Verification of Multi-Agent Systems with Public Actions against Strategy Logic

Francesco Belardinelli
Department of Computing, Imperial College London, UK and Laboratoire IBISC, UEVE, France

Alessio Lomuscio
Department of Computing, Imperial College London, UK

Aniello Murano
DIETI, Università degli Studi di Napoli, Italy

Sasha Rubin
University of Sydney, Australia

Abstract
Model checking multi-agent systems, in which agents are distributed and thus may have different observations of the world, against strategic behaviours is known to be a complex problem in a number of settings. There are traditionally two ways of ameliorating this complexity: imposing a hierarchy on the observations of the agents, or restricting agent actions so that they are observable by all agents. We study systems of the latter kind, since they are more suitable for modelling rational agents. In particular, we define multi-agent systems in which all actions are public and study the model checking problem of such systems against Strategy Logic with equality, a very rich strategic logic that can express relevant concepts such as Nash equilibria, Pareto optimality, and due to the novel addition of equality, also evolutionary stable strategies. The main result is that the corresponding model checking problem is decidable.

Keywords: Strategy Logic, Multi-agent systems, Imperfect Information, Verification, Formal Methods

1. Introduction

Logics expressing individual and joint strategic abilities offer powerful formalisms for reasoning about the behaviour of rational agents in multi-agent systems (MAS), a subject of growing interest in the area of formal methods for Artificial Intelligence. Coalition
Logic [1] and Alternating-time Temporal Logic (ATL) [2] were among the first and most influential logics that were introduced for this purpose. These logics can be used to express formally what states of affairs coalitions of agents may bring about in a MAS irrespective of what other agents in the system may do. For example, in a scenario where several autonomous robots are competing for resources, a coalition of two robots may be able to enforce that a particular resource can be brought under their control, irrespective of the actions of the other robots in the system.

Over the years ATL has been enriched in a number of directions, including by incorporating epistemic operators to reason about both the knowledge and the strategic power of the agents in the system [3, 4, 5, 6], and by accounting explicitly for the resources agents have available [7]. More recently, formalisms more expressive than ATL have been introduced. In the framework of Strategy Logic (SL) strategies are first-class objects that can be named and associated with agents [8, 9, 10, 11]. This enables the representation of game-theoretic concepts, such as Nash equilibria, that cannot be rendered by formalisms such as ATL, but are of high importance in MAS. Like ATL, SL has also been combined with epistemic concepts [12, 13, 14, 15].

A key focus of attention in these lines of work concerns formal verification, notably the model checking problem [16], of MAS against strategy-based specifications expressed in these languages. Various methods and accompanying implementations have been developed supporting ATL and variations [7, 17, 14, 18, 19, 20, 21]. These range from explicit to symbolic model checkers, as well as SAT-based engines. By using these tools practical scenarios ranging from strategic games [7] to autonomous vehicles [22, 23], have been analysed and debugged.

A crucial consideration in assessing the practical feasibility of verification via model checking is the computational complexity of its decision problem. In this light, an attractive feature of ATL lies in the fact that its model checking problem is PTIME-complete [2]. This is, however, limited to the case of perfect information, i.e., under the assumption that the agents in the system have full visibility of its global state. In MAS this assumption is of limited relevance, as the agents can normally access only a fraction of the information available. Much more important in applications is the case of imperfect information, particularly in the context of perfect recall, where agents in the system have full memory of their local histories. While this is a compelling set-up from a modelling point of view, it is challenging from a verification standpoint, as the corresponding model checking problem is undecidable [24]. It follows that the model checking problem for all extensions of ATL, including Strategy Logic, is also undecidable under the assumptions of perfect recall and imperfect information.

Given this limitation, it is of interest to identify cases for which the model checking problem for strategic reasoning is decidable even under perfect recall and imperfect information. This paper provides one concrete setting, relevant for applications, where we show it to be the case.

More specifically, there appear to be three possible directions to tame undecidability in this context. One option involves restricting the syntax of the specification languages. This option generally results in a loss of expressiveness; however, useful specification patterns might be identified within the fragment [25]. A second possibility might concern modifying the standard semantics for the specification language in question. This might involve amending the standard notion of strategy or to consider minor modifications of the underlying complete information and perfect recall setting [26]. A third line of attack...
consists in identifying semantical subclasses of MAS, still analysed under perfect recall and incomplete information, for which the model checking is decidable. In the following we pursue this latter option.

**Contribution.** In this work we introduce and study a variant of SL under incomplete information (henceforth SLi), and exemplify its applicability in the context of MAS. In particular, we describe a number of formulas of SLi that capture important concepts, such as winning strategies, Nash equilibria, evolutionary stable strategies. We observe that the corresponding model checking problem is undecidable in general, but identify a subclass of MAS for which the same question is decidable. The subclass, that we isolate and investigate, consists of systems of agents that can communicate only via public actions. Examples of such systems include games with fully observable (public) moves, open-cry auction protocols where all bidding is public, and, more generally, systems evolving via broadcasting actions. Clearly, this is a broad class of systems of interest in applications. We analyse the related complexity and show that the model-checking problem for this subclass is in \((k + 2)\text{-EXPTIME}\), where \(k\) is the quantifier-block depth of the formula to be checked. We also provide a lower bound in \((k − 1)\text{-EXPSPACE}\). Thus, this subclass provides a middle ground between the full observability case (which is well-understood, more tractable, but has limited expressiveness and applicability) and the partial observability case (which is undecidable, but extremely expressive).

**Related Work.** The work here presented builds upon and extends the framework of Strategy Logic [9, 11]. In those papers, SL is interpreted on concurrent game structures and, barring the exceptions below, it is typically employed and analysed under complete information and perfect recall. In contrast, here we use a variant of interpreted systems as the underlying semantics and study the verification problem under the assumption of imperfect information and perfect recall.

Variants on the semantics of Strategy Logic have been previously explored. In [27] an alternative setting is studied in which strategies that are not bound to agents are not propagated when temporal operators are evaluated. Under this semantics, the model checking problem becomes undecidable. The notion of dependency between strategies in SL is analysed in [28]. In this work when a strategy \(x\) is quantified in an SL formula, it depends on all other strategies quantified before it. In particular, the value of \(x\) on a given history depends on the value of other strategies on all histories. The same paper introduces, motivates, and studies weaker dependencies (e.g., when a strategy \(x\) is quantified, its value on a history \(h\) depends only on the values of earlier quantified strategies on prefixes of \(h\)). Further, [29] introduces an extension of SL in which strategies can be nondeterministic and there is an unbinding operator that allows agents to revoke their strategies. These extensions allow one to express the notion of sustainable control for an agent, while retaining a decidable model checking problem.

In addition to the above, the verification of MAS against various strategy-based specifications, enriched with epistemic specifications, has been investigated in [7, 17, 30, 18, 19, 14]. However, this has been typically limited to observational or positional semantics, where an agent’s strategy depends only on her current state. In contrast we here analyse the case of perfect recall, which is undecidable. These works also focus mainly on the interplay between strategic and epistemic modalities. While we show that epistemic modalities are also supported in our setting, they are shown to be derivable and do not need to be introduced as first-class citizens. [31] also introduced an epistemic strategy logic and studied the corresponding model checking problem. In this setup,
however, the strategies the agents use are directly encoded in their local states, resulting in a rather different framework. As in [14], the focus is on observational semantics, while here we deal with the arguably more complex case of perfect recall.

Reasoning about strategic abilities in MAS under imperfect information is known to be computationally difficult even for logics less expressive than SL. For example, model checking MAS against ATL specifications goes from PTIME-complete to $\Delta^P_2$-complete under incomplete information and memoryless strategies [32], and it is undecidable under perfect recall [24]. The results of this paper confirm these findings and extend them to Strategy Logic.

Other approaches have been introduced to retain decidability when reasoning about strategies in MAS. A notable direction involves imposing a hierarchy on the information, or the observations, of the agents [33, 34, 35, 36, 37]. This constraints in a well-structured way the information that agents possess. Hierarchies have also been studied in the context of variants of SL, thereby achieving similar results [37]. While we share the motivation and the result of these approaches, these restrictions are considerably different from ours. In particular, we impose no a-priory hierarchy on the information and the observations of the agents. In other words, hierarchies can be represented in the framework presented here, but they are not a constitutive feature.

Differently from the contributions above, we here introduce the use of public actions as a way to retain decidability for the verification problem. There are strong correspondences between our notion of public action and communication by broadcasting, which has previously been studied in the context of MAS in [38, 39, 40]. While the semantics is similar, previous approaches focused on the modelling and axiomatisations of epistemic and temporal-epistemic logics on these structures. Instead, we here study variants of SL and focus on the model checking problem instead.

Also related to the above are recent proposals to approximate the verification problem. For instance, [41] studies an approximation of the model checking problem for ATL under imperfect information, specifically one in which the ATL operators admit fixpoint characterisations. By doing so, while the original undecidable problem cannot be solved, a closely related verification question is offered a solution. Differently from [41], we solve the same verification problem but under the restricted communication assumption. Also, we here work on SL and focus on the model checking problem instead.

