with broad applicability. In particular, as we show in Section 3.2, temporal epistemic logic extended with the indices $\sigma(i)$ can express alternating temporal logic constructs (both revocable and irrevocable).

The extension can also express many of the subtly different notions that have been proposed in the literature on alternating temporal epistemic logics. We demonstrate this (in Section 3.3) by results that show how a number of such logics can be translated into our setting. We also present a number of other applications including game-theoretic solution concepts (Section 3.5), issues of concern in computer security (Section 3.6), and reasoning about possible implementations of knowledge-based programs (Section 3.7).

In some applications, however, some richer expressiveness is required. One such application, discussed in Section 3.3, concerns expressing an operator and combining common knowledge and strategic concerns from an extended alternating temporal epistemic logic of Jamroga and van der Hoek [38]. We address this by adding to the logic constructs that can be used to express quantification over strategies. This leads to a logic that, like *strategy logic* [13, 42], supports explicit naming and quantification over strategies. Technically, we achieve this in a slightly more general way: We first generalize temporal epistemic logic to include operators $\exists x$ for quantification over global states x, as well as statements $\mathsf{e}_i(x)$ that say that component i in the current global state is the same as component i in the global state denoted by x. Even before the introduction of strategic concerns, this gives a novel extension of temporal epistemic logic in the flavour of *hybrid logic* [4]. (As we show in Section 2, this extension enables the expression of security notions such as *nondeducibility* [53] that cannot be naturally expressed in standard temporal epistemic logics.) We then apply this generalization to a system that includes strategies encoded in the global states and references these using the "strategic" indices $\sigma(i)$. The resulting logic can express that agent i knows what strategy agent j is using, by means of the formula

$$\exists x(\mathsf{e}_{\sigma(j)}(x) \wedge K_i \mathsf{e}_{\sigma(j)}(x)),$$

in which the first occurrence of $\mathsf{e}_{\sigma(j)}(x)$ binds x to a global state in which the strategy of agent j is the same as at the current state, and the remainder of the formula states that every global state considered possible by agent i has the same strategy for agent j. (This cannot be expressed in most alternating temporal epistemic logics, e.g., ATEL [29], since their semantics fails to encode the strategy being run by an agent in the locus of evaluation of formulas.) The framework is able to express the above-mentioned operator from Reference [38], as well as notions of information flow security that quantify over agent strategies, such as *nondeducibility on strategies* [57], which we discuss in Section 3.1.

The main theoretical contribution of the article is a set of results on the complexity of model checking the resulting logic and its fragments. We consider several dimensions: Does the logic have quantifiers, and what is the temporal basis for the logic, branching-time (CTL) or linear time (LTL)? The richest logics in our spectrum turn out to have EXSPACE-complete model-checking problems. However, we identify a number of special cases where model checking is in PSPACE, i.e., no more than the complexity of model checking the temporal logic LTL. One is the fragment where we allow the constructs $\exists x$ and $\mathsf{e}_i(x)$ but restrict the temporal operators to be those of the branching-time logic CTL. Another is the fragment in which we do not allow $\exists x$ and $\mathsf{e}_i(x)$ but allow strategic indices $\sigma(i)$ in knowledge operators and take the temporal operators from the richer branching-time logic CTL*, which extends the linear time logic LTL.

The structure of the article is as follows. In Section 2, we first develop an extension of temporal epistemic logic that adds the ability to quantify over global states and refer to global state components. We then present a semantic model for the environments in which agents choose their actions. Building on this model, we show how to construct a model for temporal epistemic logic called *strategy space* in which runs build in information about the strategy being used by each of the agents. We then define a spectrum of logics defined over the resulting semantics. These logics are obtained as fragments of the extended temporal epistemic logic, interpreted in strategy space. Section 3 deals with applications of the resulting logics. In particular, we show that the logics can express reasoning about implementations of knowledge-based programs, many notions that have been proposed in the area of alternating temporal epistemic logic, game-theoretic solution concepts, and problems from computer security. Next, in Section 4, we provide results on the complexity of the model-checking problem for the various fragments of the logic, identifying fragments with lower complexity than the general problem. In Section 5, we conclude with a discussion of related literature.

2 AN EXTENDED TEMPORAL EPISTEMIC LOGIC

The usual *interpreted systems* semantics for temporal epistemic logic [19] deals with runs, in which each moment of time is associated with a global state that is composed of a local state for each agent in the system. We begin by defining the syntax and semantics of an extension of temporal epistemic logic that adds the ability to quantify over global states and refer to global state components. This syntax and semantics will be instantiated in what follows by taking some of the global state components to be the strategies being used by agents.

To quantify over global states, we extend temporal epistemic logic with a set of variables *Var*, a quantifier $\exists x$, and a construct $\mathsf{e}_i(x)$, where x is a variable. The formula $\exists x.\phi$ says, intuitively, that there exists in the system a global state x such that ϕ (a formula that may contain uses of the variable x) holds at the current point. The formula $\mathsf{e}_i(x)$ asserts the equality of the local states of agent i at the current point and in the global state x.

Let *Prop* be a set of atomic propositions and let *Ags* be a finite set of agent names, excluding the special name e, which we use to designate the environment in which the agents operate. We write Ags^+ for the set $\{e\} \cup Ags$. The language ETLK($Ags, Prop, Var$) (or just ETLK when the parameters are obvious) has syntax given by the grammar:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid A\phi \mid \bigcirc\phi \mid \phi_1 U \phi_2 \mid \exists x.\phi \mid \mathsf{e}_i(x) \mid D_G\phi \mid C_G\phi,$$

where $p \in Prop$, $x \in Var$, $i \in Ags^+$, and $G \subseteq Ags^+$. The construct $D_G\phi$ expresses that agents in G have distributed knowledge of ϕ, i.e., could deduce ϕ if they pooled their information, and $C_G\phi$ says that ϕ is common knowledge to group G. The temporal formulas $\bigcirc\phi$, $\phi_1 U \phi_2$, $A\phi$ have the same intuitive meanings as in the temporal logic CTL^* [17], i.e., $\bigcirc\phi$ says that ϕ holds at the next moment of time, $\phi_1 U \phi_2$ says that ϕ_1 holds until ϕ_2 does, and $A\phi$ says that ϕ holds in all possible evolutions from the present situation.

Other operators can be defined in the usual way, e.g., $\phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2)$, $\Diamond\phi = (true U \phi)$, which says that ϕ holds eventually, $\Box\phi = \neg\Diamond\neg\phi$, which says that ϕ always holds, $E\phi = \neg A \neg\phi$, which says that ϕ holds on some path from the current point, and so on. The universal form $\forall x.\phi = \neg\exists x.\neg\phi$ expresses that ϕ holds for all global states x that occur in the system. For an agent $i \in Ags^+$, we write $K_i\phi$ for $D_{\{i\}}\phi$—this expresses that agent i knows the fact ϕ. The notion of everyone in group G knowing ϕ can then be expressed as $E_G\phi = \bigwedge_{i \in G} K_i\phi$. We write $\mathsf{e}_G(x)$ for $\bigwedge_{i \in G} \mathsf{e}_i(x)$. This says that at the current point, the agents in G have the same local state as they do at the global state named by variable x.

We will be interested in a fragment of the logic that restricts the occurrence of the temporal operators to some simple patterns, in the style of the branching-time temporal logic CTL [15]. We write ECTL($Ags, Prop, Var$) (or just ECTL when the parameters are obvious) for the fragment of the language ETLK($Ags, Prop, Var$) in which the temporal operators occur only in the particular forms $A\bigcirc\phi$, $E\bigcirc\phi$, $A\phi_1 U \phi_2$, and $E\phi_1 U \phi_2$. In the context of temporal logic, these restrictions reduce the complexity of model checking from PSPACE to PTIME [15]. It is therefore interesting to study the impact on complexity of a similar restriction in the context of our additional operators.

The semantics of ETLK($Ags, Prop, Var$) builds straightforwardly on the following definitions used in the standard semantics for temporal epistemic logic [19]. Consider a system for a set of agents Ags. A *global state* is an element of the set $\mathcal{G} = \Pi_{i \in Ags^+} L_i$, where L_e is a set of states of the environment and L_i is a set of *local states* for each agent $i \in Ags$. A *run* is a mapping $r : \mathbb{N} \to \mathcal{G}$ giving a global state at each moment of time. For $n \leq m$, write $r[n \ldots m]$ for the sequence $r(n)r(n+1) \ldots r(m)$. We also write $r[n \ldots]$ for the infinite sequence $r(n)r(n+1) \ldots$. A *point* is a pair (r, m) consisting of a run r and a time $m \in \mathbb{N}$. An *interpreted system* is a pair $\mathcal{I} = (\mathcal{R}, \pi)$, where $\mathcal{R}$ is a set of runs and π is an *interpretation*, mapping each point (r, m) with $r \in \mathcal{R}$ to a subset of $Prop$. Elements of $\mathcal{R} \times \mathbb{N}$ are called the *points* of $\mathcal{I}$. For each $i \in Ags^+$, we write $r_i(m)$ for the corresponding component of $r(m)$ in L_i and then define an equivalence relation on points by $(r, m) \sim_i (r', m')$ if $r_i(m) = r'_i(m')$. We also define $\sim_G^D \equiv \cap_{i \in G} \sim_i$, and $\sim_G^E \equiv \cup_{i \in G} \sim_i$, and $\sim_G^C \equiv (\cup_{i \in G} \sim_i)^*$ for $G \subseteq Ags$, where $*$ denotes the reflexive transitive closure of a relation. We take $\sim_\emptyset^D$ to be the universal relation on points and (for the sake of preserving monotonicity of these relations in these degenerate cases) take $\sim_\emptyset^E$ and $\sim_\emptyset^C$ to be the identity relation.

To extend this semantic basis for temporal epistemic logic to a semantics for ETLK, we just need to add a construct that interprets variables as global states. A *context* for an interpreted system $\mathcal{I}$ is a mapping Γ from Var to global states occurring in $\mathcal{I}$, i.e., such that for all $x \in Var$ there exists a point (r, m) of $\mathcal{I}$ such that $\Gamma(x) = r(m)$. When g is a global state and $x \in V$, we write $\Gamma[g/x]$ for the context Γ' with $\Gamma'(x) = g$ and $\Gamma'(y) = \Gamma(y)$ for all variables $y \neq x$. The semantics of the language ETLK is given by a relation $\Gamma, \mathcal{I}, (r, m) \models \phi$, representing that formula ϕ holds at point (r, m) of the interpreted system $\mathcal{I}$, relative to context Γ. This is defined inductively on the structure of the formula ϕ, as follows:

- $\Gamma, \mathcal{I}, (r, m) \models p$ if $p \in \pi(r, m)$;
- $\Gamma, \mathcal{I}, (r, m) \models \neg\phi$ if not $\Gamma, \mathcal{I}, (r, m) \models \phi$;
- $\Gamma, \mathcal{I}, (r, m) \models \phi \wedge \psi$ if $\Gamma, \mathcal{I}, (r, m) \models \phi$ and $\Gamma, \mathcal{I}, (r, m) \models \psi$;
- $\Gamma, \mathcal{I}, (r, m) \models A\phi$ if $\Gamma, \mathcal{I}, (r', m) \models \phi$ for all $r' \in \mathcal{R}$ with $r[0 \ldots m] = r'[0 \ldots m]$;
- $\Gamma, \mathcal{I}, (r, m) \models \bigcirc\phi$ if $\Gamma, \mathcal{I}, (r, m+1) \models \phi$;
- $\Gamma, \mathcal{I}, (r, m) \models \phi U \psi$ if there exists $m' \geq m$ such that $\Gamma, \mathcal{I}, (r, m') \models \psi$ and $\Gamma, \mathcal{I}, (r, k) \models \phi$ for all k with $m \leq k < m'$;
- $\Gamma, \mathcal{I}, (r, m) \models \exists x.\phi$ if $\Gamma[r'(m')/x], \mathcal{I}, (r, m) \models \phi$ for some point (r', m') of $\mathcal{I}$;
- $\Gamma, \mathcal{I}, (r, m) \models \mathsf{e}_i(x)$ if $r_i(m) = \Gamma(x)_i$;
- $\Gamma, \mathcal{I}, (r, m) \models D_G\phi$ if $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all (r', m') such that $(r', m') \sim_G^D (r, m)$;
- $\Gamma, \mathcal{I}, (r, m) \models C_G\phi$ if $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all (r', m') such that $(r', m') \sim_G^C (r, m)$.

The definition is standard, except for the constructs $\exists x.\phi$ and $\mathsf{e}_i(x)$. The clause for the former says that $\exists x.\phi$ holds at a point (r, m) if there exists a global state $g = r'(m')$ such that ϕ holds at the point (r, m), provided we interpret x as referring to g. Note that it is required that g is attained at some point (r', m'), so actually occurs in the system $\mathcal{I}$. The clause for $\mathsf{e}_i(x)$ says that this holds at a point (r, m) if the local state of agent i, i.e., $r_i(m)$, is the same as the local state $\Gamma(x)_i$ of agent i at the global state $\Gamma(x)$ that interprets the variable x according to Γ.

We remark that these novel constructs introduce some redundancy, in that the set of epistemic operators D_G could be reduced to the "universal" operator $D_\emptyset$, since $D_G\phi \equiv \exists x.(\mathsf{e}_G(x) \wedge D_\emptyset(\mathsf{e}_G(x) \Rightarrow \phi))$. Evidently, given the syntactic complexity of this formulation, D_G remains a useful notation.

Example 2.1. As an example of a property that can be naturally expressed in ESL, but not in most standard temporal epistemic logics (e.g., ESL minus the operators $\exists x$ and $\mathsf{e}_i(x)$), consider information flow security properties in the spirit of *nondeducibility* [53]. Suppose that there are two agents Hi and Lo, representing two information security levels High and Low, respectively. The High level contains secrets that need to be protected from an attacker, represented by the Low level. Nondeducibility security properties, intuitively, assert that Lo always has no information about Hi. When the information that needs to be protected is represented in the local state of Hi, this means that Lo should always consider all local states of Hi possible. This can be expressed using the formula

$$\Box(\neg\exists x(K_{Lo}(\neg \mathsf{e}_{Hi}(x))).$$

Here, $K_{Lo}(\neg \mathsf{e}_{Hi}(x))$ expresses that Lo has some information about Hi, because there exists some local state of Hi that Lo is able to exclude, namely, the local state g_{Hi} where g is the global state denoted by x. By asserting that it is always the case that there does not exist such a state x whose Hi-local component Lo is able to exclude, we say that Lo never has information about Hi. Equivalently, pushing the outer negation inwards gives the form $\Box(\forall x(\neg K_{Lo}(\neg \mathsf{e}_{Hi}(x)))$ which says that Lo always considers all local states of Hi to be possible.

We remark that the operators $\exists x$ and $\mathsf{e}_i(x)$ may be eliminated from the above formula if the system $\mathcal{I}$ is known and has a sufficiently rich set of atomic propositions that each local state h of Hi is associated with a conjunction ϕ_h of literals that is true exactly at global states g with $g_{Hi} = h$. Let L_{Hi} be the set of local states of Hi. This gives the equivalence

$$\exists x(K_{Lo}(\neg \mathsf{e}_{Hi}(x))) \equiv \bigvee_{h \in L_{Hi}} K_{Hi}(\neg\phi_h),$$

which is valid in $\mathcal{I}$. However, if the system $\mathcal{I}$ over all systems, then no single formula of the logic without the operators $\exists x$ and $\mathsf{e}_i(x)$ can be equivalent to $\exists x(K_{Lo}(\neg \mathsf{e}_{Hi}(x)))$, because a fixed set of propositions cannot distinguish an arbitrarily large set of states.

2.1 Strategic Environments

To semantically represent settings in which agents operate by strategically choosing their actions, we introduce *environments*, a type of transition system that models the available actions and their effects on the state. This modelling is long established in the literature on reasoning about knowledge [41] and is similar to models used in the tradition of alternating temporal logic [2]. From an environment and a class of strategies, we construct an instance of the interpreted systems semantics defined in the previous section. One of the innovations in this construction is to introduce new names that refer to global state components that represent the strategies being used by the agents.

An *environment* for agents Ags is a tuple $E = \langle S, I, \{Acts_i\}_{i \in Ags}, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$, where

(1) S is a set of states,
(2) I is a subset of S, representing the initial states,
(3) for each $i \in Ags$, component $Acts_i$ is a nonempty set of actions that may be performed by agent i; we define $Acts = \Pi_{i \in Ags} Acts_i$ to be the set of joint actions,
(4) $\rightarrow \subseteq S \times Acts \times S$ is a transition relation, labelled by joint actions,
(5) for each $i \in Ags$, component $O_i : S \rightarrow L_i$ is an observation function, and
(6) $\pi : S \rightarrow \mathcal{P}(Prop)$ is a propositional assignment.

Here, the range L_i of the observation function O_i is any set, and what will matter in the semantics is an equivalence relation derived from this function.

An environment is said to be finite if all its components, i.e., S, Ags, $Acts_i$, L_i, and $Prop$, are finite. Intuitively, a joint action $a \in Acts$ represents a choice of action $a_i \in Acts_i$ for each agent $i \in Ags$, performed simultaneously, and the transition relation resolves this into an effect on the state. We assume that $\rightarrow$ is serial in the sense that for all $s \in S$ and $a \in Acts$ there exists $t \in S$ such that $(s, a, t) \in \rightarrow$. We also write $s \xrightarrow{a} t$ for $(s, a, t) \in \rightarrow$.

Example 2.2. We describe an environment for a secure message transmission problem, which models a sender agent HS at a High security level that has a bit of information to be transmitted to a receiver agent HR, also at a High security level, via a channel represented by an agent Lo at the Low security level (e.g., the internet). The transmission is handled by an agent Cr that models cryptography that may be applied to the message before transmission. Thus, we take $Ags = \{HS, Cr, HR, Lo\}$. The environment has the following components:

- The set of states S is the set of assignments to the following variables:
  — $\mathtt{s}$, representing the sender's secret bit, with value in $\{0, 1\}$
  — $\mathtt{k}$, representing a secret encryption key, with value in $\{0, 1\}$
  — $\mathtt{c}$, representing the unsecured communication channel, with value in $\{0, 1, \bot\}$
  We represent a state in S in the format $\langle s, k, c\rangle$, corresponding to the values of the three variables.
- The set I of initial states is the set $\langle s, k, \bot\rangle$ where $s, k \in \{0, 1\}$. That is, the value of the channel $\mathtt{c}$ is initially $\bot$, representing that no message has yet been sent.
- We associate the following sets of actions with the agents: $Acts_{HS} = Acts_{HR} = Acts_{Lo} = \{\mathtt{skip}\}$ and $Acts_{Cr} = \{\mathtt{c} := \mathtt{s} \oplus \mathtt{k},\ \mathtt{c} := \mathtt{s} \oplus \overline{\mathtt{k}}\}$. Thus, agents HS, HR, Lo are inert; they can perform only the action $\mathtt{skip}$, which has no effect on their local states. The only active agent is Cr, which has two actions, each of which encrypts the message bit $\mathtt{s}$ using the key k and places the result in the channel $\mathtt{c}$. Encryption is done by computing the exclusive-or $\oplus$ of the message with information from the key. The two actions correspond to taking the information from the key to be either the key bit $\mathtt{k}$ itself or its complement $\overline{\mathtt{k}}$. Since agents HS, HR, Lo always perform skip, we may, for brevity, name joint actions using the action names for agent Cr, i.e., if a is one of Cr's actions, then we denote a joint action $\langle \mathtt{skip}, a, \mathtt{skip}, \mathtt{skip}\rangle$ in $Acts = Acts_{HS} \times Acts_{Cr} \times Acts_{HR} \times Acts_{Lo}$ as just a.
- The transition relation resolves joint actions denoted as $a \in Acts_{Cr}$ as follows:

$$\langle s, k, c\rangle \xrightarrow{a} \langle s', k', c'\rangle$$

  if either a is $\mathtt{c} := \mathtt{s} \oplus \mathtt{k}$ and $s' = s$, $k' = k$ and $c' = s \oplus k$, or a is $\mathtt{c} := \mathtt{s} \oplus \overline{\mathtt{k}}$ and $s' = s$, $k' = k$ and $c' = s \oplus \overline{k}$.
- We define the observation functions for each of the agents on states $\langle s, k, c\rangle \in S$ as follows:
  — Agent HS observes just the bit to be transmitted, i.e., $O_{HS}(\langle s, k, c\rangle) = s$.
  — Agent Cr observes both the bit to be transmitted and the value of the encryption key, i.e., $O_{HS}(\langle s, k, c\rangle) = \langle s, k\rangle$.
  — Agent HR observes the communication channel and the value of the encryption key, i.e., $O_{HR}(\langle s, k, c\rangle) = \langle k, c\rangle$.
  — Agent Lo observes just the communication channel, i.e., $O_{Lo}(\langle s, k, c\rangle) = c$.
- We do not need any propositions in our later uses of this environment, so we take $Prop = \emptyset$ and $\pi : S \to Prop$ to be the trivial assignment.

A *strategy* for agent $i \in Ags$ in an environment E is a function $\alpha_i : S \to \mathcal{P}(Acts_i) \setminus \{\emptyset\}$, selecting a nonempty set of actions of the agent at each state.[1] We call these actions *enabled* at the state for agent i. A *group strategy*, or *strategy profile*, for a group G is a tuple $\alpha_G = \langle \alpha_i\rangle_{i \in G}$ where each α_i is a strategy for agent i. A *joint* strategy is a group strategy for the group Ags of all agents. If $\alpha = \langle \alpha_i\rangle_{i \in G}$ is a group strategy for group G, and $H \subseteq G$, then we write $\alpha \upharpoonright H$ for the restriction $\langle \alpha_i\rangle_{i \in H}$ of α to H.

[1]More generally, a strategy could be a function of the history, but we focus here on strategies that depend only on the final state.

A strategy α_i for agent i is *deterministic* if $\alpha_i(s)$ is a singleton for all s. A strategy α_i for agent i is *uniform* if for all states s, t, if $O_i(s) = O_i(t)$, then $\alpha_i(s) = \alpha_i(t)$. Intuitively, uniformity captures the constraint that agents' actions are chosen using no more information than they obtain from their observations.[2] A strategy $\alpha_G = \langle \alpha_i \rangle_{i \in G}$ for a group G is *locally uniform (deterministic)* if α_i is uniform (respectively, deterministic) for each agent $i \in G$.[3] Given an environment E, we write $\Sigma^{det}(E)$ for the set of deterministic joint strategies, $\Sigma^{unif}(E)$ for the set of all locally uniform joint strategies, and $\Sigma^{unif,det}(E)$ for the set of all deterministic locally uniform joint strategies.

Example 2.3. We present some joint strategies in the environment of Example 2.2. For agents $i \in \{HS, HR, Lo\}$, the only available action is $\mathtt{skip}$, so all joint strategies α have $\alpha_i(s) = \{\mathtt{skip}\}$ for all $s \in S$. Thus, each joint strategy α is determined by its component α_{Cr}, the strategy of the encryption agent.