Moreover, we note that there are points of contacts between the present work and developments in Dynamic Epistemic Logic (DEL). DEL [42, 43] is a framework whereby an epistemic logic is augmented with dynamic modal operators to model information updates. A noteworthy model update operator in DEL is truthful public announcement. In DEL, as well as in related earlier frameworks [38, 44], this is modelled via an epistemic model update incorporating the postconditions of the action. While the framework here presented and DEL for public announcements address related classes of MAS, the technical approaches are rather different. While in DEL the models are instantaneous representations of the agents’ epistemic alternatives and time is modelled via the update operations, our models, in line with interpreted systems semantics, ATL, and SL models, incorporate primitively the concepts of time in the notion of run and history. Moreover, normally the syntax of DEL does not include operators for strategic abilities.

Related to our contribution is also some of the work in epistemic planning [45, 46], whereby one asks whether there is a sequence of event-models such that the resulting model satisfies a given epistemic formula. Similarly to our findings, the problem is
undecidable in general, but becomes decidable when public actions are assumed [45]. Moreover, [47, 48] study a reduction of epistemic planning with public announcements to classical planning. The encoding allowing this reduction uses an idea similar to what we use here, in that both reductions record the current state for each possible initial state.

An important application of the formalism presented in this paper concerns reasoning about rich solution concepts such as Nash equilibria in which agents have LTL goals, as well as generalisations thereof known as strong rational synthesis. In [49] the strong rational synthesis problem with LTL objectives and aggregation of finitely many objectives is shown to be 2EXPTIME-complete. In [50], Equilibrium Logic is introduced to reason about Nash equilibria in games with LTL and CTL objectives. However, both cases assume that agents have perfect information. In case agents have imperfect information, the existence of Nash equilibria is undecidable for three agents, but decidable for two (cf. [51]). In the case of multiple agents, [37] shows that decidability for a language similar to the one here presented, which can also express the existence of Nash equilibria, can be retained by imposing a hierarchy on the agent observations. In contrast, in Section 3 we show that one can regain decidability, and thus decide the existence of Nash equilibria, by assuming that agents use public actions only, while making no restriction on the agent observations.

On the purely technical side of our contribution, we remark that one of the ideas used in the proof of our decidability result uses ideas similarly to those employed in Littman’s PhD Thesis [52][Lemma 6.1] and [53], in that one can convert a deterministic partially observable Markov decision process (POMDP) into an MDP with exponentially many states. The states of the derived MDP are functions $D_t : S \rightarrow S$ where $D_t(s)$ says that, after a fixed sequence of actions and observations, if $t$ were the initial state then $s$ would be the current state. The main differences with our work are that i) we consider a set of initial states (rather than a probability distribution over the initial states), and ii) we model check a very expressive logic, rather than simply solving the synthesis problem.

**Previous Work.** Preliminary versions of this work were published in non-archival conference papers by the same authors [54, 55, 56]. There are a number of notable differences that are introduced here to make the paper uniform, mature, and self-contained. First, differently from [54, 55], this paper uses interpreted systems, rather than concurrent game structures, as the underlying semantical model. This enables us, as we did in [56] (and this is the only overlap with that paper), to give a more intuitive and precise definition of what it means for an action of an agent to be public. Indeed, in [54, 55] we only referred to joint actions being public, not individual ones. Second, our logic no longer includes explicit epistemic operators, but does allow equality between strategies. Third, we include extensive examples of the expressiveness of the logic. Fourth, we provide a a different, conceptually simpler, and fully detailed proof of the main decidability result, including a complexity analysis. Indeed, instead of giving a reduction to monadic second-order logic [55], or using automata-theoretic techniques [54], we here give a reduction to an intermediate logic (quantified CTL*) that has served as a useful and natural bridge between strategic logics and monadic second-order logic.

**Outline.** The rest of the paper is organised as follows.

- In Section 2 we define the syntax and semantics of $SLi$, as well as provide a number of formulas of $SLi$ and discuss their importance and relevance to express the strategic abilities of agents in MAS.
− In Section 3 we introduce the subclass of systems in which agents operate only through public actions, we prove that the corresponding model checking problem is decidable, and provide upper and lower bounds on its complexity.

− In Section 4 we summarise the main findings of the paper and point to future directions of research.

2. Strategy Logic under Imperfect Information

In this section we introduce Strategy Logic under imperfect information (SLi), a logic for strategic reasoning in multi-agent systems. The logic is inspired by Strategy Logic (SL) [10], in which strategies are treated syntactically in the language. This is accomplished by having quantification on first-order variables ranging over strategies. In SL the strategy quantifier \( \exists x \) is read as “there exists a strategy \( x \)”, and the binding operator \( \text{bind}(i, x) \) is read as “agent \( i \) uses strategy \( x \)”. Moreover, SL includes linear-temporal operators \( \text{X} \) and \( \text{U} \) [57] for reasoning about the temporal evolution of the system in which the agents are bound to particular strategies. The logic we introduce, SLi, inherits these features, and allows us to express game-theoretic concepts such as existence of winning strategies, Nash equilibria, etc. [58].

The main difference between SL and SLi is that the semantics of SLi permits one to reason about agents with imperfect information (other similar extensions of SL such as [37, 14] are discussed in the related-work section). Thus, if an agent is associated to a strategy \( x \), then strategy \( x \) will prescribe actions that are enabled by the protocol of that agent, and that only depend on the local state of the agent; cf. [10, 14]. SLi is equipped with two types of strategy quantifiers (\( \exists_o x \) and \( \exists_s x \)); these are inspired by the distinction between the objective and subjective semantics of alternating-time temporal logic (ATL) [59]. Intuitively, these corresponds to whether or not the quantified strategy is known to succeed by the agents that use them. Finally, SLi has the ability to express whether two strategies are equal; this is inspired by SL with Graded Modalities [60] and first-order logic with equality.

2.1. Syntax of SLi

In what follows we fix \( AP \) to be a finite, non-empty set of atoms, and \( Ag \) to be a finite, non-empty set of agents. Further, let \( Var \) be a finite, non-empty set of strategy variables denoted by \( x, y, \ldots \), and \( x_i, y_i, \ldots \).

**Definition 1 (SLi).** The following grammar defines the SLi formulas \( \varphi \):

\[
\varphi ::= p \mid x = y \mid \neg \varphi \mid \varphi \lor \varphi \mid \text{X} \varphi \mid \varphi \text{U} \varphi \mid \exists_o x \varphi \mid \exists_s x \varphi \mid \text{bind}(i, x) \varphi
\]

where \( p \in AP, \ x, y \in Var, \ i \in Ag \).

Without loss of generality, we assume that every variable \( x \) is quantified at most once in \( \varphi \) (this can be done, without changing the size of the formula, by renaming variables). We use the standard abbreviations, e.g., \( \varphi_1 \land \varphi_2 \) abbreviates \( \neg (\neg \varphi_1 \lor \neg \varphi_2) \), \text{true} abbreviates \( p \lor \neg p \), \text{F} \( \varphi \) and \text{G} \( \varphi \) are shorthands for \text{true} \text{U} \varphi \) and \( \neg \text{F} \neg \varphi \) respectively, and \( \forall x \varphi \) abbreviates \( \exists x \neg \varphi \).

**Discussion.** The formal semantics of SLi is presented in Section 2.2. Here, we give intuitions about how to interpret formulas of SLi.
Inspired by the distinction between the objective and subjective semantics of strategy modalities in alternating-time temporal logic under imperfect information [59], we use $\exists_o$ to refer to the objective interpretation and $\exists_s$ to refer to the subjective interpretation.

Quantified strategies are intended as being coherent and uniform, i.e., if an agent uses a strategy then the actions it prescribes should be available to that agent (coherency), and these actions should only depend on the local state of the agent (uniformity). Thus, a formula $\exists_o x \phi$ (or $\exists_s \phi$) is read as “there exists a strategy $x$, that is coherent and uniform for all agents that use it (in the subformula $\phi$) such that $\phi$ holds”. Note that normally, both in SL and in ATL the formula is simply read as “there exists a strategy $x$ such that $\phi$”.

The formula $x = y$ tests whether the strategies denoted by $x$ and $y$ are equal. This is inspired by the distinction between first-order logic with and without equality. Moreover, it allows us to express complex properties such as uniqueness of solution concepts (see Example 5), and the existence of an evolutionary stable strategy (see Example 6). Previous versions of SL (e.g., [10]) do not include equality on strategies, although some versions allow strategy counting [60].

To define the semantics we need some further notation defined below.

**Free Variables and Agents.** We introduce the set $\text{free}(\phi)$ to denote the set of free variables and agents appearing in a formula $\phi$ (cf. [10]). Intuitively, a variable $x$ is free in $\phi$ if one needs to associate $x$ with a strategy in order to evaluate $\phi$, and an agent $a$ is free in $\phi$ if one needs to bind a strategy to $a$ in order to evaluate $\phi$.