The encryption agent could always choose the action $\mathtt{c} := \mathtt{s} \oplus \mathtt{k}$, giving the strategy α^0_{Cr} defined by $\alpha^0_{Cr}(s) = \{\mathtt{c} := \mathtt{s} \oplus \mathtt{k}\}$ for all states s. This strategy is both locally uniform and deterministic.

If the encryption agent chooses its action non-deterministically, then we have the strategy α^1_{Cr} defined by $\alpha^1_{Cr}(s) = Acts_{Cr}$ for all states s. This strategy is locally uniform but not deterministic.

An alternate strategy for the encryption agent is to choose its action based on the values it observes. Consider the strategy α^2_{Cr} defined by letting $\alpha^2_{Cr}(\langle s, k, c \rangle)$ be the singleton set $\{\mathtt{c} := \mathtt{s} \oplus \mathtt{k}\}$ if $k = 0$ and the action $\{\mathtt{c} := \mathtt{s} \oplus \overline{\mathtt{k}}\}$ otherwise. This strategy is deterministic. Also, since the value k is always part of the agent's observation, this strategy is locally uniform.

2.2 Strategy Space

We now define an interpreted system, called the *strategy space* of an environment, that contains all the possible runs generated when agents Ags behave by choosing a strategy from some set Σ of joint strategies in the context of an environment E. To enable reference to the strategy being used by agent $i \in Ags$, we introduce the notation "$\sigma(i)$" as a name referring to agent i's strategy. For $G \subseteq Ags$, we write $\sigma(G)$ for the set $\{\sigma(i) \mid i \in G\}$.

Technically, $\sigma(i)$ will be treated as if it were an agent in the context of temporal epistemic logic, in the sense that it will be the index of a local state component of the global state. In particular, we take the value of the local state at index $\sigma(i)$ to be the strategy in use by agent i. We will permit use of the indices $\sigma(i)$ in epistemic operators. This provides a way to refer, using distributed knowledge operators D_G, where G contains the strategic indices $\sigma(i)$, to what agents would know should they take into account not just their own observations but also information about other agents' strategies. For example, the distributed knowledge operator $D_{\{i,\sigma(i)\}}$ captures the knowledge that agent i has, taking into account the strategy that it is running. Operator $D_{\{i,\sigma(i),\sigma(j)\}}$ captures what agent i would know, taking into account its own strategy and the strategy being used by agent j. Various applications of the usefulness of this expressiveness are given in Section 3.

We note, however, that unlike the base agent $i \in Ags$, the index $\sigma(i)$ is not one of the agents in the environment E, and it is not associated with any actions. The index $\sigma(i)$ exists only in the interpreted system that we generate from E. (A similar remark applies to the special agent e, which is also not associated with any actions.) Since the indices $\sigma(i)$ are not agents in the same sense as agents $i \in Ags$, the reader may prefer to read $D_G\phi$ with $\sigma(i) \in G$ as "ϕ is *deducible* from the

[2]Recall that we work in this article with agents with imperfect recall. For agents with perfect recall, we would use a notion of uniformity that allows agents choice of action to depend on all their past observations.

[3]We prefer the term "locally uniform" to just "uniform" in the case of groups, since we could say a strategy α for group G is *globally uniform* if for all states s, t, if $O_i(s) = O_i(t)$ for all $i \in G$, then $\alpha_i(s) = \alpha_i(t)$ for all $i \in G$. While we do not pursue this in the present article, this notion would be interesting in settings where agents share information to collude on their choice of move.

information contained in state components G" rather than the more standard "it is distributed knowledge to agents G that ϕ."

Formally, suppose we are given an environment $E = \langle S, I, \{Acts_i\}_{i \in Ags}, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$ for agents Ags, where $O_i : S \rightarrow L_i$ for each $i \in Ags$, and a set $\Sigma \subseteq \Pi_{i \in Ags} \Sigma_i$ of joint strategies for the group Ags. We define the *strategy space* interpreted system $\mathcal{I}(E, \Sigma) = (\mathcal{R}, \pi')$ as follows.[4] The system $\mathcal{I}(E, \Sigma)$ has global states $\mathcal{G} = S \times \Pi_{i \in Ags} L_i \times \Pi_{i \in Ags} \Sigma_i$. Intuitively, each global state consists of a state of the environment E, a local state for each agent i in E, and a strategy for each agent i. We index the components of this Cartesian product by e, the elements of Ags and the elements of $\sigma(Ags)$, respectively. We take the set of runs $\mathcal{R}$ of $\mathcal{I}(E, \Sigma)$ to be the set of all runs $r : \mathbb{N} \rightarrow \mathcal{G}$ satisfying the following constraints, for all $m \in \mathbb{N}$ and $i \in Ags$:

(1) $r_e(0) \in I$ and $\langle r_{\sigma(i)}(0) \rangle_{i \in Ags} \in \Sigma$,
(2) $r_i(m) = O_i(r_e(m))$,
(3) $(r_e(m), a, r_e(m+1)) \in \rightarrow$ for some joint action $a \in Acts$ such that for all $j \in Ags$ we have $a_j \in \alpha_j(r_e(m))$, where $\alpha_j = r_{\sigma(j)}(m)$, and
(4) $r_{\sigma(i)}(m+1) = r_{\sigma(i)}(m)$.

The interpretation π' of $\mathcal{I}(E, \Sigma)$ is determined from the interpretation π of E by taking $\pi'(r, m) = \pi(r_e(m))$ for all points (r, m).

The first constraint on runs says, intuitively, that runs start at an initial state of E, and the initial strategy profile at time 0 is one of the profiles in Σ. The second constraint states that the agent i's local state at time m is the observation that agent i makes of the state of the environment at time m. The third constraint says that evolution of the state of the environment is determined at each moment of time by agents choosing an action by applying their strategy at that time to the state at that time. The joint action resulting from these individual choices is then resolved into a transition on the state of the environment using the transition relation from E. The final constraint says that agents' strategies are fixed during the course of a run. Intuitively, each agent picks a strategy and then sticks to it.

Our epistemic strategy logic is now just an instantiation of the extended temporal epistemic logic in the strategy space generated by an environment. That is, we start with an environment E and an associated set of strategies Σ and then work with the language ETLK$(Ags \cup \sigma(Ags), Prop, Var)$ in the interpreted system $\mathcal{I}(E, \Sigma)$. (Recall that this notation implicitly includes a local state component e to represent the state of the environment.) We call this instance of the language ESL$(Ags, Prop, Var)$ or just ESL when the parameters are implicit.

Since interpreted systems are always infinite objects, we use environments to give a finite input for the model-checking problem. For an environment E, a set of strategies Σ for E, and a context Γ for $\mathcal{I}(E, \Sigma)$, we write $\Gamma, E, \Sigma \models \phi$ if $\Gamma, \mathcal{I}(E, \Sigma), (r, 0) \models \phi$ for all runs r of $\mathcal{I}(E, \Sigma)$. Often, the formula ϕ will be a sentence, i.e., will have all variables x in the scope of an operator $\exists x$. In this case the statement $\Gamma, E, \Sigma \models \phi$ is independent of Γ and we write simply $E, \Sigma \models \phi$.

We will be interested in a number of fragments of ESL that turn out to have lower complexity. We define ESL$^-(Ags, Prop, Var)$, or just ESL$^-$, to be the language

$$\text{ECTL}(Ags \cup \sigma(Ags), Prop, Var).$$

[4] The construction given here is for an "observational" or "imperfect recall" modelling of knowledge that assumes that an agent reasons, and chooses its next action, on the basis of its current observation only. It is straightforward to give other constructions such as a synchronous perfect recall semantics, where we work with the sequence of observations and actions of the agent instead. Model checking for such a variant would be undecidable, so we do not pursue this here.

Another fragment of the language that will be of interest is the language, denoted

$$\mathrm{CTL}^*\mathrm{K}(Ags \cup \sigma(Ags), Prop, Var),$$

in which we omit the constructs $\exists$ and $\mathsf{e}_i(x)$; this is a standard branching-time temporal epistemic language except that it contains the strategy indices $\sigma(Ags)$.

3 APPLICATIONS

We now consider a range of applications of the logic ESL and show how it can represent notions from earlier work on alternating temporal epistemic logic. (In a few cases, we prove precise translation results, but due to the large number of operators and distinct semantics underlying these logics in the literature, we just sketch intuitive correspondences in most cases.)

3.1 Variants of Nondeducibility

We already mentioned the notion of nondeducibility in Example 2.1, which shows one way that our logic extends the expressiveness of previous work on temporal epistemic logic by allowing quantification over agents' local states to be expressed. We continue discussion of this example here in the context of the environment E of Example 2.2. We also show that our logic can represent a related notion from the security literature called *nondeducibility on strategies* [57] that involves an agent reasoning not just based on its local state but also using knowledge of the strategy being employed by another agent. This demonstrates a further dimension in which we can express more than prior work on alternating temporal epistemic logic and shows the value of allowing the strategic indices $\sigma(i)$ to occur in epistemic operators. (Our discussion in this section loosely follows examples used in Reference [57] to motivate nondeducibility on strategies.)

Consider first the instance

$$NonDed = \Box(\neg\exists x.(K_{Lo}(\neg \mathsf{e}_{HS}(x)))$$

of the formula from Example 2.1, which expresses that the low-level attacker Lo never learns any information about the high-level secret held in the local state of the high sender HS. However, the formula

$$Ded(G) = \Diamond(\exists x.(D_G(\mathsf{e}_{HS}(x)))$$

states that group G does eventually learn the value of the secret held by HS. (Note that the formula $D_G(\mathsf{e}_{HS}(x))$ says that the group G has distributed knowledge that the local state of component HS is the same as the local state of HS in the global state denoted by variable x. The formulas $NonDed$ and $Ded(\{Lo\})$ are not opposites, as one might expect from the names. Actually, the negation of $NonDed$ and $Ded(\{Lo\})$ lead to similar formulas, except that the former has a negation before $\mathsf{e}_{HS}(x)$.) Clearly, for cryptography to be effective, we require that the specification $NonDed \wedge Ded(\{HR\})$ be satisfied, which expresses that the High receiver HR eventually learns the secret, but that the adversary Lo never has any information about the secret.

In what follows, given a joint strategy α, we write $\Sigma(\alpha)$ for the singleton set of strategies $\{\alpha\}$.

Suppose first that encryption is always done using the action $\mathsf{c} := \mathsf{s} \oplus \mathsf{k}$, so that the joint strategy is the strategy α^0 from Example 2.3, with $\alpha^0_{Cr}(s) = \{\mathsf{c} := \mathsf{s} \oplus \mathsf{k}\}$ for all states s. Then we work in the interpreted system generated by the set of strategies $\Sigma(\alpha^0) = \{\alpha^0\}$. Note that in $\mathcal{I}(E, \Sigma(\alpha^0))$, it is common knowledge that the strategy being used by Cr is α^0_{Cr}. The following result shows that in this case, the system satisfies the specification $NonDed \wedge Ded(\{HR\})$.

Proposition 1. $E, \Sigma(\alpha^0) \models NonDed \wedge Ded(\{HR\})$.

Proof. We first show that $E, \Sigma(\alpha^0) \models NonDed$. Note that, since there is only one joint strategy, and agents' observations are derived from the state of the environment, a run r of $\mathcal{I}(E, \Sigma(\alpha^0))$

is determined by the sequence of states of the environment $r_e[0\ldots] = r_e(0), r_e(1)\ldots$. These sequences all have the form

$$\langle s, k, \bot\rangle\langle s, k, s \oplus k\rangle^\infty$$

for some $s, k \in \{0, 1\}$, where t^∞ indicates infinitely many copies of the state t. For each run r of this form, there exists another run r' with $r'_e[0\ldots] = \langle\overline{s}, \overline{k}, \bot\rangle\langle\overline{s}, \overline{k}, \overline{s} \oplus \overline{k}\rangle^\infty$. Now, we have that $(r, n) \sim_{Lo} (r', n)$ for all $n \in \mathbb{N}$, since

$$r_{Lo}(0) = O_{Lo}(\langle s, k, \bot\rangle) = \bot = O_{Lo}(\langle\overline{s}, \overline{k}, \bot\rangle) = r'_{Lo}(0)$$

and

$$\begin{aligned} r_{Lo}(n) &= O_{Lo}(\langle s, k, s \oplus k\rangle) \\ &= s \oplus k \\ &= \overline{s} \oplus \overline{k} \\ &= O_{Lo}(\langle\overline{s}, \overline{k}, \overline{s} \oplus \overline{k}\rangle) \\ &= r'_{Lo}(n) \end{aligned}$$

for $n \geq 1$. Since also $(r, n) \sim_{Lo} (r, n)$, we have that *Lo* considers both possible values of the local state of agent *HS* possible, so $\mathcal{I}(E, \Sigma(\alpha^0)), (r, 0) \models NonDed$.

However, we have $\mathcal{I}(E, \Sigma(\alpha^0)), (r, 0) \models Ded(\{HR\})$. For, at time 1, we have $r_{HR}(1) = O_{HR}(\langle s, k, s \oplus k\rangle) = \langle k, s \oplus k\rangle$. Let (r', m) be any point with $(r, 1) \sim_{HR} (r', m)$, and let $r'_e[0\ldots] = \langle s', k', \bot\rangle\langle s', k', s' \oplus k'\rangle^\infty$. Then $m \geq 1$ and

$$r'_{HR}(m) = O_{HR}(\langle s', k', s' \oplus k'\rangle) = \langle k', s' \oplus k'\rangle.$$

Thus, from $r_{HR}(1) = r'_{HR}(m)$, we obtain $k = k'$ and $s \oplus k = s' \oplus k'$. Hence also $r'_{HS}(m) = s' = (s' \oplus k') \oplus k' = (s \oplus k) \oplus k = s = r_{HS}(1)$. This shows that $\mathcal{I}(E, \Sigma(\alpha^0)), (r, 1) \models \exists x(K_{HR}(\mathrm{e}_{HS}(x)))$, so $\mathcal{I}(E, \Sigma(\alpha^0)), (r, 0) \models Ded(\{HR\}))$. □

However, not every strategy for the encryption agent similarly satisfies the specification. Consider the joint strategy α^2 from Example 2.3. Here we have that *Lo* and *HR* both always learn the value of the secret.

Proposition 2. $E, \Sigma(\alpha^2) \models Ded(\{HR\}) \wedge Ded(\{Lo\})$.

Proof. Strategy α^2 is deterministic. Note that if $k = 0$, then $s \oplus k = s$, and if $k = 1$, then $s \oplus \overline{k} = s$. Thus, the runs of α^2 have sequence of environment states $r_e[0\ldots] = \langle s, k, \bot\rangle\langle s, k, s\rangle^\infty$. As above, since $\Sigma(\alpha^2)$ is a singleton, this sequence determines the run as a whole. Since $O_{Lo}(\langle s, k, s\rangle) = s$ and $O_{HR}(\langle s, k, s\rangle) = \langle k, s\rangle$, both *Lo* and *HR* directly observe the value of the secret s in the local state of *HS* from time 1, so know this value. □

A corollary of this result is that if we work in a system where all (uniform) strategies for *Cr* are possible (represented by the set of strategies Σ^{unif}), then while *Lo* cannot deduce the secret in general, there are encryption strategies for *Cr* such that, if *Lo* knew that this strategy is being applied by *Cr*, then *Lo* would be able to deduce the secret.

Proposition 3. *$E, \Sigma^{unif} \models NonDed$, but not $E, \Sigma^{unif} \models \neg Ded(\{Lo, \sigma(Cr)\})$.*

Proof. For $E, \Sigma^{unif} \models NonDed$, we note that *Lo* always considers it possible that *Cr* is running strategy α^0 from above and argue exactly as in Proposition 1. To show that not $E, \Sigma^{unif} \models \neg Ded(\{Lo, \sigma(Cr)\})$, let r be a run in which *Cr* runs strategy α^2_{Cr}. Note that if $(r, 0) \sim_{\{Lo, \sigma(Cr)\}} (r', m)$, then $r_{\sigma(Cr)}(0) = r'_{\sigma(Cr)}(m)$, i.e., *Cr* uses the same strategy in the runs r and r'. Essentially the same argument as applied in Proposition 2 to show that $Ded(\{Lo\})$ holds then shows that $\mathcal{I}(E, \Sigma^{unif}), (r, 0) \models Ded(\{Lo, \sigma(Cr)\})$. □

By means of a similar example, Wittbold and Johnson [57] argued that nondeducibility is too weak a notion of security to capture information flow security attacks in which the attacker exploits a covert channel in a system. Intuitively, it does not take into account that the attacker may have information about the strategies being used by other agents. One example of how such knowledge of another agent's strategy may arise in practice is when the attacker *Lo* has succeeded in infiltrating a virus (here represented by the strategy of *Cr*) into the system being attacked (here composed of components *HS*, *HR*, and *Cr*, i.e., the High sender, the High receiver, and the encryption agent, respectively). When this is the case, a more appropriate modality for the attacker's knowledge is the modality $D_{\{Lo,\sigma(Cr)\}}$, which captures what *Lo* can deduce when it also knows the strategy $\sigma(Cr)$ being employed by *Cr* rather than the modality $D_{\{Lo\}}$ used in $Ded(\{Lo\})$. (The modality $D_{\{Lo,\sigma(Lo),\sigma(Cr)\}}$ that says that *Lo* also reasons knowing its own strategy would also make sense in general, though in the model under discussion it is identical to $D_{\{Lo,\sigma(Cr)\}}$, since *Lo* has only one action to choose from, so all its uniform strategies are the same.) Wittbold and Johnson's notion of *nondeducibility on strategies* (NDS) is a definition of security that takes into account such reasoning by the attacker. For a two-agent system, composed of Low-level agent *Lo* and High-level agent Hi, Wittbold and Johnson define a system to satisfy non-deducibility on strategies if every Low view is compatible with every High strategy. NDS may be expressed directly in our logic by the formula[5]

$$D_\emptyset \forall x.(\neg K_{Lo}(\neg \mathsf{e}_{\sigma(Hi)}(x))),$$

which says that at all points of the system (identifying a *Lo* view/local state, in particular) for all global states x (identifying a High strategy, in particular), *Lo* considers the High strategy in x to be possible. This notion cannot be expressed in alternating temporal epistemic logics such as ATEL, discussed below, which do not allow reference to what can be deduced about other agents' strategies.

3.2 Revocable and Irrevocable Strategies in ATL

ATL [2] is a generalization of the branching-time temporal logic CTL that can express the capability of agents' strategies to bring about temporal effects. We show in this section that ESL is able to express several variants of ATL. The following section relates various epistemic extensions of ATL to ESL.

The syntax of ATL formulas ϕ is given as follows:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \langle\langle G\rangle\rangle \bigcirc \phi \mid \langle\langle G\rangle\rangle \Box \phi \mid \langle\langle G\rangle\rangle (\phi_1 U \phi_2),$$

where $p \in Prop$, $i \in Ags$, and $G \subseteq Ags$. Essentially, each branching construct $A\phi$ of CTL is generalized in ATL to an *alternating* construct $\langle\langle G\rangle\rangle\phi$ for a group G of agents, where ϕ is a "prefix temporal" formula such as $\bigcirc\phi'$, $\Diamond\phi'$, $\Box\phi'$, or $\phi_1 U \phi_2$, as would be used to construct a CTL formula. Intuitively, $\langle\langle G\rangle\rangle\phi$ says that the group G has a strategy for ensuring that ϕ holds, irrespective of what the other agents do.

The semantics of ATL is given using *concurrent game structures*, which are very similar to environments as defined above, with the main differences being the following. For each point of difference, we sketch how to view concurrent game structures as equivalent to environments.

[5]The perfect recall semantics in combination with perfect recall strategies would give the interpretation of this formula that is most adequate for security applications.

- Concurrent game structures lack a set of initial states. It is convenient for technical reasons to treat a concurrent game structure as an environment with all of its states initial.
- Concurrent game structures allow that not all actions are available at every state, whereas in environments all actions are always available. In environments, we can treat a choice of a non-enabled action as equivalent to a choice of a default enabled action in the transition relation.
- The transition relation in concurrent game structures is deterministic, in the sense that for each state s and joint action a, there exists a unique state t such that $s \xrightarrow{a} t$. Nondeterminism in environments can be modelled in concurrent game structures by adding an agent that makes the nondeterministic choice through its actions.
- ATL's concurrent game structures do not have a notion of observation. Intuitively, all agents always have perfect information concerning the current state. We may capture this in environments by taking $O_i(s) = s$ for all agents i and states s.

Using such correspondences, we can express the ATL semantics in environments E as follows. For reasons discussed below, we generalize the ATL semantics by parameterizing the definition on a set Δ of strategies for groups of agents in the environment E. That is, Δ is a collection of tuples of agent strategies of the form $\langle \alpha_i \rangle_{i \in G}$, with both the strategies α_i and the set G of agents varying. The semantics uses a relation $E, s \models^\Delta \phi$, where $E = \langle S, I, Acts, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$ is an environment and $s \in S$ is a state of E, and ϕ is a formula.

For the definition, we need the notion of a path in E: This is a function $\rho : \mathbb{N} \rightarrow S$ such that for all $k \in \mathbb{N}$ there exists a joint action a with $(\rho(k), a, \rho(k+1)) \in \rightarrow$. A path ρ is *from* a state s if $\rho(0) = s$. A path ρ is *consistent* with a strategy $\alpha = \langle \alpha_i \rangle_{i \in G}$ for a group G if for all $k \in \mathbb{N}$ there exists a joint action a such that $(\rho(k), a, \rho(k+1)) \in \rightarrow$ and $a_i \in \alpha_i(\rho(k))$ for all $i \in G$. It is also convenient to identify the *path formulas* of ATL as formulas of the form $\bigcirc\phi$, $\Box\phi$ or $\phi U \psi$ where ϕ and ψ are ATL formulas.

The relation $E, s \models^\Delta \phi$, where s is a state of E and ϕ is an ATL formula, is defined by a mutual recursion with the relation $E, \rho \models^\Delta \phi$, where ρ is a path of E and ϕ is a path formula, as follows. Note that if $\langle\!\langle G \rangle\!\rangle\phi$ is an ATL formula, then ϕ is a path formula. For evaluation of ATL formulas at a state we have the clauses

- $E, s \models^\Delta p$ if $p \in \pi(s)$;
- $E, s \models^\Delta \neg\phi$ if not $E, s \models^\Delta \phi$;
- $E, s \models^\Delta \phi \wedge \psi$ if $E, s \models^\Delta \phi$ and $E, s \models^\Delta \psi$;
- $E, s \models^\Delta \langle\!\langle G \rangle\!\rangle\phi$ if there exists a strategy $\alpha_G \in \Delta$ for group G such that for all paths ρ from s that are consistent with α_G, we have $E, \rho \models^\Delta \phi$;

and for evaluation of a path formula at a path we have the clauses

- $E, \rho \models^\Delta \bigcirc\phi$ if $E, \rho(1) \models^\Delta \phi$;
- $E, \rho \models^\Delta \Box\phi$ if have $E, \rho(k) \models^\Delta \phi$ for all $k \in \mathbb{N}$;
- $E, \rho \models^\Delta \phi U \psi$ if there exists $m \geq 0$ such that $E, \rho(m) \models^\Delta \psi$, and for all $k < m$, we have $E, \rho(k) \models^\Delta \phi$.