**Definition 2 (Free variables and agents).** The set $\text{free}(\phi) \subseteq \text{Ag} \cup \text{Var}$ representing free agents and variables is defined inductively as follows:

\[
\begin{align*}
\text{free}(p) &= \emptyset \\
\text{free}(x = y) &= \{x, y\} \\
\text{free}(\neg \phi) &= \text{free}(\phi) \\
\text{free}(X \phi) &= \text{Ag} \cup \text{free}(\phi) \\
\text{free}(\phi_1 \cup \phi_2) &= \text{Ag} \cup (\text{free}(\phi_1) \cup \text{free}(\phi_2)) \\
\text{free}(\phi_1 \lor \phi_2) &= \text{free}(\phi_1) \cup \text{free}(\phi_2) \\
\text{free}(\exists_o x \phi) &= \text{free}(\exists_s x \phi) = \text{free}(\phi) \setminus \{x\} \\
\text{free}(\text{bind}(i, x) \phi) &= \begin{cases} 
(\text{free}(\phi) \setminus \{i\}) \cup \{x\} & \text{if } i \in \text{free}(\phi) \\
\text{free}(\phi) & \text{otherwise.}
\end{cases}
\end{align*}
\]

A formula $\phi$ without free agents (resp., variables), i.e., with $\text{free}(\phi) \cap \text{Ag} = \emptyset$ (resp., $\text{free}(\phi) \cap \text{Var} = \emptyset$), is agent-closed (resp., variable-closed). If $\phi$ is both agent- and variable-closed, it is called a sentence.

**Agents using strategies in a subformula.** We introduce the set $\text{use}(x, \phi)$ to denote the agents using strategy $x$ in evaluating formula $\phi$. Formally, let $\text{use}(x, \phi)$ consist of all agents $i \in \text{Ag}$ such that $\phi$ has a subformula of the form $\text{bind}(i, x) \varphi'$ such that $i \in \text{free}(\varphi')$. This set will be used to provide an imperfect information semantics to SLi.
2.2. Interpreted Systems

We introduce a novel variant in which we distinguish specific actions that we call public and transition function [38]. In this section we recall this semantics, whereas in Section 3 we introduce a model for multi-agent systems where each agent is defined by its local states, actions, and transition function [38]. In this section we recall this semantics, whereas in Section 3 we introduce a new variant in which we distinguish specific actions that we call public.

**Notation.** We write \([n]\) for the set \(\{i \in \mathbb{N} : 1 \leq i \leq n\}\). The length of a finite sequence \(u \in X^*\) is denoted by \(|u| \in \mathbb{N}\). For \(i \geq 1\), we write \(u_i\) for the \(i\)-th element of \(u\), and \(u_{<i}\) for the prefix of \(u\) of length \(i\). Then, we denote its first element \(u_1\) by \(\text{first}(u)\), and its last element \(u_{|u|}\) by \(\text{last}(u)\). To ease notation, we sometimes write \(u(i)\) instead of \(u_i\). The empty sequence is denoted by \(\epsilon\). The length of an infinite sequence is the cardinal \(\omega\). For a vector \(v \in \prod X_j\) we denote the \(i\)-th co-ordinate of \(v\) by \(v_i\). The powerset of \(X\) is denoted \(\mathcal{P}(X)\). We use the following convention: let \(f, f' : X \to Y\) be partial functions and \(\sim\) a binary relation on \(Y\); then whenever we write \(f(x) \sim f'(x')\) we mean, in particular, that both \(f(x)\) and \(f'(x')\) are defined.

**Definition 3 (Interpreted Systems).** An interpreted system (IS) is a tuple

\[ S = (Ag, \{L_i, Act_i, P_i, \tau_i\}_{i \in Ag}, S_0, AP, \lambda) \]

where

1. \(Ag = [n]\) for some \(n \in \mathbb{N}\), is a finite non-empty set of agents.

2. For each agent \(i \in Ag:\)
   (a) \(L_i\) is a finite non-empty set of local states.
   (b) \(Act_i\) is a finite non-empty set of local actions.
   (c) \(P_i : L_i \to \mathcal{P}(Act_i) \setminus \{\emptyset\}\) is the local protocol.
   (d) \(\tau_i : L_i \times \prod_{j \in Ag} Act_j \to L_i\) is a partial function, called the local transition function, such that for every \(l \in L_i\), \(a \in \prod_{j \in Ag} Act_j\), \(\tau_i(l, a)\) is defined iff \(a_i \in P_i(l)\).

3. \(S_0 \subseteq \prod_{i \in Ag} L_i\) is the set of initial global states.

4. \(AP\) is the finite set of atomic propositions (also called atoms).

5. \(\lambda : AP \to \mathcal{P}(\prod_{i \in Ag} L_i)\) is a labelling function.

Note that we do not assume that sets \(Act_i\)s of local actions are disjoint. Intuitively, an interpreted system describes the synchronous evolution of a group \(Ag\) of agents: at any point in time, each agent \(i\) is in some local state \(l \in L_i\), which encodes the (possibly partial) information she has about the state of the system. The local protocol \(P_i\) specifies which actions from \(Act_i\) agent \(i\) can execute from each local state. The execution of a joint action \(a \in \prod_{i \in Ag} Act_j\) gives rise to the transition from the present state \(l\) to the successor state \(\tau_i(l, a)\). The actions in \(P_i(l)\) are said to be available to agent \(i\) in local state \(l \in L_i\). Thus, the local transition function \(\tau_i(l, a)\) is defined iff action \(a_i\) is available to agent \(i\) in local state \(l\).

We now recall some standard terminology about interpreted systems (IS) that allows us to reason about temporal, epistemic and strategic properties in an IS.
Global States, Joint Action, and Transitions. We introduce the following notions:

- the set $S \triangleq \prod_{i \in Ag} L_i$ is called the set of global states;
- the set $Act \triangleq \prod_{i \in Ag} Act_i$ is called the set of joint actions;
- the partial function $\tau : S \times Act \to S$ is called the global transition function; it is defined so that $\tau(s, a) = s'$ iff for every $i \in Ag$, $\tau_i(s_i, a) = s_i'$;
- the set of all actions $\cup_{a \in Ag} Act_a$ is denoted $act$.

Notice that transitions are synchronous executions of individual actions, one for each agent in the system. Atomic facts about the system at every point in time are given by the labelling function $\lambda$.

Runs and Histories. A run (resp. history) is an infinite (resp. finite non-empty) sequence $r = r(1)r(2)\cdots$ of global states starting in an initial state and respecting the global transition function, i.e., $r(1) \in S_0$ and for every $t < |r|$ there exists a joint action $a \in Act$ such that $\tau(r(t), a) = r(t + 1)$. The set of all histories is denoted by Hist. Notice that $S_0 \subseteq Hist$. For a run (or history) $r$, agent $i \in Ag$, and index $t < |r|$, let $r(t)_i$ be the local state of agent $i$ in the global state $r(t)$ (we use this notation as it is easier to read than $r(t)(i)$).

Perfect recall of observations. We assume that agents have perfect recall. Intuitively, this means that they remember the full history of their observations. Formally, we define the indistinguishability relation of agent $i$ as the equivalence relation $\sim_i$ over global states $S$ defined as follows: $s \sim_i s'$ iff $s_i = s'_i$, that is, two global states are indistinguishable for agent $i$ iff agent $i$'s local state is the same in both [38]. In order to capture agents with perfect recall of their observations, this relation is lifted to histories in a synchronous, pointwise fashion: $h \sim h'$ iff (1) $|h| = |h'|$ and (2) $h(t) \sim h'(t)$ for $1 \leq t \leq |h|$.

Strategies. We define a strategy to be a function $\sigma : Hist \to \cup_{j \in Ag} Act_j$ from histories to actions. Then, let Str denote the set of all strategies. A strategy $\sigma$ is coherent for agent $i$ if action $\sigma(h)$ is available to agent $i$ in local state $last(h)_i$, that is, $\sigma(h) \in P_i(last(h)_i)$; it is uniform for agent $i$ if $h \sim h'$ implies $\sigma(h) = \sigma(h')$, that is, in indistinguishable states agent $i$ is required to execute the same action [59, 61].

Assignments. A valuation is a function $\nu : Var \to Str$ that maps variables to strategies. A valuation $\nu$ is $\varphi$-compatible if for every variable $x \in Var$, the strategy $\nu(x)$ is coherent and uniform for every agent in $use(x, \varphi)$. A binding is a function $\beta : Ag \to Var$ that maps agents to variables. Note that the composition function $\nu(\beta(\cdot))$ maps agents to strategies (in the game-theory literature, such a function is called a strategy profile). An assignment $\chi$ is a pair $(\nu, \beta)$ such that for all $i \in Ag$, the strategy $\nu(\beta(i))$ is coherent and uniform for $i$. An assignment $(\nu, \beta)$ and a history $h$ determine a unique infinite run $\pi(h, \nu, \beta)$ in which agents play according to the assigned strategies, i.e., agent $i$ plays starting from $h$ according to $\nu(\beta(i))$. Formally, $\pi(h, \nu, \beta)$ is defined as the run $\pi$ such that (1) $\pi_{\leq |h|} = h$; and (2) for $t > |h|$, $\pi_t = \tau(\pi_{t-1}, a)$ where $a_i = \nu(\beta(i))(\pi_{\leq t-1})$ for every $i \in Ag$. 

9
Remark 1. We provide the derived semantics of the universal strategy quantifiers \( \forall_s \) and \( \forall_o \):

\[
(S, h, \nu, \beta) \models s \varphi \quad \text{if for every } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } \text{use}(x, \varphi) \text{, we have that } (S, h, \nu|x \mapsto \sigma, \beta) \models \varphi
\]

\[
(S, h, \nu, \beta) \models o \varphi \quad \text{if for every } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } \text{use}(x, \varphi) \text{, we have that } (S, h, \nu|x \mapsto \sigma, \beta) \models \varphi
\]

We now give the semantics of the satisfaction relation for the logic.

**Definition 4 (Semantics).** For a given IS \( S \) and an SLi formula \( \varphi \), we define the satisfaction relation \( (S, h, \nu, \beta) \models \varphi \) inductively on the structure of \( \varphi \), where \( h \) is a history, \( \nu \) is a \( \varphi \)-compatible valuation, and \( \beta \) is a binding such that \( (\nu, \beta) \) is an assignment:

\[
(S, h, \nu, \beta) \models p \quad \text{if last}(h) \in \lambda(p)
\]

\[
(S, h, \nu, \beta) \models x = y \quad \text{if for every history } h' \text{ extending } h, \nu(x)(h') = \nu(y)(h')
\]

\[
(S, h, \nu, \beta) \models \neg \varphi \quad \text{if } (S, h, \nu, \beta) \not\models \varphi
\]

\[
(S, h, \nu, \beta) \models \varphi \lor \varphi' \quad \text{if } (S, h, \nu, \beta) \models \varphi \text{ or } (S, h, \nu, \beta) \models \varphi'
\]

\[
(S, h, \nu, \beta) \models \Box \varphi \quad \text{if } (S, \pi \leq |h+1|(h, \nu, \beta)) \models \varphi
\]

\[
(S, h, \nu, \beta) \models \exists x \varphi \quad \text{if for some } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } \text{use}(x, \varphi) \text{, we have that } (S, h, \nu|x \mapsto \sigma, \beta) \models \varphi
\]

\[
(S, h, \nu, \beta) \models \exists o x \varphi \quad \text{if for every } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } \text{use}(x, \varphi) \text{, we have that } (S, h', \nu|x \mapsto \sigma, \beta) \models \varphi
\]

\[
(S, h, \nu, \beta) \models \text{bind}(i, x) \varphi \quad \text{if } (S, h, \nu, \beta[i \mapsto x]) \models \varphi
\]

Note that the satisfaction relation is well defined in the sense that the valuation-binding pairs introduced at every step of the inductive definition are indeed assignments and that the valuations are compatible. We prove this in Appendix A.

It is routine to show (by structural induction) that if \( \varphi \) is a sentence, i.e., \( \text{free}(\varphi) = \emptyset \), then \( (S, h, \nu, \beta) \models \varphi \) does not depend on the assignment \( (\nu, \beta) \). Thus, for a sentence \( \varphi \) we write \( (S, h) \models \varphi \) to mean that \( (S, h, \nu, \beta) \models \varphi \) for some (equivalently every) assignment \( (\nu, \beta) \). Further, we say that \( \varphi \) is true in \( S \), and write \( S \models \varphi \), iff for every initial state \( s \in S_0 \), \( (S, s) \models \varphi \).

**Remark 1.** We provide the derived semantics of the universal strategy quantifiers \( \forall_s \) and \( \forall_o \):

\[
(S, h, \nu, \beta) \models s \varphi \quad \text{if for every } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } \text{use}(x, \varphi) \text{, we have that } (S, h, \nu|x \mapsto \sigma, \beta) \models \varphi
\]

\[
(S, h, \nu, \beta) \models o \varphi \quad \text{if for every } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } \text{use}(x, \varphi) \text{, we have that } (S, h', \nu|x \mapsto \sigma, \beta) \models \varphi
\]

for some \( h' \sim_i h \) where \( i \in \text{use}(x, \varphi) \).

Note that in the subjective semantics (i.e., \( \exists_s \) and \( \forall_s \)), we only consider reachable epistemic alternatives \( h' \), as histories are defined to start in initial states and being consistent with the transition function \( \tau \).
We also observe that the formal meaning of the $\exists_s$ quantifier seems disaligned from the usual intuition about universal quantification (since it includes an existential quantification over histories). However, this reading is consistent with the subjective interpretation of operator $[[A]]$ in ATL. Specifically, no matter what strategy we consider, $\varphi$ is epistemically consistent.

**Remark 2.** The subjective existential quantifier $\exists_s$ allows us to introduce an epistemic operator $K_i$ (for certain formulas) that represents the individual knowledge of agent $i$ as “truth in indistinguishable histories” [38]. Indeed, define $K_i\varphi \triangleq \exists_z \text{bind}(i, z) \varphi$, where $z$ is a fresh variable not appearing in $\varphi$ and $i \notin \text{free} (\varphi)$. Since, $i$ does not appear free in $\varphi$, the truth of $K_i\varphi$ does not depend on the particular strategy assigned to $z$, and therefore we have that:

$$(S, h, \nu, \beta) \models K_i\varphi \quad \text{if for every history } h' \sim_i h \text{ we have that } (S, h', \nu, \beta) \models \varphi$$

We remark that the operator $K_i$ defines a notion of knowledge based on truth in indistinguishable histories, a mainstream notion in knowledge representation and multi-agent systems [38]. On the other hand, the epistemic and strategic dimensions of multi-agent systems can be combined in many different ways (see [59, 15] for some examples). Such an analysis is beyond the scope of the current contribution.

**Remark 3.** We discuss the definition of equality $=$. Informally, we consider two strategies to be equal if they agree on all histories. However, since formulas of SLi cannot talk about the past, we may restrict this definition to histories extending the current one. That is, $(S, h) \models \sigma = \sigma'$ iff $\sigma$ and $\sigma'$ coincide on all histories extending $h$ (which includes $h$ itself). This ensures that also $= \models$ does not talk about the past, which is technically helpful (in the proof of Proposition 2).

Note that the behaviour of $\sigma$ and $\sigma'$ cannot be distinguished in SLi without subjective quantifiers $\forall_s$ and $\exists_s$. Indeed, well-known principles characterising equality, such as the substitution of identicals: $\forall_s x \forall_s y (x = y \to (\varphi \leftrightarrow \varphi[x/y]))$, are valid whenever neither $\forall_s$ nor $\exists_s$ appear in $\varphi$.

Furthermore, we might want to consider a notion of equality that also accounts for extensions of epistemic alternatives of the current history $h$. It turns out that such “subjective” equality $=_{s}$ can be defined by using $=$ and the epistemic operator $K_i$ introduced in Remark 2. More formally, define $x =_s y ::= \bigwedge_{i \in A_g} K_i (x = y)$. Then, the meaning of $=_s$ is as follows:

$$(S, h, \nu, \beta) \models x =_s y \quad \text{if for every agent } i, \text{ every history } h' \sim_i h, \text{ and every history } h'' \text{ extending } h', \text{ we have that } \nu(x)(h') = \nu(y)(h'')$$

Since $=_{s}$ is definable in terms of $=$ and $K_i$, we take the latter as primitive. Note however, that still formula $\forall_s x \forall_s y (x =_s y \to (\varphi \leftrightarrow \varphi[x/y]))$ is not valid unrestrictedly.

**Remark 4 (Syntactic Fragments of SLi).** It is well-known that, in the perfect information setting, SL subsumes the alternating-time temporal logic ATL* [2], and therefore also ATL and the temporal logics LTL, CTL, CTL*.

Similarly, in the imperfect information setting of this work, ATL* can be seen as a syntactic fragment of SLi. To show this, we present the syntax of ATL*, where we
explicitly distinguish between strategy operators \( \langle\!\langle A \rangle\!\rangle_s \) (resp. \( \langle\!\langle A \rangle\!\rangle_o \)) interpreted according to the objective (subjective, resp.) semantics for ATL*:

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid \langle\!\langle A \rangle\!\rangle_o \psi \mid \langle\!\langle A \rangle\!\rangle_s \psi
\]

\[
\psi ::= \varphi \mid \neg \psi \mid \psi \lor \psi \mid X \psi \mid \psi U \psi
\]

where \( p \in AP \), and \( A \subseteq Ag \).

Here we do not provide the semantics of ATL* but refer to [59] for full details.