The semantics for ATL given in Reference [2] corresponds to the instance of this definition with Δ equal to the set of perfect recall, perfect information group strategies, but we focus here on the variant where Δ contains just imperfect information strategies.

We argue that the ATL construct $\langle\!\langle G \rangle\!\rangle\phi$ can be expressed in CTL*K($Prop, Ags \cup \sigma(Ags)$) as

$$\neg K_e \neg D_{\{e\} \cup \sigma(G)} \phi.$$

Intuitively, here the outer operator $\neg K_e \neg$ existentially switches to a point that has the same state of the environment as the current state (and hence the same local state for all agents in *Ags*) but may have different strategies for any of the agents. The inner operator $D_{\{e\} \cup \sigma(G)}$ then fixes both the state of the environment and the strategies selected by the group G but allows all other agents to vary their strategy. It quantifies universally over these possibilities. Thus, the formula says that the group G has a strategy that achieves ϕ from the current state, whatever strategy the other agents play. (An alternate way to express the formula using the richer expressive power of ESL is as $\exists x (K_e(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi))$.)

More formally, consider the following translation from ATL to CTL*K($Prop, Ags \cup \sigma(Ags)$). For an ATL formula ϕ, we write ϕ^* for the translation of ϕ, defined inductively on the construction of ϕ by the following rules:

$$
\begin{aligned}
&p^* = p\\
&(\neg\phi)^* = \neg\phi^*\\
&(\phi_1 \wedge \phi_2)^* = \phi_1^* \wedge \phi_2^*\\
&(\langle\langle G\rangle\rangle\phi)^* = \neg K_e \neg D_{\{e\}\cup\sigma(G)}\phi^*\\
&(\bigcirc\phi)^* = \bigcirc\phi^*\\
&(\Box\phi)^* = \Box\phi^*\\
&(\phi_1 U \phi_2)^* = \phi_1^* U \phi_2^*.
\end{aligned}
$$

Note that the semantics of the operators using $\langle\langle G\rangle\rangle$ quantifies over runs in which the agents G run a particular strategy α_G, but there is no constraint on the behaviour of the other agents: These are not assumed to choose their actions according to any particular strategy. A natural alternative to the definition above would be to use the clause

> $E, s \models^{\Delta} \langle\langle G\rangle\rangle\phi$ if there exists a strategy $\alpha \in \Delta$ for group G such that for all joint strategies $\beta \in \Delta$ for group *Ags* with $\beta \upharpoonright G = \alpha$, and all paths ρ from s that are consistent with β, we have $E, \rho \models^{\Delta} \phi$.

This variant corresponds more directly to the formula $\neg D_e \neg D_{\{e\}\cup\sigma(G)} \Box\phi$ than does the ATL semantics. It is reasonable to take the position that it more naturally captures a concept of interest in competitive situations where agents are constrained in the strategies they are able to use.

In the original semantics of ATL, where perfect information, perfect recall strategies were considered, the two definitions are equivalent, since for any behaviour of the other agents, there is a strategy that matches it. However, for the imperfect information, epistemic extension we consider, this does not hold. For example, if all strategies in Δ are deterministic, then the above variant would not allow paths in which some agent in the complement of G chooses an action a at the first occurrence of a state s, but some other action b at a later occurrence of s. However, such runs are allowed in the ATL semantics given above. Since the semantics of ESL assumes that all runs are generated by all agents running some strategy, we need to make some technical assumptions on Δ to set up a correspondence with ATL.

Define the "random" strategy for agent i to be the strategy $rand_i$ defined by $rand_i(s) = Acts_i$ for all states $s \in S$. Given a strategy $\alpha = \langle\alpha_i\rangle_{i\in G}$ for a group of agents G in an environment E, define the *completion* of the strategy to be the joint strategy $comp(\alpha) = \langle\alpha'_i\rangle_{i\in Ags}$ with $\alpha'_i = \alpha_i$ for $i \in G$ and with $\alpha'_i = rand_i$ for all $i \in Ags \setminus G$. Intuitively, this operation completes the group strategy to a joint strategy for all agents, by adding the "random" strategy for all agents not in G, so that these agents are completely unconstrained in their behaviour. Given a set of strategies Δ for groups of agents, we define the set of joint strategies $comp(\Delta) = \{comp(\alpha) \mid \alpha \in \Sigma\}$.

A second technicality is needed that results from the way we have used Δ as a parameter in a generalization of the ATL semantics. A constraint on this set is needed to prove our correspondence result. Say that a set Δ of group strategies is *restrictable* if for every $\alpha \in \Delta$ for group of agents G and every group $H \subseteq G$, the restriction $\alpha \restriction H$ of α to agents in H is also in Δ. Say that Δ is *extendable* if for every strategy α for a group H and group $G \supseteq H$, there exists a strategy $\alpha' \in \Delta$ for group G whose restriction $\alpha' \restriction H$ to H is equal to α. Intuitively, restrictability says that group strategies are closed under formation of subgroups, and extensibility says that a group is not able to prevent any other agent from having *some* strategy that they are able to follow at the same time as the group follows its choice of strategy.

The requirement that a set Δ of group strategies be restrictable and extendable is quite mild. For example, if Δ_i is a set of strategies for agent i, for each agent $i \in Ags$, then the natural set of "Cartesian product strategies"

$$\Delta = \{\langle \alpha_i \rangle_{i \in G} \mid G \subseteq Ags, \forall i \in G\, (\alpha_i \in \Delta_i)\}$$

is both restrictable and extendable. In particular, the set of all group strategies, and the set of all locally uniform group strategies, are both restrictable and extendable. Another example of a collection of strategies satisfying this condition is the set of group strategies α in which at most k agents follow a strategy that differs from a designated "correct" strategy σ. Note that this collection is extendable, because an agent always has the option to choose the correct strategy, even if k others have already deviated. This collection models a common assumption in the analysis of fault-tolerant distributed algorithms.

A final technicality relates to the fact that whereas runs of an environment start at an initial state of the environment, and hence an environment may have unreachable states, models in the ATL semantics lack a notion of initial state, and formulas may be evaluated at any state. As already noted above, we resolve this difference by viewing ATL models as environments in which all states are initial (hence reachable).

The following result now captures in a precise way that the ATL semantics can be expressed in our logic as claimed above, provided we allow joint strategies in which some agents run the random strategy.

Theorem 3.1. *For every environment E in which all states are initial, for every nonempty set of group strategies Δ that is restrictable and extendable, for every state s of E and ATL formula ϕ, we have $E, s \models^{\Delta} \phi$ iff for all (equivalently, some) points (r, m) of $\mathcal{I}(E, comp(\Delta))$ with $r_e(m) = s$ we have $\mathcal{I}(E, comp(\Delta)), (r, m) \models \phi^*$.*

Proof. For brevity, we write just $\mathcal{I}$ for $\mathcal{I}(E, comp(\Delta))$. For the claim that the quantifiers "for all" and "some" are interchangeable in the right-hand side, note that formulas of the form ϕ^* are Boolean combinations of atomic propositions and formulas of the form $K_e\psi$, whose semantics at a point (r, m) depends only on $r_e(m)$. This gives the implication from the "some" case to the "for all" case. For the implication from the "for all" case to the "some" case, note that the "for all" case is never trivial, because for all states s of E, there exists a point (r, m) of $\mathcal{I}$ with $r_e(m) = s$. This follows from the fact that all states are initial in E and that the transition relation is serial, so that any group strategy α in Δ is consistent with an infinite path from initial state s. This corresponds to a run r with $r(0) = (s, comp(\alpha))$.

It therefore suffices to show that $E, s \models^{\Delta} \phi$ iff for all points (r, m) of $\mathcal{I}$ with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \phi^*$. Additionally, for path subformulas ϕ of the form $\bigcirc\psi$, $\Box\psi$ and $\psi_1 U \psi_2$ of ATL formulas, we show that for all paths ρ, we have $E, \rho \models^{\Delta} \phi$ iff for all points (r, m) of $\mathcal{I}$ with $r_e[m \ldots] = \rho$ we have $\mathcal{I}, (r, m) \models \phi^*$

We proceed by induction on the construction of ϕ. The base case of atomic propositions, as well as the cases for the Boolean constructs, are trivial. The claim concerning path formulas is also straightforward from the semantics of the temporal operators and, inductively, the claim concerning state formulas.

We consider next the case of $\phi = \langle\langle G\rangle\rangle\psi$. We show that $E, s \models^\Delta \langle\langle G\rangle\rangle\psi$ iff for all points (r, m) of $\mathcal{I}$ with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\}\cup\sigma(G)}\phi^*$.

Suppose, first, $E, s \models^\Delta \langle\langle G\rangle\rangle\psi$. Let (r, m) be a point of $\mathcal{I}$ with $r_e(m) = s$. We show that $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\}\cup\sigma(G)}\psi^*$. By the ATL semantics, there exists a strategy $\alpha_G \in \Delta$ for group G such that for all paths ρ of E from s that are consistent with α_G we have $E, \rho \models^\Delta \psi$. Let $\alpha = comp(\alpha_G)$ (note that this is in $comp(\Delta)$), and (using the fact that all states are initial) let r' be a run of $\mathcal{I}$ with $r'(0) = (s, \alpha)$. Because $r_e(m) = s = r'_e(0)$, we have $(r, m) \sim_e (r', 0)$, and it suffices to show that $\mathcal{I}, (r', 0) \models D_{\{e\}\cup\sigma(G)}\psi^*$. For this, suppose that (r'', m'') is any point of $\mathcal{I}$ with $(r', 0) \sim_{\{e\}\cup\sigma(G)} (r'', m'')$. We show that $\mathcal{I}, (r'', m'') \models \psi^*$. Now $r''(m'') = (t, \alpha')$ implies that $\alpha'_i = \alpha_i$ for all $i \in G$. Thus, the path $\rho = r''_e(m'')r''_e(m'' + 1)\ldots$ in E is consistent with α_G, and $\rho(0) = r''_e(m'') = r'_e(0) = s$. It follows that $E, \rho \models^\Delta \psi$. Using the induction hypothesis, it follows that $\mathcal{I}, (r'', m'') \models \psi^*$. This completes the argument that $\mathcal{I}, (r', 0) \models D_{\{e\}\cup\sigma(G)}\psi^*$.

Conversely, suppose that for all points (r, m) of $\mathcal{I}$ with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\}\cup\sigma(G)}\psi^*$. We show that $E, s \models^\Delta \langle\langle G\rangle\rangle\psi$. Using the fact that all states are initial, let r be a run of $\mathcal{I}$ with $r_e(0) = s$, and, hence, $\mathcal{I}, (r, 0) \models \neg K_e \neg D_{\{e\}\cup\sigma(G)}\psi^*$. Then there exists a point (r', m') of $\mathcal{I}$ such that $r'_e(m') = s$ and $\mathcal{I}, (r', m') \models D_{\{e\}\cup\sigma(G)}\psi^*$. Let $r'(m') = (s, \alpha)$. Then there exists a strategy $\beta \in \Delta$ for some set of agents G' such that $\alpha = comp(\beta) \in comp(\Delta)$. Let $H = G \cap G'$. By restrictability, we have $\beta \restriction H \in \Delta$. By extendability, there exists a strategy $\gamma \in \Delta$ for group G such that $\gamma \restriction H = \beta \restriction H$. Taking $\alpha' = comp(\beta \restriction H)$, it follows that $\alpha' \in comp(\Delta)$. Note that $\alpha' \restriction G = \alpha \restriction G$ and $\alpha'_i = rand_i$ for $i \in Ags \setminus G$. In particular, $\alpha'_i = rand_i$ for $i \in G \setminus H$. Thus, any path consistent with γ is consistent with α'.

To prove that $E, s \models^\Delta \langle\langle G\rangle\rangle\psi$, we show that for every path ρ of E from s consistent with γ, we have $E, \rho \models^\Delta \psi$. For this, let ρ be a path from s consistent with the strategy γ for group G. By the conclusion of the previous paragraph, ρ is consistent with the joint strategy α' for all agents. Since s is an initial state of E, there exists a run r'' of $\mathcal{I}$ with $r''(0) = (s, \alpha')$ and $r''_e[0\ldots\infty] = \rho$. Moreover, $(r', m') \sim_{\{e\}\cup\sigma(G)} (r'', 0)$. Thus, we obtain from $\mathcal{I}, (r', m') \models D_{\{e\}\cup\sigma(G)}\psi^*$ that $\mathcal{I}, (r'', 0) \models \psi^*$. By the induction hypothesis, we obtain that $E, \rho \models^\Delta \psi$. □

The ESL interpretation unpacks the alternating double quantification in the semantics of $\langle\langle G\rangle\rangle\phi$. ESL offers the advantage of being able to express notions that are not expressible in ATL. For example, under assumptions similar to those of Theorem 3.3,

$$\neg D_e \neg((\neg D_{\{e\}\cup\sigma(Ags)} \neg \Diamond p) \wedge (D_{\{e\}\cup\sigma(G)} \Box q))$$

says that, from the current state, there is a joint strategy for all agents, such that, some runs of this joint strategy satisfy $\Diamond p$, and group G's strategy alone suffices to ensure that $\Box q$.

There has been discussion in the literature on ATL about whether strategies should be *revocable* or *irrevocable*. Consider a formula such as

$$\langle\langle A\rangle\rangle \Box (p \wedge \langle\langle A, B\rangle\rangle \Diamond q).$$

This says that A has a strategy that ensures that it is always the case both that p holds and that A and B together have a strategy that ensures that eventually q. Under the ATL semantics, the strategy of A used to satisfy the inner formula $\langle\langle A, B\rangle\rangle \Diamond q$ is allowed to be different from the strategy of A referred to by the outer operator. That is, to satisfy the inner formula, A is allowed to *revoke* the strategy selected by the outer operator.

This aspect of the ATL semantics has been questioned [1], and it has been proposed that the semantics of the formula $\langle\langle G\rangle\rangle\phi$ should be defined so that it fixes the strategies of agents in the group G and does not allow these to be varied in interpreting operators in the formula ϕ. In such a semantics, the strategy choices are *irrevocable*. Using our framework, both revocable and irrevocable interpretations of the formula can be represented. We show this with two formulas that are almost identical, with the point of difference indicated by use of bold type. The interpretation allowing strategy revocation would be captured by translating both operators as described above, yielding the formula

$$\neg D_e \neg D_{\{e,\sigma(A)\}}\,(\Box p \wedge \neg \mathbf{D_e} \neg D_{\{e,\sigma(A),\sigma(B)\}}\,\Diamond q).$$

Note that here the outer operator prefix $\neg D_e \neg D_{\{e,\sigma(A)\}}$ selects a strategy for A and plays it against all strategies of the other agents, and because the operator $\mathbf{D_e}$ allows all agent's strategies to vary, the inner operator prefix $\neg \mathbf{D_e} \neg D_{\{e,\sigma(A),\sigma(B)\}}$ drops the selected strategy of A and selects a fresh strategy for A and B together to play against all strategies of other agents. However, we can force the strategy of agent A to remain fixed in the inner choice of strategies by means of the formula

$$\neg D_e \neg D_{\{e,\sigma(A)\}}\,(\Box p \wedge \neg \mathbf{D}_{\{\mathbf{e},\sigma(\mathbf{A})\}}\,\neg D_{\{e,\sigma(A),\sigma(B)\}}\,\Diamond q).$$

Note that the inner operator $\mathbf{D}_{\{\mathbf{e},\sigma(\mathbf{A})\}}$ varies all agent's strategies, except that of A. Evidently, at any point in a nested formula, our approach gives us the freedom to choose which players' strategies we wish to vary and which to fix.

A logic with revocable strategies is presented in Brihaye et al. [7], which considers the extension of ATL with strategy context, or ATL_{sc}. Formulas are evaluated with respect to a context that is a group strategy γ_G for some group G. The logic has modalities $\cdot\rangle H\langle\cdot\phi$, and $\langle\cdot H\cdot\rangle\phi$. Intuitively, $\cdot\rangle H\langle\cdot\phi$ reduces the context group G to $G \setminus H$ by restricting γ_G to $G \setminus H$. The modality $\langle\cdot H\cdot\rangle\phi$ selects a new group strategy γ_H for group H and constructs the new context $\gamma_H \circ \gamma_G$ for group $G \cup H$ in which agents i in H play $\gamma_H(i)$, and agents i in $G \setminus H$ play $\sigma_G(i)$. The formula ϕ is then evaluated with respect to context $\gamma_H \circ \gamma_G$ in all runs in which $G \cup H$ plays $\gamma_H \circ \gamma_G$ against an arbitrary behaviour of all other agents.

Evaluation of formulas commences with respect to the empty context, so each subformula is evaluated with respect to a context for a group G that can be determined from the operators on the path from the root to that subformula. This means that to represent a formula ϕ of ATL_{sc}, we need to translate it with respect to a group G; we write the translation as ϕ^G. Roughly, with respect to a context for group G, the formula $\langle\cdot H\cdot\rangle\phi$ can then be expressed with our logic as

$$(\langle\cdot H\cdot\rangle\phi)^G = \neg D_{\{e,\sigma(G\setminus H)\}}\,\neg D_{\{e,\sigma(G\cup H)\}}\phi^{G\cup H},$$

and the formula $\cdot\rangle H\langle\cdot\phi$ can be expressed as

$$(\,\cdot\rangle H\langle\cdot\phi\,)^G = \phi^{G\setminus H}.$$

However, we note that the semantics in Reference [7] is based on perfect recall. This explains that the complexity of model checking ATL_{sc} is non-elementary, while the complexity of model checking our logic ESL is EXPSPACE-complete (Theorem 4.1 and Theorem 4.2).

Q5 Another work by van der Hoek, Jamroga, and Wooldridge [28] introduces constants that refer to strategies and adds to ATL a new (counterfactual) modality $C_i(c,\phi)$, with the intended reading "if it were the case that agent i committed to the strategy denoted by c, then ϕ." (The meaning of c is bound in the semantic context, and the logic does not allow quantification over c.) The formula ϕ here is not permitted to contain further references to agent i strategies. To interpret the formula $C_i(c,\phi)$ in an environment E, the environment is first updated to a new environment E' by removing all transitions that are inconsistent with agent i running the strategy referred to by c, and then the formula ϕ is evaluated in E'. In ESL, the assertion that i is running a particular strategy can be made by the formula $\mathsf{e}_{\sigma(i)}(x)$, where x is taken to denote a global state in which the local component $\sigma(i)$ denotes the strategy denoted by c. The formula $C_i(c,\phi)$ can then be expressed in our framework as

$$D_{\{e\}\cup\sigma(Ags\setminus\{i\})}(\mathsf{e}_{\sigma(i)}(x) \Rightarrow \phi^{+\sigma(i)}),$$

where in the translation $\phi^{+\sigma(i)}$ of ϕ we ensure that there is no further deviation from the strategy of agent i by adding $\sigma(i)$ to the group of every knowledge operator occurring later in the translation. We remark that because it deletes information from the transition relation, strategy choices made by the construct $C_i(c,\phi)$ are irrevocable, whereas our logic is richer in that it allows revocation of the corresponding choices.

3.3 Connections to Variants of ATEL

Alternating temporal epistemic logic (ATEL) adds epistemic operators to ATL [29]. As a number of subtleties arise in the formulation of such logics, several variants of ATEL have since been developed. In this section, we consider a number of such variants and argue that our framework is able to express the main strategic concepts from these variants. We begin by recalling ATEL as defined in Reference [29].

The syntax of ATEL is given as follows:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \langle\langle G\rangle\rangle\bigcirc\phi \mid \langle\langle G\rangle\rangle\Box\phi \mid \langle\langle G\rangle\rangle(\phi_1 U \phi_2) \mid K_i\phi \mid D_G\phi \mid C_G\phi,$$

where $p \in Prop$, $i \in Ags$, and $G \subseteq Ags$. This just adds the operators K_i, D_G, and C_G to the syntax for ATL given above. As usual, we may define $E_G\phi$ as $\bigwedge_{i\in G} K_i\phi$. The intuitive meaning of the constructs is as in CTL*K above, with, additionally, $\langle\langle G\rangle\rangle\phi$ having the intuitive reading that group G has a strategy for assuring that ϕ holds.

The relation $E, s \models^\Delta \phi$ is extended from ATL to ATEL by adding the following clauses to the inductive definition:

- $E, s \models^\Delta K_i\phi$ if $E, t \models^\Delta \phi$ for all $t \in S$ with $t \sim_i s$;
- $E, s \models^\Delta D_G\phi$ if $E, t \models^\Delta \phi$, for all $t \in S$ with $(s, t) \in \bigcap_{i\in G} \sim_i$;
- $E, s \models^\Delta C_G\phi$ if $E, t \models^\Delta \phi$ for all $t \in S$ with $(s, t) \in (\cup_{i\in G} \sim_i)^*$;

where we define, for each $i \in Ags$, the equivalence relation $\sim_i$ on states S by $s \sim_i t$ if and only if $O_i(s) = O_i(t)$.

The specific version of ATEL defined in Reference [29] is obtained from the above definitions by taking $\Delta = \{\sigma_G \mid G \subseteq Ags,\ \sigma_G \text{ a deterministic } G\text{-strategy in } E\}$. That is, following the definitions for ATL, this version works with arbitrary deterministic group strategies, in which an agent selects its action as if it had full information of the state. This aspect of the definition has been criticized by Jonker [39] and (in the case of ATL without epistemic operators) by Schobbens [50], who argue that this choice is not in the spirit of the epistemic extension, in which observations are intended precisely to represent that agents do not have full information of the state. They propose that the definition instead be based on the set $\Delta = \{\sigma_G \mid G \subseteq Ags,\ \sigma_G \text{ a locally uniform deterministic } G\text{-strategy in } E\}$. This ensures that in choosing an action, agents are able to use only the information available in their observations.

We concur that the use of locally uniform strategies is the more appropriate choice, but in either event, we now argue that our approach using strategy space is able to express everything that can be expressed in ATEL. We may extend the translation into our logic given above from ATL to ATEL by adding the following rules:

$$(K_i\phi)^* = K_i\phi^* \quad (D_G\phi)^* = D_G\phi^* \quad (C_G\phi)^* = C_G\phi^*.$$

To obtain a correspondence with ATEL, which does not have a notion of initial states, we again work with environments in which all states are initial. The following result shows that Theorem 3.1 extends from ATL to the logic ATEL.