We can define a translation \( t \) from ATL* to SLi that is the identity on atoms (i.e., \( t(p) = p \)) and that commutes with the Boolean and temporal operators (e.g., \( t(\neg \varphi) = \neg t(\varphi) \) and \( t(X \psi) = X t(\psi) \)), and whose translation of strategy formulas is given as follows, for \( Ag = [n] \) and \( A = [m] \):

\[
\begin{align*}
t(\langle\!\langle A \rangle\!\rangle_o \psi) & = (\exists_o x_1)_{1 \leq i \leq m} (\forall_o x_1)_{m < i \leq n} (\text{bind}(x_i, a_i))_{1 \leq i \leq n} t(\psi) \\
t(\langle\!\langle A \rangle\!\rangle_s \psi) & = (\exists_o x_1)_{1 \leq i \leq m} \bigwedge_{i < m} K_i(\forall_o x_1)_{m < i \leq n} (\text{bind}(x_i, a_i))_{1 \leq i \leq n} t(\psi)
\end{align*}
\]

Specifically, the translation of operator \( \langle\!\langle A \rangle\!\rangle_o \) closely corresponds to its informal reading: there exist (uniform) strategies for the agents in coalition \( A \) such that, no matter what the agents in \( Ag \setminus A \) do, it is the case that \( \psi \) holds. As regards \( \langle\!\langle A \rangle\!\rangle_s \), its translation states that “there exists (uniform) strategies for the agents in coalition \( A \) such that in all histories indistinguishable for some agent in \( A \), no matter what the agents in \( Ag \setminus A \) do it is the case that \( \psi \) holds”.

In particular, the truth-preserving implication from \( \langle\!\langle A \rangle\!\rangle_o \psi \) to \( t(\langle\!\langle A \rangle\!\rangle_o \psi) \) holds independently from the assumptions on knowledge and memory, as the choice of any strategy for the adversary coalition \( Ag \setminus A \) generates some path in the iCSG. As for the converse implication, we claim that if \( \langle\!\langle A \rangle\!\rangle_o \psi \) is false, then \( t(\langle\!\langle A \rangle\!\rangle_o \psi) \) is false as well. Indeed, suppose that for every strategy available to coalition \( A \), there exists some path \( \lambda \) such that \( \psi \) is false on \( \lambda \). Given such a path \( \lambda \), we can define a joint strategy for the adversary coalition \( Ag \setminus A \) that basically returns the actions played by \( Ag \setminus A \) along \( \lambda \). The fact that agents have perfect recall allows them to play possibly different actions whenever they end up in an indistinguishable state along the path, and therefore the strategy for \( Ag \setminus A \) is well-defined. Moreover, the strategy can be assumed to be uniform w.l.o.g. by simply associating the same action in indistinguishable histories. As a result, the translation \( t(\langle\!\langle A \rangle\!\rangle_o \psi) \) is false as well.

Further, note that a naive translation of \( \langle\!\langle A \rangle\!\rangle_s \) that makes use of a suite \( (\exists_s x_i)_{1 \leq i \leq m} \) of subjective quantifiers will not achieve the same effect, as for each quantifier \( \exists_s x_i \) a possible different set of histories (i.e., those indistinguishable for agent \( i \)) may be selected.

Following the intuitions above, by suitably adapting the semantics in [59] to interpreted systems, it can be shown that a formula \( \varphi \) in ATL* is true in an IS S iff its translation \( t(\varphi) \) is. Hence, our logic SLi is a (conservative) extension of ATL* under imperfect information, both in its objective and subjective interpretation.

2.3. MAS Specifications in SLi

In this section we illustrate the use of SLi for the specification of strategic interplay in MAS. As we show below, SLi is a very expressive specification language to reason about MAS under incomplete information.
Example 1. [Winning strategies] We begin by observing that since ATL\(^*\) formulas can be expressed in SLi (Remark 4), SLi can express express specifications often used in voting (“a coercer can ensure that the voter will eventually either have voted for a given candidate or be punished” [41]), bridge endplay (“a given player can ensure that her team takes more than half of the remaining tricks” [41]), scheduler systems (mutual exclusion and lack of starvation [14]), and anonymity protocols (such as dining cryptographers [14]). The corresponding SLi specifications used in these context are variations of the property expressing that a player in a game has a winning strategy.

Suppose \(S\) represents a card game between multiple players in which the atom \(points^p_j\) represents that player \(j\) has scored \(p\) points (see Example 7 for more details). The SLi formula
\[
\text{winning}_1 \triangleq \left( \bigwedge_p points^p_1 \rightarrow \bigvee_{q:q<p} points^q_2 \right)
\]
expresses that player 1 has scored more points than player 2. This can be generalised to player 1 having scored more points than any of the other players. Let \(end\) be an atom denoting that the game has ended, and define the SLi formula
\[
\psi \triangleq \text{bind}(1,x)\text{bind}(2,y) F(end \land \text{winning}_1)
\]
that expresses that if player 1 uses strategy \(x\) and player 2 uses strategy \(y\) then eventually the game ends with player 1 having more points than player 2.

Consider the formula schema \(\varphi \triangleq \exists x \forall y \psi\) that expresses, intuitively, that player 1 has a strategy that dominates all of player 2’s strategies. We will consider all 4 variations of this schema in which the quantifiers are subjective or objective. Since these are sentences, we consider, for a given a history \(h\), whether \((S,h) \models \varphi\). Note that in all cases the strategy quantified by \(x\) must be coherent and uniform for agent 1 since only agent 1 uses strategy \(x\) (formally, \(\text{use}(x, \forall y \text{bind}(1,x)\text{bind}(2,y) F(end \land \text{winning}_1)) = \{1\}\); similarly, throughout we will assume that the strategy quantified by \(y\) must be coherent and uniform for agent 2. For simplicity, we will assume that \(h\) is a history of length 1, i.e., player \(i\) (for \(i = 1, 2\)) has been dealt a set \(H_i\) of cards that the other player cannot see, and the game is about to commence.

1. The sentence \(\exists o x \forall o y \psi\) represents that there is a strategy \(\sigma_1\) for player 1, such that for every strategy \(\sigma_2\) for player 2, if each player uses their strategy starting from \(h\), player 1 will win. In words, player 1, with hand \(H_1\), can defeat player 2 if his hand is \(H_2\).

2. The sentence \(\exists x \forall y \forall o \psi\) means that there is a strategy \(\sigma_1\) for player 1, such that for every \(h' \sim_1 h\), and every strategy \(\sigma_2\) for player 2, if each player uses their strategy starting from \(h'\), player 1 will win. In words, player 1, with hand \(H_1\), can defeat player 2 no matter what his hand is.

3. The sentence \(\exists x \forall y \forall o \psi\) means that there exists a strategy \(\sigma_1\) for player 1 such that for all strategies \(\sigma_2\) for player 2 there exists \(h'' \sim_2 h\), such that if each player uses their strategy, then starting at \(h''\) player 1 will win. In words, player 1, with hand \(H_1\), can ensure that player 2 will consider it possible that player 2 (not knowing player 1’s hand) will be defeated.
4. The sentence $\exists x \forall y \psi$ means that there exists a strategy $\sigma_1$ for player 1, such that for every $h' \sim_1 h$, and every strategy $\sigma_2$ for player 2, there exists $h'' \sim_2 h'$, such that if each player uses their strategy, then starting at $h''$ player 1 will win. In words, player 1 has a strategy that she knows player 2 will think it may defeat him (player 2).

Observe that $\exists x \psi$ logically implies $\exists o x \psi$, whereas the converse does not hold. Thus, e.g., $\exists x \forall y \psi$ implies $\exists o x \forall y \psi$. Moreover, the formulas $\exists x \forall y \psi$ and $\exists o x \forall y \psi$ have the same interpretation as the formulas $\langle\{1\}\rangle o \psi$ and $\langle\{1\}\rangle o \psi$ in $\text{ATL}^*$ respectively. However there are no simple translations for the formulas $\exists o x \psi$ and $\exists s x \psi$ into $\text{ATL}^*$, since the latter cannot express both the subjective and objective interpretation of quantifiers.

In what follows, let $\bar{x}$ denote a tuple $(x_1, x_2, \ldots, x_n)$ of strategy variables, and let $\text{bind}(i, x_i)_{i \in [n]}$ stand for the binding prefix $\text{bind}(1, x_1) \text{bind}(2, x_2) \cdots \text{bind}(n, x_n)$.

Example 2. [Dependencies of coalitions] By alternating quantifiers, $\text{SLi}$ can express various dependencies of coalitions in games. For instance, the formula $\exists x_1 \exists x_3 \forall x_2 \text{bind}(i, x_i)_{i \in [3]} \psi$ represents that players 1 and 2 can collude to ensure that $\psi$ holds no matter what player 2 does. Compare this with the formula $\exists x_1 \forall x_2 \exists x_3 \text{bind}(i, x_i)_{i \in [3]} \psi$ which is similar except that player 3’s strategy may depend on player 2’s strategy. Note that (1) can be expressed by the $\text{ATL}^*$ formula $\langle\{1, 2\}\rangle o (\psi)$; in contrast, formula (2) cannot be expressed in $\text{ATL}^*$.