Theorem 3.2. *For every environment E in which all states are initial, for every nonempty set of group strategies Δ that is restrictable, for every state s of E and ATEL formula ϕ, we have $E, s \models^{\Delta} \phi$ iff for all (equivalently, some) points (r, m) of $\mathcal{I}(E, comp(\Delta))$ with $r_e(m) = s$ we have $\mathcal{I}(E, comp(\Delta)), (r, m) \models \phi^*$.*

Proof. The proof extends the proof of Theorem 3.1. The argument for the equivalence of the universal and existential quantifications in the right-hand side of the "iff" continues to apply, even though the translation now contains formulas of the form $K_i\phi$, because, by construction in Section 2.2, $r_e(m) = r'_e(m')$ implies $r_i(m) = r'_i(m')$. The remainder of the proof extends the inductive argument.

Consider $\phi = K_i\psi$. Then $(K_i\psi)^* = K_i\psi^*$. We suppose first that $E, s \models^{\Delta} \phi$ and show that for all points (r, m) of $\mathcal{I}$ with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \phi^*$, i.e., $\mathcal{I}, (r, m) \models K_i\psi^*$. Let (r, m) be a point of $\mathcal{I}$ with $r_e(m) = s$. We need to show that for all points (r', m') of $\mathcal{I}$ with $(r, m) \sim_i (r', m')$ we have $\mathcal{I}, (r', m') \models \psi^*$. But if $(r, m) \sim_i (r', m')$ then $r'_e(m') \sim_i r_e(m) = s$ in E. Thus, from $E, s \models^{\Delta} K_i\psi$ it follows that $E, r'_e(m') \models^{\Delta} \psi$. By the induction hypothesis, we obtain that $\mathcal{I}, (r', m') \models \psi^*$, as required.

Conversely, suppose that for all points (r, m) of $\mathcal{I}$ with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models K_i\psi^*$. We show that $E, s \models^{\Delta} K_i\psi$. Let t be any state of E with $s \sim_i t$. We have to show $E, t \models^{\Delta} \psi$. First, since s is an initial state of E, there exists a run r of $\mathcal{I}$ with $r_e(0) = s$, and joint strategy equal to any strategy in $comp(\Delta)$, so we take $m = 0$, and we have $\mathcal{I}, (r, m) \models K_i\psi^*$. Then for all points (r', m') of $\mathcal{I}$ with $r'_e(m') = t$, we have $(r, 0) \sim_i (r', m')$, from which it follows that $\mathcal{I}, (r', m') \models \psi^*$. By the induction hypothesis, we have $E, t \models \psi$, as required. This completes the proof for the case of $\phi = K_i\psi$. The argument for the distributed and common knowledge operators is similar, and left to the reader. □

We remark that our translation maps ATEL into CTLK($Ags \cup \sigma(Ags), Prop$), the fragment that that we show in Theorem 4.5 below to have PSPACE-complete model-checking complexity. This strongly suggests that this fragment has a strictly stronger expressive power than ATEL, since the complexity of model-checking ATEL, assuming uniform strategies, is known to be P^{NP}-complete. (The class P^{NP} consists of problems solvable by PTIME computations with access to an NP oracle.) For ATEL, model checking can be done with a polynomial time (with respect to the size of formula) computation with access to an oracle that is in NP with respect to both the number of states and the number of joint actions. In particular, Reference [50] proves this upper bound, and Reference [37] proves a matching lower bound.

Similar translation results can be given for other alternating temporal epistemic logics from the literature. We sketch a few of these translations here.

Jamroga and van der Hoek [38] discuss issue of *de dicto* and *de re* interpretations of ATEL formulas. They consider the formula $K_i\langle\langle i\rangle\rangle\phi$. (Note that here ϕ is a path formula). The ATEL semantics states that for an environment E and a state s, we have $E, s \models K_i\langle\langle i\rangle\rangle\phi$ when in every state t consistent with agent i's knowledge, some strategy for agent i, depending on t, is guaranteed to satisfy ϕ. This is consistent with there being no *single* strategy for agent i that agent i knows will work to achieve ϕ in all such states t. To express that a single strategy is known to guarantee ϕ, they formulate a general construct $\langle\langle G\rangle\rangle^{\bullet}_{\mathcal{K}(H)}\phi$ that says, effectively, that there is a strategy for a group G that another group H knows (for notion of group knowledge $\mathcal{K}$) to achieve goal ϕ. (Here again, ϕ is a path formula.) The notion of group knowledge $\mathcal{K}$ could be E for everyone knows, D for distributed knowledge, or C for common knowledge. More precisely[6],

$E, s \models^{\Delta} \langle\langle G \rangle\rangle^{\bullet}_{\mathcal{K}(H)} \phi$ if there exists a locally uniform group strategy $\alpha \in \Delta$ for group G such that for all states t with $s \sim^{\mathcal{K}}_{H} t$, and for all paths ρ from t that are consistent with α, we have that $E, \rho \models^{\Delta} \phi$.

Here, $\sim^{\mathcal{K}}_{H}$ is the appropriate epistemic indistinguishability relation on states of E. The particular case $\langle\langle G \rangle\rangle^{\bullet}_{E(G)} \phi$ is also proposed as the semantics for the ATL construct $\langle\langle G \rangle\rangle \phi$ in References [35, 39, 50].

The construct $\langle\langle G \rangle\rangle^{\bullet}_{D(H)} \phi$ can be represented in the CTLK($Ags \cup \sigma(Ags), Prop$) fragment of ESL as

$$\neg K_e \neg D_{H \cup \sigma(G)} \phi.$$

Intuitively, here the first modal operator $\neg K_e \neg$ switches the strategy of all the agents while maintaining the state s, thereby selecting a strategy α for group G in particular, and the next operator $D_{H \cup \sigma(G)}$ verifies that the group H knows that the strategy being used by group G guarantees ϕ. Similarly, $\langle\langle G \rangle\rangle^{\bullet}_{E(H)} \phi$ can be represented as

$$\neg K_e \neg \bigwedge_{i \in H} D_{\{i\} \cup \sigma(G)} \phi.$$

The precise statement and proof of these correspondences is similar to that in Theorem 3.3.

In the case of the construct $\langle\langle G \rangle\rangle^{\bullet}_{C(H)} \phi$, the definition involves the common knowledge that a group H of agents would have if they knew a particular strategy being used by another group G. By analogy with the above cases, one might expect this to be expressible using the formula $\neg K_e \neg C_{H \cup \sigma(G)} \phi$. However, this does not give the intended meaning. Note that the semantics of the formula $C_{H \cup \sigma(G)} \phi$ quantifies over points (r', m') reachable through chains $(r, m) = (r_0, m_0) \sim_{i_1} (r_1, m_1) \sim_{i_2} \ldots \sim_{i_n} (r_n, m_n) = (r', m')$, where each i_j is in the set $H \cup \sigma(G)$. But this loses the connection to common knowledge of group H and fails to fix the strategy of group G. Instead, what we would need to capture is chains of the form $(r_0, m_0) \sim_{\{i_1\} \cup \sigma(H)} (r_1, m_1) \sim_{\{i_2\} \cup \sigma(H)} \ldots \sim_{\{i_n\} \cup \sigma(H)} (r_n, m_n) = (r', m')$, where each i_j is in the set G. For this, it appears we need to be able to express the greatest fixpoint X of the equation $X \equiv \bigwedge_{i \in G} D_{\{i\} \cup \sigma(H)} (X \wedge \phi)$. The language CTLK($Ags \cup \sigma(Ags), Prop$) does not include fixpoint operators and it does not seem that this fixpoint is expressible. Indeed, the construct $\langle\langle G \rangle\rangle^{\bullet}_{C(H)} \phi$ does not appear to be expressible using the fragment CTLK($Ags \cup \sigma(Ags), Prop$).

However, common knowledge of group H about the effects of a fixed strategy of group G can be expressed with ESL in a natural way by the formula

$$C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi),$$

which says that it is common knowledge to the group H that ϕ holds if the group G is running the strategy profile captured by the variable x. Using this idea, the construct $\langle\langle G \rangle\rangle^{\bullet}_{C(H)} \phi$ can be represented with ESL as

$$\exists x. C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi).$$

The following result states this claim precisely.

Theorem 3.3. *Let E be an environment in which all states are initial, and let Δ be a restrictable and extendable set of group strategies in E. Let $\mathcal{I} = \mathcal{I}(E, comp(\Delta))$. Assume that ϕ is a path formula and*

[6] As above, we have generalized the definition to be relative to a set of group strategies Δ. The strategies used in Reference [38] are imperfect information, perfect recall strategies; we formulate the definition here with imperfect information, imperfect recall strategies.

that ϕ^* is an ESL formula without free variables, such that for every path ρ of E, we have $E, \rho \models^\Delta \phi$ iff for all (equivalently, some) points (r, m) of $\mathcal{I}$ with $r_e[m \ldots] = \rho$ we have $\mathcal{I}, (r, m) \models \phi^*$.

Then for all states s of E, we have $E, s \models^\Delta \langle\langle G \rangle\rangle^\bullet_{C(H)} \phi$ iff for all (equivalently, some) points (r, m) of $\mathcal{I}$ with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \exists x. C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$.

Proof. The argument for the equivalence between the universal and existential versions of the right hand side of the iff is similar to that in Theorem 3.1.

Suppose, first, that $E, s \models^\Delta \langle\langle G \rangle\rangle^\bullet_{C(H)} \phi$. Let (r, m) be a point of $\mathcal{I}$ with $r_e(m) = s$. We need to prove that $\mathcal{I}, (r, m) \models \exists x. C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$. From $E, s \models^\Delta \langle\langle G \rangle\rangle^\bullet_{C(H)} \phi$ it follows that there exists a strategy $\alpha \in \Delta$ for group G, such that for all states t with $s \sim^C_H t$ and paths ρ from t consistent with α, we have $E, \rho \models^\Delta \phi$. Let r' be any run with $r'_{\sigma(G)}(0) = \alpha$, and define Γ to be a context with $\Gamma(x) = r(0)$. To prove $\mathcal{I}, (r, m) \models \exists x. C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$, we show that $\Gamma, \mathcal{I}, (r, m) \models C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$. For this, suppose that $(r, m) = (r^0, m^0) \sim_{i_1} (r^1, m^1) \sim_{i_2} \ldots \sim_{i_k} (r^k, m^k)$, where $i_j \in H$ for $j = 1 \ldots k$, and assume that $\Gamma, \mathcal{I}, (r^k, m^k) \models \mathsf{e}_{\sigma(G)}(x)$. We need to show that $\Gamma, \mathcal{I}, (r^k, m^k) \models \phi^*$.

Note that we have $s = r(m) = r^0_e(m^0) \sim_{i_1} \ldots \sim_{i_k} r^k_e(m^k)$. Since $\Gamma, \mathcal{I}, (r^k, m^k) \models \mathsf{e}_{\sigma(G)}(x)$, we have that $r^k_{\sigma(G)}(m^k) = \Gamma(x)_{\sigma(G)} = \alpha$. Thus, the sequence $\rho = r^k_e[m^k \ldots]$ is a path of E consistent with the group strategy α. It follows that $E, \rho \models^\Delta \phi$. By assumption, this means that $\Gamma, \mathcal{I}, (r^k, m^k) \models \phi^*$.

Conversely, let (r, m) be a point of $\mathcal{I}$ with $r_e(m) = s$ and $\mathcal{I}, (r, m) \models \exists x. C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$, witnessed by $\Gamma, \mathcal{I}, (r, m) \models C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$. Note that $\Gamma(x)_{\sigma(Ags)} = comp(\beta)$, where $\beta \in \Delta$ is a group strategy for some group G'. For agents $i \in G \setminus G'$, we have that $\Gamma(x)_{\sigma(i)}$ is the random strategy $rand_i$. It follows that any path consistent with $\Gamma(x)_{\sigma(G \cap G')}$ is also consistent with $\Gamma(x)_{\sigma(G)}$. Let $\alpha \in \Delta$ be any group strategy for group G with $\alpha \restriction (G \cap G') = \Gamma(x)_{\sigma(G \cap G')}$. Such a strategy exists by the fact that Δ is restrictable and extendable: We may take α to be an extension of $\beta \restriction (G \cap G')$. Then we have that any path consistent with α is consistent with $\Gamma(x)_{\sigma(G)}$.

We show $E, s \models^\Delta \langle\langle G \rangle\rangle^\bullet_{C(H)} \phi$, with α as the witnessing strategy for group G. For this, let $s = s_0 \sim_{i_1} s_1 \sim_{i_2} \ldots \sim_{i_k} s_k$, where $i_j \in H$ for $j = 1 \ldots k$, and let ρ be a path from s_k consistent with α. We show $E, \rho \models^\Delta \phi$. By the observation above, ρ is also consistent with $\Gamma(x)_{\sigma(G)}$. Let r^k be a run with $r^k_e[0 \ldots] = \rho$, and $r^k_{\sigma(G)}(0) = \Gamma(x)_{\sigma(G)}$. (We can take $r^k_{\sigma(Ags)} = comp(\beta \restriction (G \cap G'))$, which is in $comp(\Delta)$.) Then $\Gamma, \mathcal{I}, (r, 0) \models \mathsf{e}_{\sigma(G)}(x)$. Moreover, for each $j = 1 \ldots k-1$, let r^j be any run with $r^j_e(0) = s_j$. Then $(r, m) = (r^0, m^0) \sim_{i_1} (r^1, 0) \sim_{i_2} \ldots \sim_{i_k} (r^k, 0)$. It follows from $\Gamma, \mathcal{I}, (r, m) \models C_H(\mathsf{e}_{\sigma(G)}(x) \Rightarrow \phi^*)$ that $\Gamma, \mathcal{I}, (r^k, 0) \models \phi^*$, and in fact $\mathcal{I}, (r^k, 0) \models \phi^*$, since ϕ^* has no free variables. Since $r^k[0 \ldots] = \rho$, by assumption, we have $E, \rho \models \phi$. This proves $E, s \models^\Delta \langle\langle G \rangle\rangle^\bullet_{C(H)} \phi$. □

The above equivalences give a reduction of the complex operators of Reference [38] that makes their epistemic content more explicit by expressing this using standard epistemic operators.

An alternate approach to decomposing the operators $\langle\langle G \rangle\rangle^\bullet_{\mathcal{K}(H)}$ is proposed in Reference [35]. By comparison with our standard approach to the semantics of the epistemic operators, this proposal uses "constructive knowledge" operators that require a nonstandard semantics in which formulas are evaluated at sets of states rather than at individual states. Evaluation at single world q is treated as equivalent to evaluation at the set $\{q\}$. For each standard (group) epistemic operator $\mathcal{K} = E, D, C$, there is a constructive version $\hat{\mathcal{K}} = \mathbb{E}, \mathbb{D}, \mathbb{C}$. Atomic propositions p are evaluated at sets of states Q by

$$E, Q \models p \quad \text{if for all states } q \in Q \text{ we have } E, q \models p.$$

(As above, we define the semantics on environments E rather than ATEL models.) For the constructive epistemic operators,

$$E, Q \models \hat{\mathcal{K}}_G \phi \quad \text{if } E, \left\{q' \in Q \mid \exists q \in Q \left(q \sim_G^{\mathcal{K}} q'\right)\right\} \models \phi$$

and for the ATL operator $\langle\!\langle G \rangle\!\rangle \phi$ we have

> $E, Q \models \langle\!\langle G \rangle\!\rangle \phi$ if there exists a strategy α for group G such that ϕ holds in all runs starting in a state in Q in which group G plays the strategy α.

Note that the ATEL formula $\mathcal{K}_G \langle\!\langle G \rangle\!\rangle \phi$ says that at each world considered possible (in the appropriate sense for $\mathcal{K}$) by group G, there exists a (possibly different) strategy for G that achieves ϕ. By contrast, $\mathcal{K}_G \langle\!\langle G \rangle\!\rangle \phi$ says that there exists a *single* strategy for G that achieves ϕ from each world considered possible (in the appropriate sense for $\mathcal{K}$) by G.

This logic is shown in Reference [36] to have a normal form, in which every subformula starting with a constructive knowledge operator $\hat{\mathcal{K}}_G^1$ is of the form $\hat{\mathcal{K}}_{G_1}^1 \ldots \hat{\mathcal{K}}_{G_n}^n \phi$, where ϕ starts with a strategy modality and each $\mathcal{K}^i \in \{E, D, C\}$. Such a normal form subformula, evaluated at a single state, can be represented in ESL as

$$\exists x. \mathcal{K}_{G_1}^1 \ldots \mathcal{K}_{G_n}^n (\mathsf{e}_{\sigma(H)}(x) \Rightarrow \phi).$$

Precise formulation and proof of the claim are similar to the proofs above and left to the reader.

3.4 Strategy Logic

Chatterjee et al's strategy logic [13], which we call CHP-SL, following the convention in Reference [42], is an extension of ATL* for two-player games. Let x, y be two variables ranging over Player 1 and Player 2's strategies. The logic allows these variables to be quantified: If ϕ is a formula, then $\exists x.\phi$ and $\forall x.\phi$ are formulas. Additionally the effects of a particular combination of player strategies can be expressed using the formula $\phi(x, y)$, which says that ϕ holds if player 1 plays strategy x and player 2 plays strategy y. Thus, the ATL* formula $\langle\!\langle 1 \rangle\!\rangle \phi$ can be expressed in CHP-SL with $(\exists x)(\forall y)\phi(x, y)$.

Strategy logic (SL) [42] generalises CHP-SL, with the syntax as follows:

$$\phi \equiv p \mid \neg\phi \mid \phi \wedge \phi \mid \bigcirc\phi \mid \phi U \phi \mid \langle\!\langle v \rangle\!\rangle \phi \mid [[v]]\phi \mid (i, v)\phi,$$

where $v \in \mathtt{Var}_{\mathrm{SL}}$ such that $\mathtt{Var}_{\mathrm{SL}}$ is a set of strategy variables, and $i \in Ags$ is an agent. Intuitively, $\langle\!\langle v \rangle\!\rangle \phi$ says that there exists a strategy v such that ϕ, formula $[[v]]\phi$ says that ϕ holds for all strategies v, and $(i, v)\phi$ says that ϕ holds if agent i plays strategy v. A formula is a *sentence* if every occurrence of (a, x) is within the scope of an occurrence of $\langle\!\langle x \rangle\!\rangle$ or $[[x]]$, and every temporal subformula $\bigcirc\phi$ or $\phi U \phi$ occurs within the context of some binding (i, x), for every agent i. The ATL* formula $\langle\!\langle 1 \rangle\!\rangle \phi$ can be expressed in SL as $\langle\!\langle x \rangle\!\rangle [[y]](1, x)(2, y)\phi$.

Let $\mathtt{Str}$ be a set of agent strategies, and $\chi : Ags \cup \mathtt{Var}_{\mathrm{SL}} \to \mathtt{Str}$ be a partial mapping from agents and variables to the set of strategies. Then, the semantics can be formulated with respect to our environments E as follows[7]:

- $E, \chi, (r, m) \models \langle\!\langle v \rangle\!\rangle \phi$ iff there exists a strategy $\sigma \in \mathtt{Str}$ such that $E, \chi[v \mapsto \sigma], s \models \phi$;
- $E, \chi, (r, m) \models [[v]]\phi$ iff for all strategies $\sigma \in \mathtt{Str}$ it holds that $E, \chi[v \mapsto \sigma], s \models \phi$;
- $E, \chi, (r, m) \models (i, v)\phi$ iff $E, \chi[i \mapsto \chi(v)], (r', m) \models \phi$ for all runs r' where $r(m) = r'(m)$ and r' is a run consistent with $\chi[i \mapsto \chi(v)]$ from time m.

[7] We make some simplifications; the authors of Reference [42] distinguish between path and state formulas.

Atomic, Boolean, and temporal formulas are handled as usual. We remark that because (1) the transition relation is assumed in SL to be deterministic, i.e., $\rightarrow$ can be written as a function of type $S \times Acts \rightarrow S$, and (2) temporal operators in a sentence appear only in contexts where every agent is bound to a strategy, the final binding (i, v) before temporal operators are evaluated in fact quantifies over just a single run.

Given an SL formula ϕ, we let $V(\phi)$ be the set of variables in the operators $\langle\!\langle\rangle\!\rangle$ or $[[]]$. SL allows the assignment of a strategy to multiple agents, e.g., in formula $\langle\!\langle v\rangle\!\rangle((i, v)\phi_1 \wedge (j, v)\phi_2)$ the agents i and j have the same strategy represented in the variable v. For this to make sense in an imperfect information system, without allowing implausible bindings or artificially complex interpretations of quantification, all agents need to have the same actions and the same observations. This does not match the setting of our framework particularly well. We remark that CHP-SL does not allow this expressivity, as players 1 and 2 are associated with their dedicated strategy variables x and y, respectively.

In the following, we consider the fragment of SL in which every variable v_i is uniquely associated with an agent $i \in Ags$, so that v_i occurs only in bindings $(j, v_i)\psi$ with $j = i$. Then we can translate a SL formula ϕ into an ESL formula ϕ^* as follows:

$$\begin{aligned}
p^* &= p \\
(\neg\phi)^* &= \neg\phi^* \\
(\phi_1 \wedge \phi_2)^* &= \phi_1^* \wedge \phi_2^* \\
(\bigcirc\phi)^* &= \bigcirc\phi^* \\
(\phi_1 U \phi_2)^* &= \phi_1^* U \phi_2^* \\
(\langle\!\langle v_i\rangle\!\rangle\phi)^* &= \exists v_i \phi^* \\
([[v_i]]\phi)^* &= \forall v_i \phi^* \\
((i, v_i)\phi)^* &= D_{e \cup \sigma(Ags \setminus \{i\})}(\mathsf{e}_{\sigma(i)}(v_i) \Rightarrow \phi^*).
\end{aligned}$$

Intuitively, to decide if $\langle\!\langle v_i\rangle\!\rangle\phi$, we need to determine the existence of a strategy v_i with respect to which the formula ϕ is satisfied. In the ESL translation, v_i refers to a global state rather than a strategy, but the only component of this global state that is used in the remainder of the evaluation is the component $\sigma(i)$, which picks out a strategy for agent i and similarly for $[[v_i]]\phi$. To decide if $(i, v_i)\phi$, we need to satisfy ϕ on (all) runs where agent i's strategy is switched to that represented in v_i. The translation handles this using the operator $D_{e \cup \sigma(Ags \setminus \{i\})}$, which refers to points in which the state of the environment and the strategies of all agents are fixed, while the strategy of agent i is allowed to vary. The assertion $\mathsf{e}_{\sigma(i)}(v_i)$ checks that the strategy of agent i is in fact switched to that represented in the global state v_i.

Similarly, CHP-SL formulas can be translated into ESL formulas as follows:

$$\begin{aligned}
(\exists x.\phi)^* &= \exists x \phi^* \\
(\forall x.\phi)^* &= \forall x \phi^* \\
(\phi(x, y))^* &= D_e(\mathsf{e}_{\sigma(1)}(x) \wedge \mathsf{e}_{\sigma(2)}(y) \Rightarrow \phi^*).
\end{aligned}$$

Finally, we remark that both the CHP-SL semantics in Reference [13] and the SL semantics in Reference [42] are for perfect recall. Since we have formulated ESL for imperfect recall, we leave the above translations as indicative rather than attempting a formal proof.

3.5 Game-theoretic Solution Concepts

It has been shown for a number of logics for strategic reasoning that they are expressive enough to state a variety of game-theoretic solution concepts, e.g., References [13, 28] show that Nash Equilibrium is expressible. We now sketch the main ideas required to show that the fragment CTLK($Ags \cup \sigma(Ags) \cup \{e\}, Prop$) of our framework also has this expressive power. We assume two players $Ags = \{0, 1\}$ in a normal form perfect information game and assume that these agents play a deterministic strategy. The results in this section can be easily generalized to multiple players and extensive form games.

Given a game $\mathcal{G}$ we construct an environment $E_{\mathcal{G}}$ that represents the game. Each player has a set of actions that correspond to the moves that the player can make. We assume that $E_{\mathcal{G}}$ is constructed to model the game so that play happens in the first step from a unique initial state and that subsequent transitions do not change the state. We let agents have perfect information in $E_{\mathcal{G}}$, i.e., we define the observation of agent i in state s by $O_i(s) = s$. (Consequently, although we use uniform strategies $\Sigma^{unif,\det}$ below, the uniformity constraint is vacuous in these environments.)

We write $-i$ to denote the adversary of player i. Let u_i for $i \in \{0, 1\}$ be a variable denoting the utility gained by player i when play is finished. Let V_i be the set of possible values for u_i, and let $V = V_0 \cup V_1$. We work with the following atomic propositions. Atomic proposition $u \leq v$, where $u, v \in V$, expresses the ordering on utilities. Atomic proposition $u_i = v$, where $i \in \{0, 1\}$ and $v \in V_i$, expresses that player i's utility has value v. We use formula

$$U_i(v) = \bigcirc(u_i = v)$$

to express that value v is player i's utility once play finishes.

Nash equilibrium (NE) is a solution concept that states that no player can gain by unilaterally changing their strategy. We may write

$$BR_i(v) = U_i(v) \land K_{\sigma(-i)} \bigwedge_{v' \in V_i} (U_i(v') \Rightarrow v' \leq v)$$

to express that, given the current strategy $\sigma(-i)$ of the adversary of i, the value v attained by player i's current strategy is the best possible utility attainable by player i, i.e., the present strategy of player i is a best response to the adversary. Thus,

$$BR_i = \bigvee_{v \in V_i} BR_i(v)$$

says that player i is playing a best-response to the adversary's strategy. The following statement then expresses the existence of a (pure) Nash equilibrium for the game $\mathcal{G}$:

$$E_{\mathcal{G}}, \Sigma^{unif,det}(E_{\mathcal{G}}) \models \neg D_{\emptyset} \neg (BR_0 \land BR_1).$$

That is, in a Nash equilibrium, each player is playing a best response to the other's strategy.

Perfect cooperative equilibrium (PCE) is a solution concept intended to overcome deficiencies of Nash equilibrium for explaining cooperative behaviour [26]. It says that each player does at least as well as she would if the other player were best-responding. The following formula:

$$BU_i(v) = D_{\emptyset}\left(\bigwedge_{v' \in V_i} ((BR_{-i} \land U_i(v')) \Rightarrow v' \leq v)\right)$$

states that v is as good as any utility that i can obtain if the adversary always best-responds to whatever i plays. Thus,

$$BU_i = \bigvee_{v \in V_i} (U_i(v) \land BU_i(v))$$

says that i is currently getting a utility as good as the best utility that i can obtain if the adversary is a best-responder. Now, the following formula expresses the existence of perfect cooperative equilibrium for the game $\mathcal{G}$:

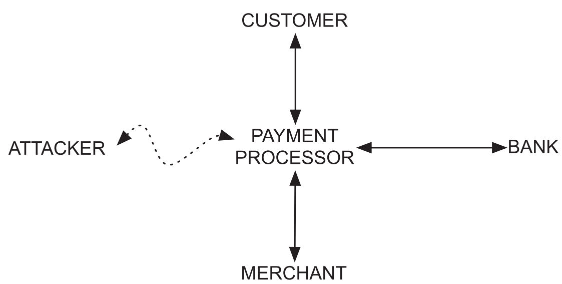
$$E_{\mathcal{G}}, \Sigma^{unif,det}(E_{\mathcal{G}}) \models \neg D_{\emptyset} \neg (BU_0 \wedge BU_1).$$

That is, in a PCE, no player has an incentive to change their strategy, on the assumption that the adversary will best-respond to any change.

3.6 Computer Security Example: Erasure Policies

Formal definitions of computer security frequently involve reference to the strategies available to the players, and to agents' reasoning based on these strategies. In this section, we sketch an example that illustrates how our framework might be applied in this context.

Consider the scenario depicted in the following diagram:



A customer C can purchase items at a web merchant M. Payment is handled by a trusted payment processor P (this could be a service or device), which interacts with the customer, merchant, and a bank B to securely process the payment. (To keep the example simple, we suppose that the customer and merchant use the same bank). One of the guarantees provided by the payment processor is to protect the customer from attacks on the customer's credit card by the merchant: The specification for the protocol that runs the transaction requires that the merchant should not obtain the customer's credit card number. In fact, the specification for the payment processor is that after the transaction has been successfully completed, the payment processor should *erase* the credit card data to ensure that even the payment processor's state does not contain information about the customer's credit card number. The purpose of this constraint is to protect the customer against subsequent attacks by an attacker A, who may be able to use vulnerabilities in the payment processor's software to obtain access to the payment processor's state.

We sketch how one might use our framework to express the specification. To capture reasoning about all possible behaviours of the agents, and what they can deduce from knowledge of those behaviours, we work in $\mathcal{I}^{unif}(E)$ for a suitably defined environment E. To simplify matters, we take $Ags = \{C, M, P, A\}$. We exclude the strategy of the bank from consideration: This amounts to assuming that the bank has no actions and is trusted to run a fixed protocol. We similarly assume that the payment processor P has no actions, but to talk about what information is encoded in the payment processor's local state, we do allow that this agent has observations. The customer C may have actions such as entering the credit card number in a web form, pressing a button to submit the form to the payment processor, and pressing a button to approve or cancel the transaction. The customer observes variable `cc`, which records the credit card number drawn from a set `CCN`, and Boolean variable `done`, which records whether the transaction is complete (which could mean either committed or aborted).

We assume that the attacker A has some set of exploit actions, as well as some innocuous actions (e.g., setting a local variable or performing *skip*). The effect of the exploit actions is to exploit a vulnerability in the payment processor's software and copy parts of the local state of the payment processor to variables that are observable by the attacker. We include in the environment state a Boolean variable `exploited`, which records whether the attacker has executed an exploit action at some time in the past. The merchant M may have actions such as sending cost information to the payment processor and acknowledging a receipt certifying that payment has been approved by the bank (we suppose this receipt is transmitted from the bank to the merchant via the payment processor).

We may then capture the statement that the system is *potentially* vulnerable to an attack that exploits an erasure flaw in the implementation of the payment processor, by the following formula:

$$\neg D_\emptyset \neg \left(\mathsf{done} \wedge \bigvee_{x \in \mathsf{CCN}} K_P(\mathsf{cc} \neq x) \right),$$

where $\mathsf{cc} \neq x$ is an atomic proposition for each $x \in \mathsf{CCN}$, with the obvious meaning that the customer's credit card number is not x. This says that there exist behaviours of the agents, which can (at least on some points in some runs) leave the payment processor in a state where the customer has received confirmation that the transaction is done, but in which the payment processor's local state somehow still encodes *some* information about the customer's credit card number. This encoding could be direct (e.g., by having a variable *customer_cc* that still stores the credit card number) or indirect (e.g., by the local state including both a symmetric encryption key K and an encrypted version of the credit card number, *enc_customer_cc*, with value $\mathtt{Encrypt}_K(\mathtt{cc})$ that was used for secure transmission to the bank). Note that for a breach of security, it is only required that the information suffices to *rule out* some credit card number (so that, e.g., knowing the first digit of the number would constitute a vulnerability)

The vulnerability captured by this formula is only potential, because it does not necessarily follow that the attacker is able to obtain the credit card information. Whether this is possible can be checked using the formula

$$\neg D_\emptyset \neg \left(\mathsf{done} \wedge \neg \mathtt{exploited} \wedge E\Diamond \bigvee_{x \in \mathsf{CCN}} D_{\{A, \sigma(A)\}}(\mathsf{cc} \neq x) \right),$$

which says that it is possible for the attacker to obtain information about the credit card number even after the transaction is done. (To focus on erasure flaws, we deliberately wish to exclude here the possibility that the attack occurs during the processing of the transaction.) Note that here we assume that the attacker knows his own strategy when making deductions from the information obtained in the attack. This is necessary, because the attacker can typically write his own local variables, so it needs to be able to distinguish between a value it wrote itself and a value it copied from the payment processor.

However, even this formula may not be sufficiently strong. Suppose that the payment processor implements erasure by writing, to its variable *customer_cc*, a random value. Then, even if the attacker obtains a copy of this value, and it happens to be equal to the customer's actual credit card number, the attacker would not have any knowledge about the credit card number, since, as far as the attacker knows, it could be looking at a randomly assigned number. However, there may still be vulnerabilities in the system. Suppose that the implementation of the payment processor operates so that the customer's credit card data is not erased by randomization until the merchant has acknowledged the receipt of payment from the bank, but to avoid annoying the customer with a hanging transaction, the customer is advised that the transaction is approved (setting done true) if the merchant does not respond within a certain time limit. It is still the case that on observing the copied value of *customer_cc*, the attacker would not be able to deduce that this is the customer's credit card number, since it might be the result of erasure in the case that the merchant responded promptly. However, if the attacker knows that the merchant has not acknowledged the receipt, then the attacker can deduce that the value is not due to erasure. One way in which the attacker might know that the merchant has not acknowledged receipt is that the attacker is in collusion with the merchant, who has agreed to omit sending the required acknowledgement messages.

This type of attack can be captured by replacing the term $D_{\{A,\sigma(A)\}}(\mathsf{cc} \neq x)$ by $D_{\{A,\sigma(A),\sigma(M)\}}(\mathsf{cc} \neq x)$, capturing that the attacker reasons using knowledge of both its own strategy as well as the strategy of the merchant or even $D_{\{A,\sigma(A),\sigma(M),M\}}(\mathsf{cc} \neq x)$ for a collusion in which the merchant shares information observed. Similarly, to focus on erasure flaws in the implementation of the payment gateway, independently of the attackers capability, we would replace the term $K_P(\mathsf{cc} \neq x)$ above by $D_{\{P,\sigma(M)\}}(\mathsf{cc} \neq x)$.

We remark that in the case of the attacker's knowledge, it would be appropriate to work with a perfect recall semantics of knowledge, but when using knowledge operators to express information in the payment gateway's state for purposes of reasoning about erasure policy, the more appropriate semantics of knowledge is imperfect recall.

This example illustrates some of the subtleties that arise in the setting of reasoning about security and the way that our framework helps to represent them. Erasure policies have previously been studied in the computer security literature, beginning with Reference [14], though generally without consideration of strategic behaviour by the adversary.

3.7 Reasoning about Knowledge-based Programs

Knowledge-based programs [20] are a form of specification of a multi-agent system in the form of a program structure that describes how an agent's actions are related to its knowledge. They have been shown to be a useful abstraction for several areas of application, including the development of optimal protocols for distributed systems [20], robot motion planning [6], and game-theoretic reasoning [24].

Knowledge-based programs cannot be directly executed, since there is a circularity in their semantics: Which actions are performed depends on what the agents know, which in turn depends on which actions the agents perform. The circularity is not vicious and can be resolved by means of a fixed point semantics, but it means that a knowledge-based program may have multiple distinct implementations (or none), and the problem of reasoning about these implementations is quite subtle. In this section, we show that our framework can capture reasoning about the set of possible implementations of a knowledge-based program.

We consider joint knowledge-based programs P (as defined by Reference [20]) where for each agent i we have a knowledge-based program

$$P_i = \textbf{do}\, \phi^i_1 \rightarrow a^i_1 \, [] \, \ldots [] \, \phi^i_{n_i} \rightarrow a^i_{n_i} \, \textbf{od},$$

where each ϕ^i_j is a formula of CTL*K($Ags, Prop$) of the form $K_i\psi$, and each a_i appears just once. The formulas ϕ^i_j are called the *guards* of the knowledge-based program.[8] Intuitively, this program says to repeat forever the following operation: Nondeterministically execute one of the actions a^i_j such that the corresponding guard ϕ^i_j is true. To ensure that it is always the case that at least one action

[8] The guards in Reference [20] are allowed to be Boolean combinations of formulas $K_i\psi$ and propositions p local to the agent: Since for such propositions $p \Leftrightarrow K_i p$, and the operator K_i satisfies positive and negative introspection, our form for the guards is equally general. They do not require that a_i appears just once, but the program can always be put into this form by aggregating clauses for a_i into one and taking the disjunction of the guards.

enabled, we assume that $\phi^i_1 \lor \ldots \lor \phi^i_{n_i}$ is a valid formula; this can always be ensured by taking the last condition $\phi^i_{n_i}$ to be the "otherwise" condition $K_i\neg(\phi^i_1 \lor \ldots \lor \phi^i_{n_i-1})$, which is equivalent to $\neg(\phi^i_1 \lor \ldots \lor \phi^i_{n_i-1})$ by introspection. In general, the guards in a knowledge-based program may contain common knowledge operators C_G, but we assume for technical reasons (explained below) that no ϕ^i_j contains such an operator.

We present a formulation of semantics for knowledge-based programs that refactors the definitions of Reference [20], following the approach of Reference [41], which uses the notion of environment defined above rather than the original notion of *context*. A *potential implementation* of a knowledge-based program P in an environment E is a joint strategy α in E. (Recall that we use "joint strategy" to refer to a group strategy for the group of all agents.) Given a potential implementation α in E, we can construct the interpreted system $\mathcal{I}_\alpha = \mathcal{I}(E, \{\alpha\})$, which captures the possible runs of E when the agents choose their actions according to the single possible joint strategy α. Given this interpreted system, we can now interpret the epistemic guards in P. Say that a state s of E is α-reachable if there is a point (r, m) of $\mathcal{I}_\alpha$ with $r_e(m) = s$. We note that for a formula $K_i\phi$, and a point (r, m) of $\mathcal{I}_\alpha$, the statement $\mathcal{I}_\alpha, (r, m) \models K_i\phi$ depends only on the state $r_e(m)$ of the environment at (r, m). Recall that $r_e(m)$ determines $r_i(m)$ for $i \in Ags$. For an α-reachable state s of E, it therefore makes sense to define satisfaction of $K_i\phi$ at s rather than at a point by $\mathcal{I}_\alpha, s \models K_i\phi$ if $\mathcal{I}_\alpha, (r, m) \models K_i\phi$ for all (r, m) with $r_e(m) = s$. We define a joint strategy α to be an *implementation* of P in E if for all α-reachable states s of E and agents i, we have

$$\alpha_i(s) = \left\{a^i_j \mid 1 \leq j \leq n_i,\ \mathcal{I}_\alpha, s \models \phi^i_j\right\}.$$

Intuitively, the right-hand side of this equation is the set of actions that are enabled at s by P_i when the tests for knowledge are interpreted using the system obtained by running the strategy α itself. The condition states that the strategy is an implementation if it enables precisely this set of actions at every reachable state. It is easily checked that a strategy α_i satisfying the above equation is uniform.

We now show that our framework for strategic reasoning can express the same content as a knowledge-based program by means of a formula and that this enables the framework to be used for reasoning about knowledge-based program implementations. In general, implementations α of a knowledge-based program P can be hard to find, and there may be one, many, or no implementations of a given knowledge-based program. We therefore work in strategy space $\mathcal{I}(E, \Sigma^{unif})$, which contains all candidate implementations, and develop a formula $\mathbf{imp}(P)$ such that for a given run r, the formula $\mathbf{imp}(P)$ holds at a point of r iff the joint strategy encoded in r is an implementation of P in E.

We need one constraint on the environment. Say that an environment E is *action recording* if for all agents i, for each $a \in Acts_i$ there exists an atomic proposition $did_i(a)$ such that for $s \in I$ we have $did_i(a) \notin \pi(s)$ and for all states s, t and joint actions a such that $(s, a, t) \in \rightarrow$, we have $did_i(b) \in \pi(t)$ iff $b = a_i$. Intuitively, this means that we can determine from a non-initial state the joint action that was executed in reaching that state. It is easily seen that any environment can be made action recording just by adding a component to the states that records the latest joint action.

We can now express knowledge-based program implementations as follows. The main issue that we need to deal with is that the semantics of knowledge formulas in knowledge-based programs is given with respect to a system $\mathcal{I}_\alpha$, in which it is *common knowledge* that the joint strategy in use is α. In general, strategies are not common knowledge in the strategy space $\mathcal{I}(E, \Sigma^{unif})$ within which we wish to reason about knowledge-based program implementations. We handle this by means of a transformation of formulas.

For a formula ϕ of CTL*K$(Ags, Prop)$, not containing common knowledge operators, write $\phi^\$$ for the formula of ESL (in fact, of CTL*K$(Ags \cup \sigma(Ags), Prop)$) obtained from the following recursively defined transformation:

$$\begin{aligned}
p^\$ &= p\\
(\neg\phi)^\$ &= \neg\phi^\$\\
(\phi_1 \wedge \phi_2)^\$ &= \phi_1^\$ \wedge \phi_2^\$\\
(D_G\phi)^\$ &= D_{G\cup\sigma(Ags)}\,\phi^\$\\
(A\phi)^\$ &= A\phi^\$\\
(\bigcirc\,\phi)^\$ &= \bigcirc\,\phi^\$\\
(\phi_1 U \phi_2)^\$ &= (\phi_1^\$\, U\, \phi_2^\$).
\end{aligned}$$

Intuitively, this substitution says that knowledge operators in ϕ are to be interpreted as if it is known that the current joint strategy is being played. In the case of an operator D_G, which includes the special case $K_i = D_{\{i\}}$, the translation handles this by adding $\sigma(Ags)$ to the set of agents that are kept fixed when moving through the indistinguishability relation.

Let

$$\mathbf{imp}(P) = D_{\sigma(Ags)}\left(\bigwedge_{i\in Ags, j=1\ldots n_i}((\phi_j^i)^\$ \Leftrightarrow E\bigcirc did_i(a_j^i))\right).$$

Intuitively, this formula says that the current joint strategy gives an implementation of the knowledge-based program P. More precisely, we have the following:

Proposition 4. *Suppose that P is a knowledge-based program in which the guards do not contain common knowledge operators. Let α be a locally uniform joint strategy in E and let r be a run of $\mathcal{I}(E, \Sigma^{unif}(E))$, in which the agents are running joint strategy α, i.e., $r(0) = (s, \alpha)$ for some state s. Let $m \in \mathbb{N}$. Then*

$$\mathcal{I}(E, \Sigma^{unif}(E)), (r, m) \models \mathbf{imp}(P)$$

iff the strategy α is an implementation of knowledge-based program P in E.

Proof. For brevity, we write just $\mathcal{I}$ for $\mathcal{I}(E, \Sigma^{unif}(E))$. First, we claim that for a formula ϕ not containing common knowledge operators, we have $\mathcal{I}, (r, m) \models \phi^\$$ iff $\mathcal{I}_\alpha, (r, m) \models \phi$, where $r(m) = (s, \alpha)$. The proof is by induction on the construction of ϕ. The base case of atomic propositions, and the cases for Boolean and linear temporal operators are straightforward.

Consider the case $\phi = A\psi$, where we have $(A\psi)^\$ = A(\psi^\$)$. Observe that if r and r' are runs of $\mathcal{I}$, with $r[0 \ldots m] = r'[0 \ldots m]$, then r and r' encode the same strategy $\alpha = r_{\sigma(Ags)}(0)$. Now $\mathcal{I}, (r, m) \models A(\psi^\$)$ iff $\mathcal{I}, (r', m) \models \psi^\$$ for all runs r' of $\mathcal{I}$ with $r[0 \ldots m] = r'[0 \ldots m]$. By the observation, this is equivalent to $\mathcal{I}, (r', m) \models \psi^\$$ for all runs r' of $\mathcal{I}_\alpha$ with $r[0 \ldots m] = r'[0 \ldots m]$. By induction, the latter is equivalent to $\mathcal{I}_\alpha, (r', m) \models \psi$ for all runs r' of $\mathcal{I}_\alpha$ with $r[0 \ldots m] = r'[0 \ldots m]$, i.e., to $\mathcal{I}_\alpha, (r, m) \models A\psi$. Hence $\mathcal{I}, (r, m) \models (A\psi)^\$$ iff $\mathcal{I}_\alpha, (r, m) \models A\psi$.

Finally, consider the case $\phi = D_G\psi$, where we have $(D_G\psi)^\$ = D_{G\cup\sigma(Ags)}(\psi^\$)$. Observe that if (r, m) and (r', m') are points of $\mathcal{I}$ with $(r, m) \sim_{G\cup\sigma(Ags)} (r', m')$, then r and r' encode the same strategy $\alpha = r_{\sigma(Ags)}(0)$ and $(r, m) \sim_G (r', m')$. Conversely, if (r, m) and (r', m') are points of $\mathcal{I}_\alpha$, i.e., both encode joint strategy α, then $(r, m) \sim_G (r', m')$ implies $(r, m) \sim_{G\cup\sigma(Ags)} (r', m')$. Now $\mathcal{I}, (r, m) \models D_{G\cup\sigma(Ags)}(\psi^\$)$ iff $\mathcal{I}, (r', m') \models \psi^\$$ for all points (r', m') of $\mathcal{I}$ with $(r, m) \sim_{G\cup\sigma(Ags)} (r', m')$. By the observation, this is equivalent to $\mathcal{I}, (r', m') \models \psi^\$$ for all points (r', m') of $\mathcal{I}_\alpha$ with $(r, m) \sim_G (r', m')$. By induction, this is equivalent to $\mathcal{I}_\alpha, (r', m') \models \psi$ for all points (r', m') of $\mathcal{I}_\alpha$ with $(r, m) \sim_G (r', m')$, i.e., to $\mathcal{I}_\alpha, (r, m) \models D_G\psi$.

This completes the proof of the claim. Next, note that, for a point (r, m) with $r(m) = (s, \alpha)$, for action α_j^i of agent i, we have $\mathcal{I}, (r, m) \models E\bigcirc did_i(a_j^i)$ iff $a_j^i \in \alpha_i(s)$.