Example 3. [Game-theoretic solution concepts] $\text{SLi}$ can express classic notions of strategic behaviour in multiplayer games, e.g., best-response and Nash equilibrium. Consider a game where the objective for player $i$ is encoded by the formula $\psi_i$ (objectives may be arbitrary $\text{SLi}$ formulas; typically, they are just LTL formulas). The $\text{SLi}$ formula $BR_i(\overline{x}) \triangleq (\exists o y \text{bind}(j, x_j)_{j \neq i} \text{bind}(i, y) \psi_i) \rightarrow \text{bind}(j, x_j)_{j \neq i} \text{bind}(i, x_i) \psi_i$ expresses that $x_i$ is a best-response to $(x_j)_{j \neq i}$, that is, if agent $i$ can achieve goal $\psi_i$ by playing the strategy $y$, then she already can by playing strategy $x_i$. Building on this, the $\text{SLi}$ formula $NE(x_1, \ldots, x_n) \triangleq \bigwedge_{i \in [n]} BR_i(\overline{x})$ expresses that each strategy $x_i$ is a best-response to the strategies of the other players. Nash equilibria (NE), as expressed by the formula above, describe optimal play in two-player zero-sum games of imperfect information [58]. $\text{SLi}$ can express properties that
build on NE, such as correctness of fair division protocols \[62\]. Related notions such as (strong) rational synthesis \[49\] and the simple one-alternation strategy formulas for two-player zero-sum games \[8\] can also be expressed in SL and thus in SLi. Also, NE are used as a basis for other solution concepts. For instance, subgame-perfect equilibria of certain infinite-duration games can be expressed in SLi \[60\]. Subgame-perfect equilibria are arguably more suited to graph-games because they eliminate some implausible NE \[63\].

Finally, SLi can express solution concepts such as \(k\)-resilience and \(t\) immunity that are also used in rational distributed computing \[64, 65\].

We now discuss in more detail a subjective interpretation of the formulas expressing best-response and NE. Recall that \(K_i\varphi\) is expressible as long as \(i\) is not free in \(\varphi\). So, the SLi formula 

\[
\text{bind}(j, x_j)_{j \neq i} \text{bind}(i, y)\psi_i \rightarrow K_i(\text{bind}(j, x_j)_{j \neq i} \text{bind}(i, x_i)\psi_i)
\]

expresses that for every strategy \(y\) to achieve goal \(\psi_i\), strategy \(x_i\) is known by agent \(i\) to be at least as good as a response to \((x_j)_{j \neq i}\) as \(y\).

Then, the formula, \(KBR_i(\pi)\) defined by

\[
(\exists s \text{ bind}(j, x_j)_{j \neq i} \text{bind}(i, y)\psi_i) \rightarrow K_i(\text{bind}(j, x_j)_{j \neq i} \text{bind}(i, x_i)\psi_i)
\]

expresses that for every strategy profile \((x_j)_{j \neq i}\), if some strategy \(y\) for agent \(i\) achieves goal \(\psi_i\), then strategy \(x_i\) is known to be a best-response to \((x_j)_{j \neq i}\).

Finally, the SLi formula \(KNE(x_1, \ldots, x_n)\), defined by \(\wedge_{i \in [n]} KBR_i(\pi)\) expresses an epistemic variant of NE, according to which the strategy each agent currently plays is not just the best response, but it is known to be so by each agent.

To illustrate these formulas, consider the turn-based game of imperfect information in Fig. 1 in which agent 1 plays first (atoms true in a state are drawn to the left of that state). Observe that such a game can be represented as an IS, in which both agents 1 and 2 are uncertain about the initial state. Agent 1 has goal \(XXp\), while agent 2 has goal \(XXq\). Notice that a uniform strategy for agent 1 consists of a single move, either \(L\) (left) or \(R\) (right); whereas agent 2’s strategies must be uniform on \(\{s_L, s'_L\}\) and \(\{s_R, s'_R\}\), even though she might choose different actions for the two knowledge sets. Now consider the strategy profile where agent 1 plays \(L\) and agent 2 plays \(L\) in all states. We can check
that playing $L$ is a best response for agent 1, that is, the formula $KBR_1(\pi)$ is true in $s_0$ for the relevant strategy profile $\pi$. Indeed, even though playing $L$ does not guarantee that agent 1 achieves his goal, the antecedent of $KBR_1$ is false as well: since $p$ is false in $s'_{RL}$, there is no strategy that guarantees that agent 1 also achieves his goal from the indistinguishable state $s'_0$, and therefore agent 1 does not know the alternative to be a best response. However, is always playing $L$ a known best response for agent 2? Although playing $R$ would guarantee that agent 2 knows he achieves his goal, this knowledge can be already obtained by playing $L$. As a result, playing $L$ is a best response for agent 2 as well. Since the chosen strategy profile is known to be a best response for both agents, it satisfies the epistemic variant $KNE$ of the existence of a Nash equilibrium.

Example 4. [Kingmaker] Consider the SLi formula

$$(\exists_o x_1 \exists_o x_2 \exists_o x_3 NE_{\varphi_1}(x_1, x_2, x_3)) \land (\exists_o x_1 \exists_o x_2 \exists_o x_3 NE_{\varphi_2}(x_1, x_2, x_3))$$

where

- $\varphi_1 = \bigvee_{p<q} \text{points}_p^l \land \text{points}_q^l$ says that player 2 gets more points than player 1,
- $\varphi_2 = \bigvee_{p<q} \text{points}_p^r \land \text{points}_q^r$ says that player 1 gets more points than player 2,
- and $NE_{\varphi}(\bar{x}) = NE(\bar{x}) \land \text{bind}(i, x_i)_{i \in [3]} \varphi$ expresses that $\bar{x}$ is a Nash equilibrium in which $\varphi$ holds.

The whole formula says that there are two Nash equilibria in which player 3 can decide which of the other players gets more points. This expresses a form of a kingmaker property, that occurs in certain forms of poker with mixed strategies such as Kuhn’s three-player, four-card poker [66] in which there are four cards, with values 1, 2, 3, 4, and each of the three players gets a single card (visible only to them), and after some rounds of betting the player with the highest card who has not folded wins the pot.

Since SLi includes equality of variables, we can express concepts that involve comparisons between strategies.

Example 5. [Unique Equilibria] The formula

$$(\exists_o x_1 \forall y. NE(y) \rightarrow \bigwedge_{i \in Ag} y_i = x_i)$$

expresses that there is a unique Nash equilibrium. Deciding if a game has a unique Nash equilibrium is relevant to the predictive power of the Nash equilibrium as a solution concept. Indeed, in case there are multiple equilibria, the outcome of the game cannot be uniquely pinned down.\(^1\)

Example 6. Consider a symmetric two-player game where $p(x, y)$ is the payoff to player 1 if she uses strategy $x$ and the opponent uses strategy $y$. Recall that a strategy $x$...
is evolutionary stable if, intuitively, no mutant strategy can replace \( x \) if all players are playing \( x \) [68]. Formally, \( x \) is an evolutionary stable strategy if for every \( y \neq x \), either
i) \( p(x, x) > p(y, x) \) or ii) \( p(x, x) = p(y, x) \) and \( p(x, y) > p(y, y) \). In case \( p(x, y) \) can only take on a finite number of values, we can express the concept of evolutionary stable strategies in SL\(_i\). Let \( p_i(x, y) \) be an atom denoting that \( p(x, y) = i \). Then the following SL\(_i\) formula defines that \( x \) is an evolutionary stable strategy:

\[
\forall y. (y \neq x) \rightarrow C_1 \lor (C_2 \land C_3)
\]

where \( C_1 \) is \( \lor_{i>j} (p_i(x, x) \land p_j(y, x)) \), and \( C_2 \) is \( \lor_i (p_i(x, x) \land p_i(y, x)) \), and \( C_3 \) is \( \lor_{i>j} (p_i(x, y) \land p_j(y, y)) \).

\[\blacksquare\]

Remark 5. Although winning strategies (Example 1) can be expressed in ATL\(_*\), richer solution concepts such as Nash equilibria in which agents have LTL goals are not expressible in ATL\(_*\), already for three players with reachability/safety goals and perfect information [69, Theorems 3 and 5]. We remark that winning strategies have been used to characterise the existence of Nash equilibria in some special cases [70, 71]; this holds, in particular, for two-player turn-based games of perfect information in which agents have LTL objectives that do not depend on finite prefixes of the play [8, Proof of Lemma 1]. A detailed study of the preservation of the existence of Nash equilibria under bisimulation is given in [72].

In case agents have imperfect-information, the existence of Nash equilibria is undecidable for three agents [51], and for two agents decidable, cf. [51]. For multiple-agents, [37] show that one can regain decidability (for a strategy logic similar to ours that can express the existence of Nash equilibria) by imposing a hierarchy on the agent observations. In contrast, in Section 3 we will show that one can regain decidability, and thus decide the existence of Nash equilibria, assuming agents use public actions (and we make no restriction on the agent observations).

2.4. The Model-checking Problem

In the rest of the paper we consider the following decision problem.

Definition 5 (Model Checking). The model-checking problem is defined as follows: given an interpreted system \( S \) and an SL\(_i\) sentence \( \varphi \), decide whether \( S \models \varphi \).

As expected, this problem is undecidable in general.

Theorem 6. Model checking IS against SL\(_i\) specifications is undecidable.