Suppose that α is an implementation of P in E, and let (r,m) be a point of $\mathcal{I}$ with $r(m) = (s,\alpha)$, We show that $\mathcal{I},(r,m) \models \mathbf{imp}(P)$. For this, we let (r',m') be a point with $(r',m') \sim_{\sigma(Ags)} (r,m)$ and show that $\mathcal{I},(r',m') \models \bigwedge_{i\in Ags, j=1\ldots n_i}((\phi^i_j)^\$ \Leftrightarrow E\bigcirc did_i(a^i_j))$. From $(r',m') \sim_{\sigma(Ags)} (r,m)$, it follows that $r'(m') = (t,\alpha)$ for some state t of E. Thus, from what was noted above, $\mathcal{I},(r',m') \models E\bigcirc did_i(a^i_j)$ iff $a^i_j \in \alpha_i(t)$. Since α is an implementation of P in E, this holds iff $\mathcal{I}_\alpha,(r',m') \models \phi^i_j$. By the claim proved above, $\mathcal{I}_\alpha,(r',m') \models \phi^i_j$ is equivalent to $\mathcal{I},(r',m') \models (\phi^i_j)^\$$. Thus, we have that $\mathcal{I},(r',m') \models (\phi^i_j)^\$ \Leftrightarrow E\bigcirc did_i(a^i_j)$. It follows that $\mathcal{I},(r,m) \models \mathbf{imp}(P)$.

Conversely, suppose that $\mathcal{I},(r,m) \models \mathbf{imp}(P)$, and let $r(m) = (s,\alpha)$. We show that α is an implementation of P in E. Let t be any α-reachable state, with, in particular, (r',m') a point of $\mathcal{I}_\alpha$ with $r'(m') = (t,\alpha)$. We need to show that for all agents i, we have

$$\alpha_i(t) = \left\{a^i_j \mid 1 \leq j \leq n_i,\ \mathcal{I}_\alpha, t \models \phi^i_j\right\},$$

i.e., that for all i,j we have $a^i_j \in \alpha_i(t)$ iff $\mathcal{I}_\alpha, t \models \phi^i_j$. Note that $(r,m) \sim_{\sigma(Ags)} (r',m')$, so we have that

$$\mathcal{I},(r',m') \models \bigwedge_{i\in Ags, j=1\ldots n_i} \left((\phi^i_j)^\$ \Leftrightarrow E\bigcirc did_i(a^i_j)\right).$$

As in the previous paragraph, $a^i_j \in \alpha_i(t)$ iff $\mathcal{I},(r',m') \models E\bigcirc did_i(a^i_j)$, which is equivalent to $\mathcal{I},(r',m') \models (\phi^i_j)^\$$, and by the claim proved above, equivalent to $\mathcal{I}_\alpha,(r',m') \models \phi^i_j$, i.e., $\mathcal{I}_\alpha, t \models \phi^i_j$. Thus, $a^i_j \in \alpha_i(t)$ iff $\mathcal{I}_\alpha, t \models \phi^i_j$, for all i,j, which is what we needed to prove. □

In particular, as a consequence of this result, it follows that several properties of knowledge-based programs (that do not make use of common knowledge operators) can be expressed in the system $\mathcal{I}(E,\Sigma^{unif}(E))$:

(1) The statement that there exists an implementation of P in E can be expressed by

$$\mathcal{I}(E,\Sigma^{unif}(E)) \models \neg D_\emptyset \neg \mathbf{imp}(P).$$

(2) The statement that all implementations of P in E guarantee that formula ϕ of CTL*K$(Ags, Prop)$ (which may contain knowledge operators) holds at all times can be expressed by

$$\mathcal{I}(E,\Sigma^{unif}(E)) \models D_\emptyset(\mathbf{imp}(P) \Rightarrow \phi^\$).$$

We remark that as a consequence of these encodings and Theorem 4.5 (in Section 4 below) that CTL*K$(Ags \cup \sigma(Ags), Prop)$ model checking in strategy space is in PSPACE, we obtain the following result:

Corollary 1. *The following are in PSPACE:*

(1) Given a finite environment E and a knowledge-based program P, determine if P has an implementation in E.
*(2) Given a finite environment E and a knowledge-based program P and a CTL*K$(Ags, Prop)$ formula ϕ, determine if $\mathcal{I}_\alpha \models \phi$ for all implementations α of P in E.*

For testing existence (part 1 of Corollary 1), this result was known [20], but the result on verification (part 2 of Corollary 1) has not previously been noted (though it could also have been shown using the techniques in Reference [20].)

One might expect that Proposition 4 can be extended to knowledge-based programs in which formulas may contain common knowledge operators, simply by adding the condition

$$(C_G\phi)^\$ = C_{G\cup\sigma(Ags)}\,\phi^\$$$

to the transformation of formulas. However, this does not work, because the interpretation of $C_G\phi$ in a subsystem $\mathcal{I}_\alpha$ is based on chains of points $(r_0, m_0) \sim_{i_1} (r_1, m_1) \sim_{i_2} \ldots \sim_{i_k} (r_k, m_k)$, such that r_j is a run of joint strategy α for all $j = 1 \ldots k$. By contrast, the semantics of $C_{G\cup\sigma(Ags)}\phi^\$$ in $\mathcal{I}$ involves chains of points that are not required to preserve the joint strategy: Rather, each step preserves the local state of one of the agents in G *or* the strategy of one of the agents. Neither does it work to use the translation

$$(C_G\phi)^\$ = \exists x(\mathsf{e}_{\sigma(Ags)}(x) \land C_G(\mathsf{e}_{\sigma(Ags)}(x) \Rightarrow \phi^\$)),$$

since the operator C_G similarly does not preserve the joint strategy, and it is not enough to test only at the end of the chain that the joint strategy has been preserved.

It is not clear that the translation we require for common knowledge is expressible in ESL. What would work is to generalize the common knowledge operator to the form $C_X\phi$, where X is a set of sets of agents (instead of a set of agents) and to define the semantics of this more general form as the greatest fixpoint of equation

$$C_X\phi = \bigwedge_{G\in X} D_G(\phi \land C_X\phi).$$

We could then use the translation

$$(C_G\phi)^\$ = C_{\{\{i\}\cup\sigma(Ags)\,|\,i\in G\}}\,\phi^\$.$$

Here the semantics involves chains of points in which we preserve the joint strategy and one of the agents in G. While this is an interesting extension, that we consider worthy of study, we do not pursue this as an ad hoc extension here, leaving it for future consideration in a broader context, such as a logic that extends ESL by mu-calculus operators.

4 MODEL CHECKING

Model checking is the problem of computing whether a formula of a logic holds in a given model. We now consider the problem of model-checking ESL and various of its fragments.

The model-checking problem is to determine whether $\Gamma, E, \Sigma \models \phi$ for a finite-state environment E, a set Σ of strategies and a context Γ, where ϕ is an ESL formula.

For purposes of results concerning the complexity of model checking, we need a measure of the size of a finite environment. Conventionally, the size of a model is taken to be the length of a string that lists its components, and, typically, this is polynomial in the number of states of the model. We note that in the case of environments, the set of labels *Acts* of the transition relation is an n-fold Cartesian product, where $n = Ags$, so (if the number of agents is a variable in the class of environments we consider) the size of the transition relation may be exponential in the number of agents.[9]

However, there is a more severe issue with respect to the parameter Σ of the model-checking problem. A strategy for a single agent is a mapping from states to sets of actions of the agent. Hence the number of strategies we may need to list to describe Σ explicitly could be exponential

[9]For certain classes of environments, we could address this by allowing that the transition relation $\rightarrow$ is presented in some notation with the property that (1) given states s, t and a joint action a, the representation of $\longrightarrow$ has size polynomial in the size of $|S|$ and $|Acts|$, and (2) determining whether $s \xrightarrow{a} t$ is in PTIME given s, a, t and the representation of $\longrightarrow$. One example of a presentation format with this property is the class of *turn-based environments*, where at each state s, there exists an agent i such that if $s \xrightarrow{a} t$ for a joint action a, then for all joint actions b with $a_i = b_i$ we have $s \xrightarrow{b} t$. That is, the set of states reachable in a single transition from s depends only on the action performed by agent i. In this case, the transition relation can be presented more succinctly as a subset of $S \times (\cup_{i\in Ags} A_i) \times S$. While it would be interesting to consider the effect of such optimized representations on our complexity results, we do not pursue this here.

in the number of *states* of the environment, even in the case of a single agent. To address this issue, we abstract the strategy set Σ to a parameterized class such that for each environment E, the set $\Sigma(E)$ is a set of strategies for E. When C is a complexity class, we say that the parameterized class Σ *can be presented in* C, if the problem of determining, given an environment E and a joint strategy α for E, whether $\alpha \in \Sigma(E)$, is in complexity class C. For example, the class Σ of all strategies for E can be PTIME-presented, as can Σ^{unif}, Σ^{det}, and $\Sigma^{unif,det}$.

We first consider the complexity of model checking the full language ESL. The following result gives an upper bound of EXPSPACE for this problem.

Theorem 4.1. *Let Σ be a parameterized class of strategies that can be presented in EXPSPACE. The complexity of deciding, given an environment E, an ESL formula ϕ and a context Γ for $\mathcal{I}(E, \Sigma(E))$, defined on the free variables of ϕ, whether $\Gamma, E, \Sigma(E) \models \phi$, is in EXPSPACE.*

Proof. The problem can be reduced to that of model checking the temporal epistemic logic CTL*K obtained by omitting the constructs $\exists$ and $\mathsf{e}_i(x)$ from the language ESL. This is known to be PSPACE-complete.[10] The reduction involves an exponential blowup of size of both the formula and the environment, so we obtain an EXPSPACE upper bound.

Model checking for temporal epistemic logic takes as input a formula and a structure that is like an environment, except that its transitions are not based on a set of actions for the agents. More precisely, an *epistemic transition system* for a set of agents Ags is a tuple $\mathcal{E} = \langle S, I, \rightarrow, \{O_i\}_{i \in Ags}, \pi\rangle$, where S is a set of states, $I \subseteq S$ is the set of initial states, $\rightarrow \subseteq S \times S$ is a state transition relation, for each $i \in Ags$, component $O_i : S \rightarrow L_i$ is a function giving an observation in some set L_i for the agent i at each state, and $\pi : S \rightarrow \mathcal{P}(Prop)$ is a propositional assignment. A *run* of $\mathcal{E}$ is a sequence $r : \mathbb{N} \rightarrow S$ such that $r(0) \in I$ and $r(k) \rightarrow r(k+1)$ for all $k \in \mathbb{N}$. To ensure that every partial run can be completed to a run, we assume that the transition relation is *serial*, i.e., that for all states s there exists a state t such that $s \rightarrow t$.

Given an epistemic transition system $\mathcal{E}$, we define an interpreted system $\mathcal{I}(\mathcal{E}) = (\mathcal{R}, \pi')$ as follows. For a run $r : \mathbb{N} \rightarrow S$ of $\mathcal{E}$, define the lifted run $\hat{r} : \mathbb{N} \rightarrow S \times \Pi_{i \in Ags} L_i$ (here $L_e = S$), by $\hat{r}_e(m) = r(m)$ and $\hat{r}_i(m) = O_i(r(m))$ for $i \in Ags$. Then we take $\mathcal{R}$ to be the set of lifted runs $\hat{r}$ with r a run of $\mathcal{E}$. The assignment π' is given by $\pi'(r, m) = \pi(r(m))$. The model-checking problem for temporal epistemic logic CTL*K is to decide, given an epistemic transition system $\mathcal{E}$ and a formula $\phi \in$ CTL*K, whether $\mathcal{I}(\mathcal{E}), (r, 0) \models \phi$ for all runs r of $\mathcal{I}(\mathcal{E})$.

We now show how to reduce ESL model checking to CTL*K model checking. Given an environment $E = \langle S, I, Acts, \rightarrow, \{O_i\}_{i \in Ags}, \pi\rangle$ for ESL$(Ags, Prop, Var)$, we first introduce a set of new propositions $Prop^* = \{p_{(s,\alpha)} \mid s \in S,\ \alpha \in \Sigma(E)\}$, which will be interpreted at global states of the generated interpreted system. Each proposition $p_{(s,\alpha)}$ will be true only at the global state (s, α). These propositions will help to eliminate the constructs $\mathsf{e}_i(x)$ and $\exists x$. We then define the epistemic transition system $\mathcal{E} = \langle S^*, I^*, \rightarrow^*, \{O^*_i\}_{i \in Ags}, \pi^*\rangle$ for the language CTL*K$(Ags \cup \sigma(Ags), Prop \cup Prop^*, Var)$, in which the propositions have been extended by the set $Prop^*$, as follows:

(1) $S^* = \{(s, \alpha) \in S \times \Sigma(E) \mid s$ is reachable in E using $\alpha\}$,
(2) $I^* = I \times \Sigma(E)$,
(3) $(s, \alpha) \rightarrow^* (t, \beta)$ iff $s \xrightarrow{a} t$ (in E) for some joint action a and $\beta = \alpha$,
(4) $O^*_i(s, \alpha) = O_i(s)$ and $O^*_{\sigma(i)}(s, \alpha) = \alpha_i$, for $i \in Ags$,
(5) $\pi^*(s, \alpha) = \pi(s) \cup \{p_{(s,\alpha)}\}$.

[10] The result is stated explicitly in Reference [18], but techniques sufficient for a proof (involving guessing a labelling of states by knowledge subformulas in order to reduce the problem to LTL model checking and also verifying the guess by LTL model checking) were already present in Reference [56]. The branching operator A can be treated as a knowledge operator for purposes of the proof.

We can treat the states $(s, \alpha) \in S^*$ as tuples indexed by $Ags \cup \sigma(Ags) \cup \{e\}$ by taking $(s, \alpha)_i = O_i(s)$ and $(s, \alpha)_{\sigma(i)} = \alpha_i$ for $i \in Ags$, and $(s, \alpha)_e = s$.

Note that a joint strategy for an environment E can be represented in space $\Sigma_{i \in Ags}|S| \times |Acts_i|$, and the number of strategies is exponential in the space requirement. Thus, the size of $\mathcal{E}$ is $O(2^{poly(|E|)})$. Note also that the construction of $\mathcal{E}$ can be done in EXPSPACE so long as verifying whether an individual strategy α is in $\Sigma(E)$ can be done in EXPSPACE.

We also need a transformation of the formula. Given a formula ϕ of ESL and a context Γ for E, we define a formula ϕ^Γ, inductively, by

(1) $p^\Gamma = p$, for $p \in Prop$,
(2) $\mathsf{e}_i(x)^\Gamma = \bigvee\{p_g \mid g \in S^*, g_i = \Gamma(x)_i\}$
(3) $(\neg\phi)^\Gamma = \neg\phi^\Gamma$, $(\phi_1 \wedge \phi_2)^\Gamma = \phi_1^\Gamma \wedge \phi_2^\Gamma$,
(4) $(\bigcirc\phi)^\Gamma = \bigcirc(\phi^\Gamma)$, $(\phi_1 U \phi_2)^\Gamma = (\phi_1^\Gamma) U (\phi_2^\Gamma)$, $(A\phi)^\Gamma = A(\phi^\Gamma)$
(5) $(D_G\phi)^\Gamma = D_G\phi^\Gamma$, $(C_G\phi)^\Gamma = C_G\phi^\Gamma$,
(6) $\exists x(\phi)^\Gamma = \bigvee\{\phi^{\Gamma[g/x]} \mid g \in S^*\}$.

Plainly, the size of ϕ^Γ is $O(2^{poly(|E|,|\phi|)})$, and this formula is in CTL*K$(Ags \cup \sigma(Ags), Prop \cup Prop^*)$. A straightforward inductive argument based on the semantics shows that

$\Gamma, E, \Sigma(E) \models \phi$ iff $\mathcal{I}(\mathcal{E}) \models \phi^\Gamma$. It therefore follows from the fact that model checking CTL*K with respect to the observational semantics for knowledge is in PSPACE that ESL model checking is in EXPSPACE. □

The following result shows that a restricted version of the model-checking problem, where we consider systems with just one agent and uniform deterministic strategies is already EXPSPACE hard.

Theorem 4.2. *The problem of deciding, given an environment E for a single agent, and an ESL sentence ϕ, whether $E, \Sigma^{unif,\det}(E) \models \phi$, is EXPSPACE-hard.*

Proof. We show how polynomial size inputs to the problem can simulate exponential space deterministic Turing machine computations. Let $T = \langle Q, q_0, q_f, q_r, A_I, A_T, \delta \rangle$ be a one-tape Turing machine solving an EXPSPACE-complete problem, with states Q, initial state q_0, final (accepting) state q_f, final (rejecting) state q_r, input alphabet A_I, tape alphabet $A_T \supseteq A_I$, and transition function $\delta : Q \times A_T \to Q \times A_T \times \{L, R\}$. We assume that T runs in space $2^{p(n)} - 2$ for a polynomial $p(n)$, and that the transition relation is defined so that the machine idles in its final state q_f on accepting and idles in state q_r on rejecting. The tape alphabet A_T is assumed to contain the blank symbol $\perp$.

Define $C_{T,Q} = A_T \cup (A_T \times Q)$ to be the set of "cell-symbols" of T. We may represent a configuration of T as a finite sequence over the set $C_{T,Q}$, containing exactly one element (x, q) of $A_T \times Q$, representing a cell containing symbol x where the machine's head is positioned, with the machine in state q. For technical reasons, we pad configurations with a blank symbol to the left and right (so configurations take space $2^{p(n)}$), so that the initial configuration has the head at the second tape cell and, without loss of generality, assume that the machine is designed so that it never moves the head to the initial or final padding blank. This means that the transition function δ can also be represented as a set of tuples $\Delta \subseteq C_{T,Q}^4$, such that $(a, b, c, d) \in \Delta$ iff, whenever the machine is in a configuration with a, b, c at cells at positions $k-1, k, k+1$, respectively, the next configuration has d at the cell at position k.

Given the TM T and a number $N = p(n)$ (for some polynomial p) we construct an environment $E_{T,N}$ such that for every input word w, with $|w| = n$, there exists a sentence ϕ_w of size polynomial in n such that $E_{T,N}, \Sigma^{unif}(E_{T,N}) \models \phi_w$ iff T accepts w. The idea of the simulation, depicted in Figure 1, is to represent a computation of the Turing machine, using space 2^N, by representing

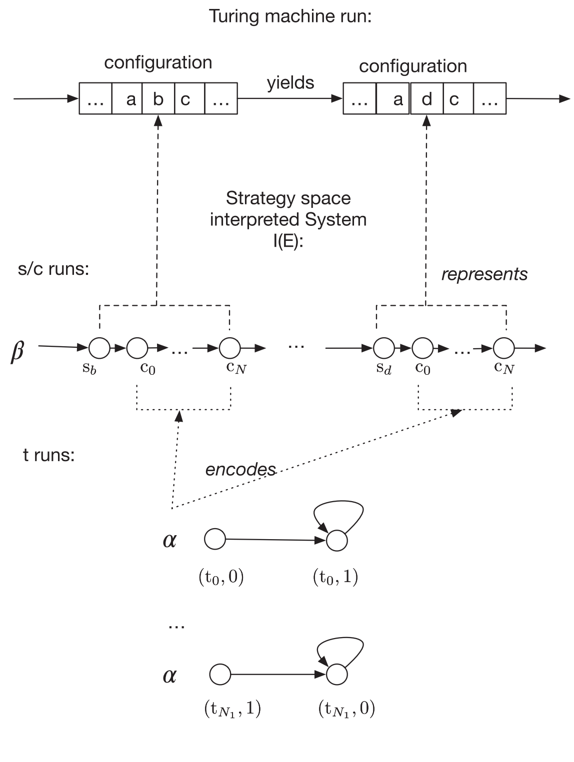


Fig. 1. Structure of the encoding.

the sequence of configurations of T for the computation consecutively along a run r of the environment $E_{T,N}$. (These runs travese a set of states we call s/c-states.) Each cell of a configuration will be encoded as a block of $N+1$ consecutive moments of time in r. In a block, the first of these moments represents the cell-symbol of the cell, and the remaining N moments represent the position of the cell in the configuration, in binary. Not all runs of $E_{T,N}$ will correctly encode a computation of the machine, so we use the formula to check whether a computation of T has been correctly encoded in a given run of E_T. To do so, the main difficulty is to check that corresponding cells of successive configurations represented along a run are updated correctly according to the yields relation of the Turing machine. For this, we need to be able to identify these corresponding cells, i.e. the cells with the same position number in the binary representation. For this, we use the behaviour of a strategy on an additional set of states (t-states) to give an alternate representation of a binary number, one that may be accessed in a formula by means of existential quantification. The formula then compares the representations of the binary number at two locations in the the s/c-run with the representation of the binary number in the strategy, to check that the numbers represented at the two locations in the s/c-run are the same. Details are given below.

The environment E has propositions $C_{T,Q} \cup \{\mathsf{c}\} \cup \{t_0, \ldots t_{N-1}\}$. Propositions from $C_{T,Q}$ are used to represent cell elements, and c is used to represents the bits of a counter that indicates the position of the cell being represented. In particular, a cell in a configuration, at position $b_{N-1} \ldots b_0$, in binary, and containing symbol $a \in C_{T,Q}$, will be represented by a sequence of $N+1$ states, the first of which satisfies proposition a, such that for $i = 0 \ldots N-1$, element $i+2$ of the sequence satisfies c iff $b_i = 1$. (Thus, low order bits are represented to the left in the run.)

We take the set of states of the environment to be

$$S = \{s_x \mid x \in C_{T,Q}\} \cup \{c_0, c_1\} \cup \{(t_i, j) \mid i = 0 \ldots N-1,\ j \in \{0,1\}\}.$$

The set of initial states of the environment is defined to be $I = \{s_\perp\} \cup \{(t_i, j) \mid i = 0 \ldots N-1,\ j \in \{0,1\}\}$. We define the assignment π so that $\pi(s_a) = \{a\}$ for $a \in C_{T,Q}$, $\pi(c_0) = \emptyset$, $\pi(c_1) = \{\mathsf{c}\}$ and $\pi((t_i, 0)) = \{t_i\}$ and $\pi((t_i, 1)) = \{t_i\} \cup \{\mathsf{c}\}$.