To see this, we observe that the model-checking problem for concurrent-game structures under perfect recall and imperfect information (iCGS) against specifications expressed in alternating-time temporal logic (ATL\(_{iR}\)) is undecidable. In fact, the latter problem is undecidable already for formulas of the form \( \langle A \rangle G p \) where \( |A| = 2 \) and \( |Ag| = 3 \) [24]. Since this is an adaptation of existing results, we only sketch the reduction. Specifically, in Appendix B we define the semantics of SL\(_i\) over iCGS and prove that the model-checking problem for a subclass of iCGS, namely the square iCGS, against SL\(_i\) is inter-reducible in polynomial time to the model-checking problem for IS against SL\(_i\) (this result is novel and may be of independent interest). A key property of square iCGS is that if agent \( i \) finds two states indistinguishable, then after applying the same
joint action, the resulting states are still indistinguishable to agent $i$ (this captures that, in an IS, an agent updates its local states using its local transition function $\tau_i$). Then, in Appendix C, we show that the undecidability proof in [24] can be adapted to hold for square iCGS.

As discussed in the introduction, a number of restrictions on the general setting have been explored to obtain decidability. In the next section we will define and study a class of IS for which the model-checking problem is decidable. Moreover, in order to make statements about the computational complexity of the problem, we need to specify how the inputs $S$ and $\varphi$ are represented. We use an explicit representation. In particular, the size of the $\text{SLi}$ formula, denoted $|\varphi|$, is the number of its symbols, and the size of the IS, denoted $|S|$, is the number of transitions in its global transition function restricted to the reachable global states. Here, a state $s$ is reachable if it occurs in some history of $S$ (recall that histories start in initial states and are consistent the global transition function). In particular, we do not measure the size of the labelling function, i.e., we assume that the number of atoms is fixed.

3. Public Action Interpreted Systems and $\text{SLi}$

In this section we introduce a class of interpreted systems and prove that their model checking problem, against $\text{SLi}$ specifications, is decidable. This result should be contrasted with the undecidability in the general case (Theorem 6).

3.1. Interpreted Systems with Public Actions

We introduce a class of interpreted systems (Definition 3) in which we distinguish explicitly public actions, that is, actions that are observable to all agents.

Definition 7 (IS with Public Actions). An interpreted system with public actions is a tuple

$$S = (Ag, \{L_{pr_i}, Act_i, Pb_{Act_i}, P_i, \tau_i\}_{i \in Ag}, S_0, AP, \lambda)$$

such that

$$(Ag, \{L_i, Act_i, P, \tau_i\}_{i \in Ag}, S_0, AP, \lambda)$$

is an interpreted system where, for every agent $i \in Ag$:

1. $Pb_{Act_i} \subseteq Act_i$ is the set of public actions of agent $i$;
2. $L_i = L_{pr_i} \times \prod_{j \in Ag}(Pb_{Act_j} \cup \{\Delta\})$ is the set of local states, where $\Delta$ is a fresh symbol;
3. the local transition function $\tau_i$ satisfies the property that $\tau_i(l, a) = (p', a')$ implies that for all $j \in Ag$, if $a_j \in Pb_{Act_j}$ then $a'_j = a_j$ and otherwise $a'_j = \Delta$.

The set $L_{pr_i}$ consists of the private (local) states of agent $i$. By the condition on the local transition functions, the public actions performed last are copied into the successor local states of all agents; and in case the last action is not public, $\Delta$ is copied instead. As a result, such actions are observable to every agent.
Remark 6. Even if an action is not in $P_\text{b}\text{Act}_a$ it may still be observed by all agents, i.e., it is recorded in the private local state of everybody. Thus, $P_\text{b}\text{Act}_a$ should really be considered as the set of all explicitly public actions of agent $a$.

Clearly, any system following Definition 7 is an interpreted system. Also notice that any interpreted system is isomorphic to some system adhering to Definition 7 for which $P_\text{b}\text{Act}_i = \emptyset$ for all $i \in \text{Ag}$. To prove the latter fact, consider the mapping $\theta : l \mapsto (l, \Delta)$; this can be lifted to a bijection $\theta$ between global states, which has the property that $\tau(s, a) = s'$ iff $\tau(\theta(s), a) = \theta(s')$. Given this, for convenience we will call systems conforming to Definition 7 simply interpreted systems (IS).

The next definition singles out interpreted systems in which all actions are public.

Definition 8. Let PAIS (Public-Action Interpreted Systems) denote the set of interpreted systems with public actions such that $\text{Act}_i = P_\text{b}\text{Act}_i$ for all $i \in \text{Ag}$.

We now discuss the expressive power of PAIS for modeling AI scenarios. Although all actions in PAIS are public, they can still model private update of an agent’s private state. For instance, if an agent’s private local state contains a Boolean variable $x$, then we can model a private update of the value of $x$ as follows. First, for every initial global state with $x = 0$ we ensure there is an identical global initial state except that $x = 1$, and vice versa. Second, the agent can update its variable with the public action “toggle the value of $x$”, which has the effect of replacing the value of $x$ by $1 - x$. In particular then, although the other agents know that the variable was toggled, if they could not distinguish between $x = 0$ and $x = 1$ before the action, then they can’t afterwards either.

Also, PAIS can model that an agent allows the other agents to see part of its local state. For instance, this can be done with the public action “the value of $x$ is 0”, which we assume can only be done by the agent who owns the variable $x$ if indeed $x = 0$.

Moreover, PAIS can be used to represent several AI scenarios of interest:

1. In community-card games such as Texas hold’em, each player is privately dealt some cards, which are combined with “community cards” that are dealt face up on the table. Moreover, all bidding is public. Single rounds, or a bounded number of rounds, of such games can be modelled as PAIS. Such single rounds appear, for instance, as endgames and other simplified forms of Poker [73, 74, 75, 66] and Bridge [41].

2. In epistemic puzzles such as the muddy children puzzle, the Russian cards puzzle, the consecutive numbers puzzle, and the sum-and-products puzzle (see [76]), all communication is public, and therefore they can be modelled as PAIS.

3. In distributed systems one of the basic communication primitives is to broadcast a message to all other components [77]. The exchange of such messages can be modelled via public actions.

We now give an example of a PAIS that represents a simple trick-taking card game.

Example 7. [Card Game] Consider an $r$-player card game parameterised by integers $k, l$ with $1 \leq l \leq k$. The game is played with $r$ many decks of $k$ cards numbered 1 through $k$. Each player starts with a subset of size $l$ of their deck of cards that only
they can see (the remaining cards are not used in the game). At each round the players simultaneously reveal one card and the player with the highest revealed card scores a point. The revealed cards are discarded. This is repeated until all the cards have been revealed, and the winner (if any) is the player that has the most points. This game can be formalised as a PAIS with the following components:

- $Ag = \{r\}$,
- $L_{pr_i}$ consists of all pairs of the form $(H, p)$ where $H \subseteq [k]$ represents the cards player $i$ currently holds, and $p \in [k] \cup \{0\}$ represents the number of points player $i$ currently has.
- $Act_i = \{\text{reveal}_m : m \in [k]\}$, i.e., $\text{reveal}_m$ is the action of revealing the card with value $m$. The set of local states $L_i$ consists of elements of the form $(H, p, a)$ where $(H, p) \in L_{pr_i}$ and $a \in \prod_{j \in Ag}(Act_j \cup \{\Delta\})$.
- $P_i(H, p, a) = \{\text{reveal}_m : m \in H\}$, i.e., one can only reveal a card one is holding.
- The local transition function $\tau_i$ maps the local state $(H, p, a)$ and joint action $a' = (\text{reveal}_{m_1}, \ldots, \text{reveal}_{m_r})$ to the local state $(H', p', a')$ where $H' = H \setminus \{m_i\}$, and $p' = p$ if $m_i \leq m_j$ for some $j \neq i$, or $p' = p + 1$ otherwise.
- $AP = \{\text{points}_j^i : i \in Ag, j \in [k]\} \cup \{\text{end}\}$
- $\lambda$ is defined as follows:
  - it maps $\text{end}$ to the global states $((H_i, w_i, a_i)_{i \in Ag})$ such that $H_i = \emptyset$ for all $i$, i.e., $\text{end}$ holds if all the cards have been revealed;
  - it maps $\text{points}_j^i$ to the global states $((H_i, w_i, a_i)_{i \in Ag})$ such that $w_i = j$, i.e., player $i$ has scored $j$ points.

Note that we do not put the entire history of the play into the local state as we assume perfect recall semantics.

PAIS can be used similarly to model repeated games in normal form with unknown initial types, more complicated scoring mechanisms (for instance, that take ties into account), infinitely many rounds (for instance, by giving each player a “pass” action that does not play a card in that round, or allowing players to reuse played cards), turn-based games including endplay scenarios in bridge (as in [41]).

Finally, we compare PAIS to similar formalisms in the literature. A broadcasting environment [40] is defined as an IS with a distinguished agent called the environment, in which each agent’s local state $L_i$ consists of two pieces of information: 1) a private part $P_i$ that only depends on its local actions, and 2) a shared part that is the value of some fixed function $obs : L_e \rightarrow O$ (the same function for all agents) of the local state of the environment agent. Similarly, as discussed above, a PAIS can model that an agent can update its private variable, as well as allow other agents to observe just part of its local state.