We take the set of actions of the single agent to be the set $\{\mathsf{a}_0, \mathsf{a}_1\}$. The transition relation $\rightarrow$ is defined so that for the only transitions are

$$\begin{array}{l} s_x \xrightarrow{a_k} c_i \\ c_i \xrightarrow{a_k} c_j \\ c_i \xrightarrow{a_k} s_x \\ (t_m, j) \xrightarrow{a_k} (t_m, k) \end{array}$$

for $x \in C_{T,Q}$ and $i, j, k \in \{0,1\}$ and $m \in \{0 \ldots N-1\}$. Intuitively, this forces the runs starting at state $s_\perp$ to alternate between selecting a symbol from $C_{T,Q}$ and a sequence of bits $\{0,1\}$ for the counter. Note that for every sequence ρ in $\perp \cdot \{c_0, c_1\}^+ \cdot (C_{T,Q} \cdot \{c_0, c_1\}^+)^\omega$, and for every strategy α for the single agent, there exists a run r with $r_{\sigma(1)} = \alpha$ and $r_e[0\ldots] = \rho$. For each $i = 0, \ldots N-1$, the states of the form (t_i, j) for $j \in \{0,1\}$ form an isolated component in the transition relation and are used to ensure that there is a sufficiently rich set of strategy choices for strategies to encode counter values.

The length of the counter sequence segments of a run generated by this transition system can vary within the run, but we can use a formula of length $O(N)$ to state that these segments always have length N wherever they appear in the run; let ϕ^N_{clock} be the formula

$$\Box\left(\alpha_{T,Q} \Rightarrow \left(\bigcirc^{N+1}\alpha_{T,Q} \wedge \bigwedge_{i=1\ldots N} \bigcirc^i \neg\alpha_{T,Q}\right)\right),$$

where we write $\alpha_{T,Q}$ for $\bigvee_{x \in C_{T,Q}} x$. By definition of the transition relation, this formula holds on a run starting in state $s_\perp$ just when it consists of states of the form s_x alternating with sequences of states of the form c_i of length exactly N.

The transition system generates arbitrary such sequences of states c_i of length N, intuitively constituting a guess for the correct counter value. Note that a temporal formula of length $O(N^2)$ can say that these guesses for the counter values are correct, in that the counter values encoded along the run are $0, 1, 2, \ldots 2^N - 1, 0, 1, 2, \ldots 2^N - 1$ (etc.). Specifically, this is achieved by the following formula ϕ^N_{count}:

$$\phi_{zero} \ \wedge \Box\left(\alpha_{T,Q} \Rightarrow \left(\begin{array}{l} (\phi_{max} \Rightarrow \bigcirc^{N+1}(\phi_{zero}))\wedge \\ \bigwedge_{i=1\ldots N}((\bigcirc c \wedge \ldots \bigcirc^{i-1} c \wedge \bigcirc^i \neg c) \Rightarrow \\ \bigcirc^{N+1}(\bigcirc\neg c \wedge \ldots \wedge \bigcirc^{i-1}\neg c \wedge \bigcirc^i c) \\ \wedge \bigwedge_{j=i+1\ldots N}((\bigcirc^j c) \Leftrightarrow (\bigcirc^{j+N+1} c))) \end{array}\right)\right),$$

where $\phi_{zero} = \bigwedge_{i=1\ldots N} \bigcirc^i \neg c$ and $\phi_{max} = \bigwedge_{i=1\ldots N} \bigcirc^i c$. Intuitively, the first line of the inner formula handles the steps from $2^N - 1$ to 0, and the remainder of the inner formula uses the fact that, in binary, $x01^i + 1 = x10^i$. (Recall that in the run, low order bits are represented to the left.)

The following formula ϕ^w_{init} then says that the run is initialized with word $w = a_1 \ldots a_n$

$$\perp \wedge \bigcirc^{N+1}((q_0, a_1) \wedge \bigcirc^{N+1}(a_2 \wedge \bigcirc^{N+1}(\ldots \bigcirc^{N+1}(a_n \wedge \bigcirc((\alpha_{T,Q} \Rightarrow (\perp \wedge \neg\phi_{zero}))U(\alpha_{T,Q} \wedge \phi_{zero})))\ldots)),$$

where $\perp$ is the blank symbol. This formula has size $O(N \cdot |w|) = O(p(n) \cdot n)$. Intuitively, the formula says that the sequence of symbols w is followed by a sequence of $\perp$ symbols until the first time that the counter has value zero (this corresponds to the start of the second configuration).

We now need a formula that expresses that whenever we consider two consecutive configurations C, C' encoded in a run, C' is derived from C by a single step of the TM T. The padding blanks are easily handled by the following formula ϕ_{pad}:

$$\Box((\alpha_{T,Q} \wedge (\phi_{zero} \vee \phi_{max})) \Rightarrow \perp).$$

For the remaining cell positions, we need to express that for each cell position $k = 1 \ldots 2^N - 2$, the cell value at position k in C' is determined from the cell value at positions $k-1, k, k+1$ in C according to the transition relation encoding Δ. This means that we need to be able to identify the corresponding positions k in C and C'. To capture the counter value at a given position in the run, we represent counter values using a strategy for the single agent, as follows.

We define the observation function O_1 for the single agent in $E_{T,N}$, so that observation $O_1((t_i, j)) = i$ for $i = 0 \ldots N-1$. (The values of the observation function on other states are not used in the encoding, and can be defined arbitrarily.) The number with binary representation $B = b_{N-1} \ldots b_0$ can then be represented by the strategy α_B such that $\alpha_B(t_i, j) = a_{b_i}$, for $i = 0 \ldots N-1$ and $j \in \{0, 1\}$ and $\alpha_B(s) = a_0$ for all other states s. (Note that this strategy is uniform, and, conversely, for any uniform strategy α there exists a unique binary number $b_{N-1} \ldots b_0$ such that $\alpha_B(t_i, j) = a_{b_i}$, for $i = 0 \ldots N-1$ and $j \in \{0, 1\}$.) Comparing this representation with the encoding of numbers along runs, the following formula $\phi_{num}(x)$ expresses that the number encoded at the present position in the run is the same as the number encoded in the strategy of agent 1 in the global state denoted by variable x:

$$\alpha_{T,Q} \wedge \bigwedge_{i=0\ldots N-1} (\bigcirc^{i+1} c) \Leftrightarrow \neg D_{\emptyset} \neg(\mathsf{e}_{\sigma(1)}(x) \wedge t_i \wedge \bigcirc c).$$

Note that, by the definition of the transition system, the value of $\bigcirc c$ at a state where t_i holds encodes whether the strategy selects a_0 or a_1 on observation $i = 0 \ldots N-1$. Note also that since all states of the form (t_i, j) are initial, for every strategy α, the value of $\alpha(t_i, j)$ is represented in this way at some point of some run. We may now check that the transitions of the TM are correctly computed along the run by means of the following formula ϕ_{trans}:

$$\Box\begin{pmatrix} \bigwedge_{(a,b,c,d)\in\Delta}(a \wedge \neg\phi_{max} \wedge \bigcirc^{N+1}(b \wedge \neg\phi_{max} \wedge \bigcirc^{N+1} c)) \Rightarrow \\ \bigcirc^{N+1}\exists x\, [\phi_{num}(x) \wedge \bigcirc ((\neg\phi_{num}(x)) U (\phi_{num}(x) \wedge d)] \end{pmatrix}.$$

Intuitively, here x captures the number encoded at the cell containing the symbol b, and the U operator is used to find the next occurrence in the run of this number. The occurrences of ϕ_{max} ensure that the three positions considered in the formula do not span across a boundary between two configurations. In Figure 1, the bottom part represents a strategy α of agent 1 encoded in some global state x. The behaviour of this strategy at the t-state runs represents a number, using the statement $\mathsf{e}_{\sigma(1)}(x)$ in the formula ϕ_{num}. The formula ϕ_{num} is used to assert that this representation of a binary number in α encodes the counter values at a position in a run. Asserting that two positions have the same counter number by this device allows us to check the yields relation at corresponding positions in the run representation of a computation of the Turing machine.

To express that the machine accepts, we just need to assert that the accepting state is reached; this is done by the formula $\phi_{accept} = \Diamond \bigvee_{a\in A_T}(a, q_f)$.

Combining these pieces, we get that the TM accepts input w if and only if

$$E_{T,N} \models (\phi^N_{clock} \wedge \phi^N_{count} \wedge \phi^w_{init} \wedge \phi_{trans}) \Rightarrow \phi_{accept}$$

holds, i.e., when every run that correctly encodes a computation of the machine is accepting. □

Combining Theorem 4.1 and Theorem 4.2 we obtain the following characterization of the complexity of ESL model checking.

Corollary 2. *Let Σ be an EXPSPACE presented class of strategies for environments, containing $\Sigma^{unif,\det}$. The complexity of deciding, given an environment E, an ESL formula ϕ and a context Γ for the free variables in an ESL formula ϕ relative to E and $\Sigma(E)$, whether $\Gamma, E, \Sigma(E) \models \phi$, is EXPSPACE-complete.*

The high complexity for ESL model checking motivates the consideration of fragments that have lower model-checking complexity. We demonstrate two orthogonal fragments for which the complexity of model checking is in a lower complexity class. One is the fragment ESL$^-$, where we allow the operators $\exists x.\phi$ and $\mathsf{e}_i(x)$ but restrict the use of the temporal operators to be those of the branching-time temporal logic CTL. In this case, we have the following result:

Theorem 4.3. *Let Σ be a PSPACE-presented class of strategies. The problem of deciding, given an environment E, a formula ϕ of ESL$^-$, and a context Γ for the free variables of ϕ relative to E and $\Sigma(E)$, whether $\Gamma, E, \Sigma(E) \models \phi$, is in PSPACE.*

Proof. We observe that the following fact follows straightforwardly from the semantics for formulas ϕ of ESL$^-$: For a context Γ for the free variables of ϕ relative to E and $\Sigma(E)$, and for two points (r, n) and (r', n') of $\mathcal{I}(E, \Sigma(E))$ with $r(n) = r'(n')$, we have that $\Gamma, \mathcal{I}(E, \Sigma(E)), (r, n) \models \phi$ iff $\Gamma, \mathcal{I}(E, \Sigma(E)), (r', n') \models \phi$. That is, satisfaction of a formula relative to a context at a point depends only on the global state at the point and not on other details of the run containing the point. For a global state (s, α) of $\mathcal{I}(E, \Sigma(E))$, define the Boolean $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ to be TRUE just when $\Gamma, \mathcal{I}(E, \Sigma(E)), (r, n) \models \phi$ holds for some point (r, n) of $\mathcal{I}(E, \Sigma(E))$ with $r(n) = (s, \alpha)$. By the above observation, we have that $\Gamma, E, \Sigma(E) \models \phi$ iff $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ holds for all initial states s of E and all strategies $\alpha \in \Sigma(E)$. Since we may check these conditions one at a time, strategies α can be represented in space linear in $|E|$, and deciding $\alpha \in \Sigma(E)$ is in PSPACE, it suffices to show that $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ is decidable in PSPACE.

We proceed by describing an APTIME algorithm for $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ and using the fact that APTIME = PSPACE [12]. The algorithm operates recursively, with the following cases:

(1) If $\phi = p$, for $p \in Prop$, then return TRUE if $p \in \pi(s)$, else return FALSE.
(2) If $\phi = \mathsf{e}_i(x)$, then return TRUE if $(s, \alpha)_i = \Gamma(x)_i$, else return FALSE.
(3) If $\phi = \phi_1 \wedge \phi_2$, then universally call $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi_1)$ and $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi_2)$.
(4) If $\phi = \neg\phi_1$, then return the complement of $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi_1)$.
(5) If $\phi = A{\bigcirc}\phi_1$ then universally choose a state t such that $s \xrightarrow{a} t$ for some for some joint action a, and call $SAT(\Gamma, E, \Sigma, (t, \alpha), \phi_1)$. The other temporal operators from CTL are handled similarly. (In the case of operators using U, we need to run a search for a path through the set of states of E generated by the strategy α, but this is easily handled in APTIME.)
(6) If $\phi = D_G\phi_1$, then universally choose a global state (t, β) such that $(s, \alpha) \sim^D_G (t, \beta)$ and universally
    (a) decide whether $\beta \in \Sigma(E)$, and
    (b) call $REACH(t, \beta)$, and
    (c) call $SAT(\Gamma, E, \Sigma, (t, \beta), \phi_1)$.

(Here $REACH(t, \beta)$ decides whether state t is reachable in E from some initial state when the agents run the joint strategy β; this is trivially in PSPACE. Deciding $\beta \in \Sigma(E)$ is in PSPACE by the assumption that Σ is PSPACE-presented.)

(7) If $\phi = C_G\phi_1$, then universally guess a global state (t, β) and universally do the following:
  (a) Decide whether $(s, \alpha) \sim_G^C (t, \beta)$ using an existentially branching binary search for a path of length at most $|S| \times |\Sigma(E)|$. For all states (u, γ) on this path it should be verified that $REACH(u, \gamma)$ and that $\gamma \in \Sigma(E)$. The maximal length of the path is in the worst case exponential in $|E|$, but the binary search can handle this in APTIME.
  (b) call $SAT(\Gamma, E, \Sigma, (t, \beta), \phi_1)$.

(8) If $\phi = \exists x(\phi_1)$, then existentially guess a global state (t, β), and universally
  (a) decide if $\beta \in \Sigma(E)$, and
  (b) call $REACH(t, \beta)$, and
  (c) call $SAT(\Gamma[(t, \beta)/x], E, \Sigma, (s, \alpha), \phi_1)$.

A straightforward argument based on the semantics of the logic shows that the above correctly computes SAT.

We remark that a more efficient procedure for checking that $(s, \alpha) \sim_G^C (t, \beta)$ is possible in the typical case where $\Sigma(E)$ is a Cartesian product of sets of strategies for each of the agents. In this case, if there exists a witness chain then there is one of length at most $|S|$. Let $G = G_1 \cup \sigma(G_2)$ such that $G_1, G_2 \subseteq Ags$. The number of steps through the relation $\cup_{i \in G} \sim_i$ required to witness $(s, \alpha) \sim_G^C (t, \beta)$ depends on the sets G_1, G_2 as follows:

(1) If $G_1 = G_2 = \emptyset$, then we must have $(s, \alpha) = (t, \beta)$ and a chain of length 0 suffices.
(2) If G_1 is nonempty and $G_2 = \emptyset$, then we must have $s(\cup_{i \in G_1} \sim_i)^* t$, but β can be arbitrary, and this component can be changed in any step. A path of length $|S|$ suffices in this case.
(3) If $G_1 = \emptyset$ and $G_2 = \{i\}$ is a singleton, then we must have $\alpha_i = \beta_i$, but s and t can be arbitrary. A path of length one suffices in this case.
(4) If $|G_1| \geq 1$, say, $i \in G_1$, and $G_2 = \{j\}$ is a singleton, then $(\cup_{i \in G} \sim_i)^*$ is the universal relation and a path of length 2 suffices. In particular, for any $(s, \alpha), (t, \beta)$ we have $(s, \alpha) \sim_i (s, \beta) \sim_{\sigma(j)} (t, \beta)$.
(5) If $|G_2| \geq 2$, then $(\cup_{i \in G} \sim_i)^*$ is the universal relation and a path of length 2 suffices. In particular, for any $(s, \alpha), (t, \beta)$ and distinct $i, j \in G_2$, there exists α' such that $\alpha'_i = \alpha_i$ and $\alpha'_k = \beta_k$ for all $k \in Ags$ with $k \neq i$, and $(s, \alpha) \sim_{\sigma(i)} (s, \alpha') \sim_{\sigma(j)} (t, \beta)$. □

The following result shows that the PSPACE upper bound from this result is tight, already for formulas that use strategy indices in the CTLK operators, but make no direct uses of the constructs $\exists x$ and $\mathsf{e}_i(x)$.

Theorem 4.4. *The problem of deciding, given an environment E for two agents and a formula ϕ of CTLK($Ags \cup \sigma(Ags), Prop$), whether $E, \Sigma^{unif, det}(E) \models \phi$ is PSPACE hard.*

Proof. We proceed by a reduction from the satisfiability of Quantified Boolean Formulae (QBF). An instance of QBF is a formula ϕ of form

$$Q_1 x_1 \ldots Q_n x_n(\gamma),$$

where $Q_1, \ldots, Q_n \in \{\exists, \forall\}$ and γ is a formula of propositional logic over propositions $x_1, \ldots, x_n$. The QBF problem is to decide, given a QBF instance ϕ, whether it is true. We construct an environment E_ϕ and a formula ϕ^* of CTLK using strategic indices $\sigma(i)$ such that the QBF formula ϕ is true iff we have $E_\phi, \Sigma^{unif, det}(E_\phi) \models \phi^*$.

Given the QBF formula ϕ, we construct the environment $E_\phi = \langle S, I, \{Acts_i\}_{i \in Ags}, \rightarrow, \{O_i\}_{i \in Ags}, \pi\rangle$ for 2 agents $Ags = \{1, 2\}$ and propositions $Prop = \{p_0, \ldots, p_n, q_1, q_2\}$ as follows:

(1) The set of states $S = \{s_0\} \cup \{s_{t,j,k} | t \in \{1 \dots n\}, j, k \in \{0, 1\}\}$.
(2) The set of initial states is $I = \{s_0\}$.
(3) The actions of agent i are $A_i = \{0, 1\}$, for each $i \in Ags$.
(4) The transition relation is defined to consist of the following transitions, where $j, j', k, k' \in \{0, 1\}$

$$
\begin{array}{ll}
s_0 \xrightarrow{(j',k')} s_{1,j',k'} & \\
s_{t,j,k} \xrightarrow{(j',k')} s_{t+1,j',k'} & \text{for } t = 1 \dots n-1 \\
s_{n,j,k} \xrightarrow{(j',k')} s_{n,j,k}. &
\end{array}
$$

(5) Observations are defined so that $O_i(s_0) = 0$ and $O_i(s_{t,j,k}) = t$.
(6) The assignment π is defined by $\pi(s_0) = \{p_0\}$, and

$$\pi(s_{t,j,k}) = \{p_t\} \cup \{q_1 \mid j = 1\} \cup \{q_2 \mid k = 1\}$$

for $t = 1 \dots n$.

Intuitively, the model sets up $n + 1$ moments of time $t = 0, \dots, n$, with s_0 the only possible state at time 0 and $s_{t,j,k}$ for $j, k \in \{0, 1\}$ the possible states at times $t = 1, \dots, n$. Both agents observe only the value of the moment of time, so that for each agent, a strategy selects an action 0 or 1 at each moment of time. We may therefore encode an assignment to the proposition variables $x_1 \dots x_n$ by the actions chosen by an agent's strategy at times $0, \dots n - 1$. The action chosen by each agent at time $t \in \{0 \dots n - 1\}$ is recorded in the indices of the state at time $t + 1$, i.e. if the state at time $t + 1$ is $s_{t+1,j,k}$ then agent 1 chose action j at time t, and agent 2 chose action k.

We work with two agents, each of whose strategies is able to encode an assignment, to alternate between the two encodings. At each step, one of the strategies is assumed to encode an assignment to the variables $x_1, \dots x_m$. This strategy is fixed, and we universally or existentially guess the other strategy to obtain a new value for the variable x_{m+1}. We then check that the guess has maintained the values of the existing assignment to $x_1, \dots x_m$ by comparing the two strategies.

More precisely, let $val_i(x_j)$ be the formula $K_{\{\sigma(i)\}}(p_{j-1} \Rightarrow E\bigcirc(q_i))$ for $i = 1, 2$ and $j = 1 \dots n$. This states that at the current state, the strategy of agent i selects action 1 at time $j - 1$, so it encodes an assignment making x_j true. For $m = 1 \dots n$, let $agree(m)$ be the formula

$$\bigwedge_{j=1\dots m} D_{\{\sigma(1),\sigma(2)\}}(p_{j-1} \Rightarrow (E\bigcirc(q_1) \Leftrightarrow E\bigcirc(q_2))).$$

This says that the assignments encoded by the strategies of the two agents agree on the values of the variables $x_1 \dots, x_m$. Assuming, without loss of generality, that n is even, and that the quantifier sequence in ϕ is $(\exists\forall)^{n/2}$, given the QBF formula ϕ, define the formula ϕ^* to be

$$
\begin{array}{l}
\neg D_\emptyset \neg (D_{\{\sigma(1)\}}(agree(1) \Rightarrow \\
\qquad \neg D_{\{\sigma(2)\}} \neg (agree(2) \wedge \\
\qquad\qquad D_{\{\sigma(1)\}}(agree(3) \Rightarrow \\
\qquad\qquad\qquad \neg D_{\{\sigma(2)\}} \neg (agree(4) \wedge \dots \\
\qquad\qquad\qquad\qquad \vdots \\
\qquad\qquad\qquad\qquad D_{\{\sigma(1)\}}(agree(m-1) \Rightarrow \gamma^+) \dots)
\end{array}
$$

where γ^+ is the formula obtained by replacing each occurrence of a variable x_j in γ by the formula $val_2(x_j)$. Intuitively, the first operator $\neg D_\emptyset \neg$ existentially chooses a value for variable x_1, encoded in $\sigma(1)$, the next operator $D_{\{\sigma(1)\}}$ remembers this strategy while encoding a universal choice of value for variable x_2 in $\sigma(2)$, and the formula $agree(1)$ checks that the existing choice for x_1 is preserved in $\sigma(2)$. Continued alternation between the two strategies adds universal or existential choices for variable values while preserving previous choices. It can then be shown that the QBF formula ϕ is true iff $E_\phi, \Sigma^{unif, det} \models \phi^*$. □

Combining Theorem 4.3 and Theorem 4.4, we obtain the following:

Corollary 3. *Let Σ be a PSPACE-presented class of strategies. The problem of deciding if $\Gamma, E, \Sigma(E) \models \phi$, given an environment E, a formula ϕ of ESL^- and a context Γ for the free variables of ϕ relative to E and $\Sigma(E)$, is PSPACE complete.*

Since PSPACE is strictly contained in EXPSPACE, this result shows a strict improvement in complexity as a result of the restriction to the CTL-based fragment. We remark that, by a trivial generalization of the standard state labelling algorithm for model checking CTL to handle the knowledge operators, the problem of model checking the logic CTLK($Ags, Prop$) in the system $\mathcal{I}(\mathcal{E})$ generated by an epistemic transition system $\mathcal{E}$ is in PTIME. Thus, there is a jump in complexity from CTLK as a result of the move to the strategic setting, even without the addition of the operators $\exists x.\phi$ and $\mathsf{e}_i(x)$. However, this jump is not so large as the jump to the the full logic ESL.

An orthogonal restriction of ESL is to retain the CTL* temporal basis, i.e., to allow full use of LTL operators, but to allow epistemic operators and strategy indices but omit use of the operators $\exists x.\phi$ and $\mathsf{e}_i(x)$. This gives the logic CTL*K($Ags \cup \sigma(Ags), Prop$). For this logic, we also see an improvement in the complexity of model checking compared to full ESL, as is shown in the following result.

Theorem 4.5. *Let $\Sigma(E)$ be a PSPACE presented class of strategies for environments E. The complexity of deciding, given an environment E and a CTL*K formula ϕ for agents $Ags(E)^+ \cup \sigma(Ags(E))$, whether $E, \Sigma(E) \models \phi$, is PSPACE-complete.*

Proof. The lower bound is straightforward from the fact that linear time temporal logic LTL is a sublanguage of CTL*K, and model checking LTL is already PSPACE-hard [52]. For the upper bound, we describe an alternating PTIME algorithm and invoke the fact that APTIME = PSPACE [12]. We abbreviate $\mathcal{I}(E, \Sigma(E))$ to $\mathcal{I}$.

For a formula ϕ, write maxk(ϕ) for the maximal epistemic subformulas of ϕ, defined to be the set of subformulas of the form $A\psi$ or $C_G\psi$ or $D_G\psi$ for some set G of basic and strategic indices, which are themselves not a subformula of a larger subformula of ϕ of one of these forms. Note that $A\psi$ can be taken to be epistemic, because it is equivalent to $D_{\{e\}\cup\sigma(Ags)}\psi$; in the following, we assume that $A\psi$ is written in this form. Also note that for epistemic formulas ψ, satisfaction at a point depends only on the global state, i.e., for all points (r, m) and (r', m') of $\mathcal{I}$, we have that if $r(m) = r'(m')$ then $\mathcal{I}, (r, m) \models \psi$ iff $\mathcal{I}, (r', m') \models \psi$. Thus, for global states (s, α) of $\mathcal{I}$, we may write $\mathcal{I}, (s, \alpha) \models \psi$ to mean that $\mathcal{I}, (r, m) \models \psi$ for some point (r, m) with $r(m) = (s, \alpha)$.

Define a *ϕ-labelling* of E to be a mapping $L : S \times \text{maxk}(\phi) \rightarrow \{0, 1\}$, giving a truth value for each maximal epistemic subformula of ϕ. A ϕ-labelling can be represented in space $|S| \times |\phi|$. Note that if we treat the maximal epistemic subformulas of ϕ as if they were atomic propositions, evaluated at the states of E using the ϕ-labelling L, then ϕ becomes an LTL formula, evaluable on any path in E with respect to the labelling L. Verifying that all α-paths from a state s satisfy ϕ with respect to L is then exactly the problem of LTL model checking, for which there exists an APTIME procedure ASAT($E, (s, \alpha), L, \phi$), since model checking LTL is in PSPACE [52] and APTIME = PSPACE [12]. For this to correspond to model checking in $\mathcal{I}$, we require that the ϕ-labelling L gives the correct answers for the truth value of the formula at each state (s, α), i.e., that $L(\psi) = 1$ iff $\mathcal{I}, (s, \alpha) \models \psi$. We handle this by means of a guess and verify technique.

To handle the verification, an alternating PTIME algorithm KSAT($E, \Sigma, (s, \alpha), \phi$) is defined, for ϕ an epistemic formula, such that KSAT($E, \Sigma, (s, \alpha), \phi$) returns TRUE iff $\mathcal{I}, (s, \alpha) \models \phi$. The definition is recursive and uses a call to the procedure ASAT. Specifically, KSAT$(E, \Sigma, (s, \alpha), D_G\phi)$ operates as follows:

(1) universally guess a state t of E and a joint strategy β in E, then
(2) verify that t is reachable in E using joint strategy β, that $(s, \alpha) \sim_G (t, \beta)$, and that β is in $\Sigma(E)$, then
(3) existentially guess a ϕ-labelling L of E, then
(4) universally,
   (a) call ASAT$(E, (t, \beta), L, \phi)$, and
   (b) for each state w and formula $\psi \in$ maxk(ϕ), call KSAT$(E, \Sigma, (w, \beta), \psi)$.

Note that step 4(b) verifies that the ϕ-labelling L is correct.

For KSAT$(E, \Sigma, (s, \alpha), C_G\phi)$, the procedure is similar, except that instead of verifying that $(s, \alpha) \sim_G (t, \beta)$ in the second step, we need to verify that $(s, \alpha)\ (\cup_{i \in G} \sim_i)^*\ (t, \beta)$. This is easily handled in APTIME by a standard recursive procedure that guesses a midpoint of the path and branches universally to verify the existence of the left and right halves of the chain. (See the proof of Theorem 4.3 for some further discussion on this point.)

To solve the model-checking problem in $\mathcal{I}$, we can now apply the following alternating procedure:

(1) universally guess a global state (s, α) of $\mathcal{I}$, then branch existentially to the following cases:
   (a) if s is an initial state of E return FALSE, else return TRUE,
   (b) if $\alpha \in \Sigma(E)$, return FALSE, else return TRUE,
   (c) call KSAT$(E, \Sigma, (s, \alpha), A\phi)$.

Evidently, each of the alternating procedures runs in polynomial time internally, and the number of recursive calls is $O(|\phi|)$. It follows that the entire computation is in APTIME = PSPACE. □

It is interesting to note that, although CTL*K$(Ags \cup \sigma(Ags))$ is significantly richer than the temporal logic LTL, the added expressiveness comes without an increase in complexity: model checking LTL is already PSPACE-complete [52].

5 CONCLUSION

We now discuss some related work and remark on some questions for future research. The sections above have already made some references and comparisons to related work on each of the topics that we cover. Beside these references, the following are also worth mentioning.

Semantics that explicitly encode strategies in runs have been used previously in the literature on knowledge in information flow security [25]; what is novel in our approach is to develop a logic that enables explicit reference to these strategies.

A variant of propositional dynamic logic (PDL) for describing strategy profiles in normal form games subject to preference relations is introduced in Reference [54]. This work does not cover temporal aspects as we have done in this article. Another approach based on PDL is given in Reference [47], which describes strategies by means of formulas.

A very rich generalization of ATEL for probabilistic environments is described in Reference [49]. This proposal includes variables that refer to *strategy choices*, and strategic operators that may refer to these variables, so that statements of the form "when coalition A runs the strategy represented by variable S1, and coalition B runs the strategy represented by variable S2, and the remaining agents behave arbitrarily, then the probability that ϕ holds is at least δ" can be expressed. Here a strategy choice maps each state, coalition and formula to a uniform imperfect recall strategy for the coalition. There are a number of syntactic restrictions compared to our logic. The epistemic operators in this approach apply only to state formulas rather than path formulas (in the sense of this distinction from CTL*.) Moreover, the strategic variables may be quantified, but only in the prefix of the formula. These constraints imply that notions such as "agent i knows that there exists a strategy by which it can achieve ϕ" and "agent i knows that it has a winning response to every strategy chosen by agent j" cannot be naturally expressed.

The extended temporal epistemic logic ETLK we have introduced, of which our epistemic strategy logic ESL is an instantiation with respect to a particular semantics, uses constructs that resemble constructs from *hybrid logic* [4]. Hybrid logic is an approach to the extension of modal logics that uses "nominals," i.e., propositions p that hold at a single world. These can be used in combination with operators such as $\exists p$, which marks an arbitrary world as the unique world at which nominal p holds. Our construct $\exists x$ is closely related to the hybrid construct $\exists p$, but we work in a setting that is richer in both syntax and semantics than previous works. There have been a few works using hybrid logic ideas in the context of epistemic logic [27, 48] but none are concerned with temporal logic. Hybrid temporal logic has seen a larger amount of study [5, 21, 22, 51], with variances in the semantics used for the model-checking problem.

We note that if we were to view the variable x in our logic as a propositional constant, it would be true at a set of points in the system $\mathcal{I}(E, \Sigma)$, hence not a nominal in that system. Results in Reference [5], where a hybrid linear time temporal logic formula is checked in all paths in a given model, suggest that a variant of ESL in which x is treated as a nominal in $\mathcal{I}(E, \Sigma)$ would have a complexity of model checking at least non-elementary, compared to our EXPSPACE and PSPACE complexity results.

Our PSPACE model-checking result for CTLK($Ags \cup \sigma(Ags)$) seems to be more closely related to the result in Reference [21] that model checking a logic HL($\exists, @, F, A$) is PSPACE-complete. Here F is essentially a branching time future operator and A is a universal operator (similar to our $D_\emptyset$), the construct $@_p\phi$ says that ϕ holds at the world marked by the nominal p, and $\exists p(\phi)$ says that ϕ holds after marking some world by p. The semantics in this case does not unfold the model into either a tree or a set of linear structures before checking the formula, so the semantics of the hybrid existential $\exists$ is close to our idea of quantifying over global states. Our language, however, has a richer set of operators, even in the temporal dimension, and introduces the strategic dimension in the semantics. It would be an interesting question for future work to consider fragments of our language to obtain a more precise statement of the relationship with hybrid temporal logics.

Strategy Logic [13] is a (non-epistemic) generalization of ATL for perfect information strategies in which strategies may be explicitly named and quantified. Strategy logic has a non-elementary model-checking problem. Work on identification of more efficient variants of quantified strategy logic includes [42], who formulate a variant with a 2-EXPTIME-complete model-checking problem. In both cases, strategies are perfect recall strategies, rather than the imperfect recall strategies that form the basis for our PSPACE-completeness result for model checking.

Most closely related to this article are a number of independently developed works that consider epistemic extensions of variants of strategy logic. Belardinelli [3] develops a logic, based on linear time temporal logic with epistemic operators, that adds an operator $\exists x_i$, the semantics of which existentially modifies the strategy associated to agent i in the current strategy profile. It omits the binding operator from Reference [42], so provides no other way to refer to the variable x. The logic is shown to have nonelementary model-checking complexity. This complexity is higher than the results we have presented, because the semantics for strategies allows agents to have perfect information and perfect recall (though the semantics for the knowledge operators is based on imperfect information and no recall), whereas we have assumed imperfect information and no recall for strategies.

Another extension of strategy logic with epistemic operators has been independently developed by Čermák et al. [10, 11]. Their syntax and semantics differs from ours in a number of respects. Although the syntax appears superficially in the form of an extension of LTL, it is more like CTL in some regards. The transition relation is deterministic in the sense that for each joint action, each state has a unique successor when that action is performed. Strategies are also assumed to be deterministic (whereas we allow nondeterministic strategies.) This means that, like CTL, the semantics of a formula depends only on the current global state and the current strategy profile, whereas for LTL it is generally the case that the future structure of the run from a given global state can vary, and the truth value of the formula depends on how it does so. Although it seems that non-determinism could be modelled, as is commonly done, through the choice of actions of the environment, treated as an agent, the fact that strategies are deterministic, uniform and memoryless means that the environment must choose the same alternative each time a global state occurs in a run. This means that this standard approach to modelling of non-determinism does not work for this logic. The syntax of the logic moreover prevents epistemic operators from being applied to formulas with free strategy variables, whereas we allow fully recursive mixing of the constructs of our logic. Consequently, epistemic notions from our logic like $D_{\{i,\sigma(i)\}}$, expressing an agent's knowledge about the effects of its own strategy, which are used in several of our applications, do not appear to be expressible in this logic. Finally, the notion of "interpreted system" in this work, which corresponds most closely to our notion of "environment," also seems less general than our notion of environment, because it defines the accessibility relations for the knowledge operators in a way that makes the environment state known to all agents.

In another article [32], we have implemented a symbolic algorithm that handles model checking for the fragment CTLK($Ags \cup \sigma(Ags)$), which, as shown above, encompasses the expressiveness of ATEL. Existing algorithms described in the literature for ATEL model checking [8, 9, 40] are based either on explicit-state model checking or are only partially symbolic in that they iterate over all strategies, explicitly represented. Our experimental results in Reference [32] show that by comparison with the partially symbolic approach, a fully-symbolic algorithm can greatly improve the performance and therefore scalability of model checking. The approach to model-checking epistemic strategy logic implemented in Reference [10, 11] is fully symbolic, but as already mentioned, this logic has a more limited expressive power than ours and its semantics does not permit representation of a nondeterministic environment. (It does not seem that the semantics could be extended to allow nondeterminism while retaining correctness of their algorithm.)

Our focus on this article has been on an observational, or imperfect recall, semantics for knowledge. Other semantics for knowledge are also worth considering, but are left for future work. We note one issue in relation to the connection to ATEL that we have established, should we consider a perfect recall version of our logic. ATEL operators effectively allow reference to situations in which agents switch their strategy after some actions have already been taken, whereas in our model an agent's strategy is fixed for the entire run. When switching to a new strategy, there is the possibility that the given state is not reachable under this new strategy. We have handled this issue in our translation by assuming that all states are initial, so that the run can be reinitialized if necessary to make the desired state reachable. This is consistent with an imperfect recall interpretation of ATEL, but it is not clear that this approach is available on a perfect recall interpretation. We leave a resolution of this issue to future work.

REFERENCES

[1] Thomas Ågotnes, Valentin Goranko, and Wojciech Jamroga. 2007. Alternating-time temporal logics with irrevocable strategies. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'07)*. 15–24. DOI: http://dx.doi.org/10.1145/1324249.1324256

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-time temporal logic. *J. ACM* 49, 5 (2002), 672–713.

[3] Francesco Belardinelli. 2014. Reasoning about knowledge and strategies: Epistemic strategy logic. In *Proceedings of the 2nd International Workshop on Strategic Reasoning (SR'14)*. 27–33.

[4] Patrick Blackburn and Jerry Seligman. 1998. What are hybrid languages? In *Advances in Modal Logic*, M. de Rijke, H. Wansing, and M. Zakharyaschev (Eds.), Vol. 1. CSLI Publications, 41–62.

[5] Laura Bozzelli and Ruggero Lanotte. 2010. Complexity and succinctness issues for linear-time hybrid logics. *Theor. Comput. Sci.* 411, 2 (2010), 454–469. DOI: http://dx.doi.org/10.1016/j.tcs.2009.08.009

[6] Ronen I. Brafman, Jean-Claude Latombe, Yoram Moses, and Yoav Shoham. 1997. Applications of a logic of knowledge to motion planning under uncertainty. *J. ACM* 44, 5 (1997), 633–668.

[7] Thomas Brihaye, Arnaud Da Costa, François Laroussinie, and Nicolas Markey. 2009. ATL with strategy contexts and bounded memory. In *Proceedings of the International Symposium on Logical Foundations of Computer Science*. 92–106.

[8] Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. 2013. Reasoning about strategies under partial observability and fairness constraints. In *Proceedings of the 1st International Workshop on Strategic Reasoning (SR'13)*. 71–79.

[9] Jan Calta, Dmitry Shkatov, and Bernd-Holger Schlingloff. 2010. Finding uniform strategies for multi-agent systems. In *Computational Logic in Multi-Agent Systems (CLIMA XI)*. 135–152.

[10] Petr Čermák. 2014. *A Model Checker for Strategy Logic*. MEng Individual Project thesis, Department of Computing, Imperial College London.

[11] Petr Čermák, Alessio Lomuscio, Fabio Mogavero, and Aniello Murano. 2014. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Proceedings of the 26th International Conference on Computer-Aided Verification (CAV'14)*. 525–532.

[12] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. 1981. Alternation. *J. ACM* 28, 1 (1981), 114–133.

[13] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. 2010. Strategy logic. *Inf. Comput.* 208, 6 (2010), 677–693. DOI: http://dx.doi.org/10.1016/j.ic.2009.07.004

[14] Stephen Chong and Andrew C. Myers. 2005. Language-based information erasure. In *Proceedings of the IEEE Computer Security Foundations Workshop*. 241–254.

[15] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8, 2 (1986), 244–263.

[16] Catalin Dima and Ferucio Laurentiu Tiplea. 2011. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR* abs/1102.4225 (2011). Q6

[17] E. Allen Emerson and Joseph Y. Halpern. 1986. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM* 33, 1 (1986), 151–178.

[18] Kai Engelhardt, Peter Gammie, and Ron van der Meyden. 2007. Model checking knowledge and linear time: PSPACE cases. In *Proceedings of the Symposium on Logical Foundations of Computer Science*. LNCS. Springer, 195–211.

[19] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. 1995. *Reasoning about Knowledge*. The MIT Press.

[20] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. 1997. Knowledge-based programs. *Distrib. Comput.* 10, 4 (1997), 199–225.

[21] Massimo Franceschet and Maarten de Rijke. 2006. Model checking hybrid logics (with an application to semistructured data). *J. Appl. Logic* 4, 3 (2006), 279–304. DOI: http://dx.doi.org/10.1016/j.jal.2005.06.010

[22] Massimo Franceschet, Maarten de Rijke, and Bernd-Holger Schlingloff. 2003. Hybrid logics on linear structures: Expressivity and complexity. In *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning and the 4th International Conference on Temporal Logic (TIME-ICTL'03)*. 166–173.

[23] Joseph Y. Halpern and Yoram Moses. 1990. Knowledge and common knowledge in a distributed environment. *J. ACM* 37, 3 (1990), 549–587.

[24] Joseph Y. Halpern and Yoram Moses. 2007. Characterizing solution concepts in games using knowledge-based programs. In *Proceedings of the 20nd International Joint Conference on Artificial Intelligence (IJCAI'07)*. 1300–1307.

[25] Joseph Y. Halpern and Kevin R. O'Neill. 2008. Secrecy in multiagent systems. *ACM Trans. Inf. Syst. Secur.* 12, 1 (Article No. 5 2008).

[26] Joseph Y. Halpern and Nan Rong. 2010. Cooperative equilibrium (extended abstract). In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*. 1465–1466.

[27] Jens Ulrik Hansen. 2011. A hybrid public announcement logic with distributed knowledge. *Electr. Not. Theor. Comput. Sci.* 273 (2011), 33–50. DOI: http://dx.doi.org/10.1016/j.entcs.2011.06.011 Q7

[28] Wiebe van der Hoek, Wojciech Jamroga, and Michael Wooldridge. 2005. A logic for strategic reasoning. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. 157–164.

[29] Wiebe van der Hoek and Michael Wooldridge. 2002. Tractable multiagent planning for epistemic goals. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*. 1167–1174.

[30] John F. Horty. 2001. *Agency and Deontic Logic*. Oxford University Press.

[31] Xiaowei Huang and Ron van der Meyden. 2014. An epistemic strategy logic (extended abstract). In *Proceedings of the 2nd International Workshop on Strategic Reasoning (SR14)*. 35–41.

[32] Xiaowei Huang and Ron van der Meyden. 2014. Symbolic model checking epistemic strategy logic. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*. 1426–1432.

[33] Xiaowei Huang and Ron van der Meyden. 2014. A temporal logic of strategic knowledge. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*.

[34] Wojciech Jamroga. 2003. Some remarks on alternating temporal epistemic logic. In *Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS'03)*.

[35] Wojciech Jamroga and Thomas Ågotnes. 2007. Constructive knowledge: What agents can achieve under imperfect information. *J. Appl. Non-Class. Logics* 17, 4 (2007), 423–475. DOI: http://dx.doi.org/10.3166/jancl.17.423-475

[36] Wojciech Jamroga, Thomas Ågotnes, and Wiebe van der Hoek. 2008. *A Simpler Semantics for Abilities under Uncertainty*. Technical Report IfI-08-01. Department of Informatics, Clausthal University of Technology.

[37] Wojciech Jamroga and Jürgen Dix. 2006. Model checking abilities under incomplete information is indeed delta2-complete. In *Proceedings of the the 4th European Workshop on Multi-Agent Systems (EUMAS'06)*.

Q8 [38] Wojciech Jamroga and Wiebe van der Hoek. 2004. Agents that know how to play. *Fund. Inf.* 62 (2004), 1–35.

[39] Geert Jonker. 2003. *Feasible Strategies in Alternating-time Temporal*. Master's thesis. University of Utrech, The Netherlands.

[40] Alessio Lomuscio and Franco Raimondi. 2006. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of the Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*. 161–168.

[41] Ron van der Meyden. 1996. Knowledge based programs: On the complexity of perfect recall in finite environments. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'96)*. 31–49.

[42] Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi. 2010. Reasoning about strategies. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*. 133–144. DOI: http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2010.133

[43] Rohit Parikh. 1983. Propositional game logic. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 195–200.

[44] Rohit Parikh and Ramaswamy Ramanujam. 1985. Distributed processes and the logic of knowledge. In *Logics of Programs 1985*. 256–268.

[45] Marc Pauly. 2002. A modal logic for coalitional power in games. *J. Logic Comput.* 12, 1 (2002), 149–166.

[46] Amir Pnueli. 1977. The temporal logic of programs. In *Proceedings of the Symposium on Foundations of Computer Science*. 46–57.

[47] Ramaswamy Ramanujam and Sunil Easaw Simon. 2008. Dynamic logic on games with structured strategies. In *Proceedings of the 1th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*. 49–58.

[48] Olivier Roy. 2009. A dynamic-epistemic hybrid logic for intentions and information changes in strategic games. *Synthese* 171, 2 (2009), 291–320. DOI: http://dx.doi.org/10.1007/s11229-009-9644-3

[49] Henning Schnoor. 2010. *Explicit Strategies and Quantification for ATL with Incomplete Information and Probabilistic Games*. Technical Report 1008. Institut für Informatik, Christian-Albrechts Universität zu Kiel.

[50] Pierre-Yves Schobbens. 2004. Alternating-time logic with imperfect recall. *Electr. Not. Theor. Comput. Sci.* 85, 2 (2004), 82–93. DOI: http://dx.doi.org/10.1016/S1571-0661(05)82604-0

[51] Thomas Schwentick and Volker Weber. 2007. Bounded-variable fragments of hybrid logics. In *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS'07)*. Springer LNCS, Vol. 4393. 561–572. DOI: http://dx.doi.org/10.1007/978-3-540-70918-3_48

[52] A. Prasad Sistla and Edmund M. Clarke. 1985. The complexity of propositional linear temporal logics. *J. ACM* 32, 3 (1985), 733–749.

[53] D. Sutherland. 1986. A model of information. In *Proceedings of the 9th National Computer Security Conference*. 175–183.

Q9 [54] Jan van Eijck. 2013. PDL as a multi-agent strategy logic. In *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*.

[55] Sieuwert van Otterloo and Geert Jonker. 2005. On epistemic temporal strategic logic. *Electronic Notes in Theoretical Computer Science* 126 (2005), 77–92. DOI: http://dx.doi.org/10.1016/j.entcs.2004.11.014

[56] Moshe Y. Vardi. 1996. Implementing knowledge-based programs. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*. 15–30.

[57] J. Todd Wittbold and Dale M. Johnson. 1990. Information flow in nondeterministic systems. In *Proceedings of the IEEE Symposium on Security and Privacy*. 144–161.

Received November 2017; revised March 2018; accepted June 2018

Author Queries

Q1: AU: Please provide CCS 2012 Concepts per author guidelines and provide XML codes as well.
Q2: AU: Please provide Additional Key Words and Phrases section.
Q3: AU: Please provide complete mailing and email addresses for all authors.
Q4: AU: Initial cap "low" and "high" with "low-level attacker" and "high-level secret"?
Q5: AU: Sentence "Another work by van der Hoek, Jamroga...": use a comma after "work" and then after "[28]"?
Q6: AU: Ref. 16: If possible, please update all CoRR and arXiv references in this list.
Q7: AU: Ref. 27: Please provide missing volume or issue number.
Q8: AU: Ref. 38: please provide missing issue or volume number.
Q9: AU: Ref. 54: Please update and complete.