A deterministic partially observable Markov decision process (POMDP) [52, 53] is a POMDP in which the transition function and observation functions are deterministic. In particular, the only stochasticity is in the initial distribution. A PAIS is also a deterministic transition system, except that instead of an initial distribution, it has an initial set of states.
3.2. Model Checking PAIS against SLi specifications

In this section we prove that model checking PAIS against SLi specifications is decidable. Then, in Section 3.5 we provide an analysis of the computational complexity of the problem.

**Theorem 9.** Model checking PAIS against SLi specifications is decidable.

Before proving Theorem 9, we outline a standard approach for evaluating the complexity of model-checking strategic logics, and discuss how to adapt it to the setting at hand. The basic idea involves encoding strategies $\sigma$ as trees $T$. Typically, the domain of $T$ is the set of histories of the system, and a node $h$ is labelled by action $\sigma(h)$. A mapping is then made from formulas into a formalism (such as tree automata or a branching-time logic) that can process trees. For instance, one approach used by algorithms for model checking SL [10] and ATL* [2] is to effectively convert $\varphi$ into a tree automaton that accepts exactly the trees $T$ that code the strategies $\sigma$ that make the formula $\varphi$ true (in the given game-structure, or model). This encoding cannot be used in the presence of imperfect information since the set of uniform strategies (for a given agent, in a given structure) is not the language of any tree automaton. Intuitively, the reason for this is that uniformity is a non-local restriction on the labels of the nodes of the tree that are “distant cousins”, and a tree automaton cannot tell if a distant cousin has the same label. One way to overcome this problem is to encode a strategy for an agent as a tree whose nodes are the set of sequences of observations of that agent. In this way, uniformity becomes a local condition. This approach can be used in the case of a single agent in an environment, or agents whose observations are hierarchical, in the sense that their indistinguishability relations are totally ordered by the refinement relation [78, 36, 37]. However, this encoding cannot be used in the multi-player case in which agents’ observations are not hierarchical. The reason, intuitively, is that neither tree of observations is a refinement of the other, and thus the automaton cannot encode the strategies that arise from incomparable observations.

Given the above, we require a novel encoding of strategies, which we now describe. The proposed encoding is based on the following insight: every history $h$ in a PAIS is uniquely determined by a pair $(s, \alpha)$ where $s$ is a state and $\alpha$ is a sequence of joint actions. Given this, strategies in a PAIS can be encoded as labellings of the tree $T$ whose domain consists of all sequences of joint actions. The labelling of a node $\alpha \in \text{Act}^*$ is a function that encodes, for every state $s$, the action of the strategy given that the starting state is $s$ and the sequence of joint actions were $\alpha$. Under this encoding, uniformity becomes a local condition on the tree. Also, the fact that all strategies are labellings of the same tree $T$ means that the infinite run determined by an assignment corresponds to an infinite path in $T$. This allows formalisms like tree-automata or branching-time logics to check properties of the path.

In what follows, we apply this encoding and translate SLi formulas and PAIS into formulas of a branching-time logic, rather than to tree automata. That is, we show how to reduce the model-checking problem of PAIS against SLi specifications to model checking regular trees against Quantified Computation Tree Logic (QCTL*). The logic QCTL* is a generalisation of CTL* that enables the quantification over atomic propositions [79]. This quantification will be used to simulate quantification over strategies. Model-checking regular trees against QCTL* specifications is decidable (and, in fact, is solved by tree
automata) [80]. In fact, QCTL* has been used as an intermediate logic between the low-level machinery of tree-automata and strategic logics such as ATL* with strategic contexts [81] and hierarchical SLi [37].

We begin by recalling the syntax and semantics of the logic QCTL* from [80].

3.3. Quantified Computation Tree Logic

This language of QCTL* adds quantification over atomic propositions to the syntax of CTL*.

Definition 10 (QCTL* Syntax). QCTL* state formulas \( \varphi \) and path formulas \( \psi \) are defined by the following grammar, where \( p \in AP \):

\[
\begin{align*}
\varphi & ::= p \mid \varphi \lor \varphi \mid \neg \varphi \mid E \psi \mid \exists p. \varphi \\
\psi & ::= \varphi \mid \psi \lor \psi \mid \neg \psi \mid X \psi \mid \psi U \psi
\end{align*}
\]

Formulas in QCTL* are all and only the state formulas in Def. 10. The intuitive reading of linear-time operators X and U is the same as in SLi; whereas E is the existential path quantifier from CTL*. Finally, a quantified formula \( \exists p. \varphi \) is read as “there exists an assignment to atom \( p \) such that \( \varphi \) is true”. Clearly, universal quantification can be expressed as \( \forall p. \varphi \equiv \neg \exists p. \neg \varphi \).

In order to introduce the semantics of QCTL*, we need the notion of a tree. Let \( D \) be a finite set of directions, and let \( \Sigma \) be a finite set of labels. A \( D \)-ary domain \( \text{dom} \subseteq D^* \) is a non-empty prefix-closed set of strings over \( D \). A \( \Sigma \)-labelling of \( \text{dom} \) is a function \( \text{lab} : \text{dom} \to \Sigma \). A \( D \)-ary \( \Sigma \)-labelled tree (or simply tree) is a pair \( T = (\text{dom}(T), \text{lab}(T)) \).

A node of \( T \) is an element \( t \in \text{dom} \), and a path of \( T \) from a node \( t \) is an infinite sequence \( \pi = t_1t_2 \ldots \) such that \( t_1 = t \) and for every \( i \geq 1 \) there is a \( d \in D \) such that \( t_{i+1} = t_i \cdot d \) and \( t_{i+1} \in \text{dom} \). The set of all paths from \( t \) is denoted \( \text{Paths}(t) \).

We need introduce the “tree-based” semantics of QCTL*. Note that the alternative semantics of QCTL*, so called “structure-based semantics” is not suitable for our purposes. Both semantics are studied in [80].

Definition 11 (QCTL* Semantics). Define the satisfaction relation \( (T, t) \models \varphi \) for QCTL* state formulas \( \varphi \) and \( (T, \pi) \models \psi \) for QCTL* path formulas by induction on the formulas,
as follows:

\[
\begin{align*}
(T, t) &\models p \quad \text{if } p \in \text{lab}(t) \\
(T, t) &\models \varphi_1 \lor \varphi_2 \quad \text{if } (T, t) \models \varphi_i \text{ for some } i \in \{1, 2\} \\
(T, t) &\models \neg \varphi \quad \text{if } (T, t) \not\models \varphi \\
(T, t) &\models \exists p. \varphi \quad \text{if for some lab' } =_{p} \text{lab}, ((\text{dom}(T), \text{lab'}), t) \models \varphi \\
(T, t) &\models \exists \psi \quad \text{if for some } \pi \in \text{Paths}(t), (T, \pi) \models \psi \\
(T, \pi) &\models \varphi \quad \text{if } (T, \pi_1) \models \varphi \\
(T, \pi) &\models \psi_1 \lor \psi_2 \quad \text{if } (T, \pi) \models \psi_i \text{ for some } i \in \{1, 2\} \\
(T, \pi) &\models \neg \psi \quad \text{if } (T, \pi) \not\models \psi \\
(T, \pi) &\models X \psi \quad \text{if } (T, \pi_{\geq 2}) \models \psi \\
(T, \pi) &\models \psi_1 \lor \psi_2 \quad \text{if for some } j \geq 1, \text{ for all } k \in [1, j], (T, \pi_{\geq j}) \models \psi_2 \text{ and } (T, \pi_{\geq k}) \models \psi_1.
\end{align*}
\]

A formula \( \varphi \) is true in a tree \( T \), or \( T \models \varphi \), if \( \varphi \) is true in the initial node \( \epsilon \), i.e., \( (T, \epsilon) \models \varphi \). The tree-unwinding of a finite-state system is called a regular tree, and its size is the number of states of the finite-state system. In [80] it is proved that model checking QCTL*, that is, deciding whether a QCTL* formula \( \phi \) is true in a regular tree \( T \), is decidable. We report this result as the following theorem.

**Theorem 12 ([80]).** Model-checking regular trees against QCTL* specifications is decidable.

Later (Theorem 16), we will cite a theorem that gives the complexity of this decision procedure. Hereafter, our model checking procedure for SLi proceeds by translating the given SLi formula and PAIS into a QCTL* formula and regular tree, and then applying Theorem 12. We now present this reduction.

### 3.4. Reducing SLi to QCTL*

We first show that one can interchange histories of PAIS with pairs consisting of a state and a sequence of joint actions. Define the function \( \mu \) that maps a state \( s \) and a sequence of actions \( \alpha \) to the history it determines, i.e., the history that starts in state \( s \) and applies the sequence of joint actions \( \alpha \).

**Definition 13.** Let \( S \) be an IS. Define the function \( \mu : S \times \text{Act}^* \to \text{Hist}(S) \) such that, for all \( s \in S \):

1. \( \mu(s, \epsilon) \triangleq s \);
2. \( \mu(s, \alpha \cdot \bar{a}) \triangleq \mu(s, \alpha) \cdot \tau(\text{last}(\mu(s, \alpha)), \bar{a}) \), for \( \alpha \in \text{Act}^* \).

The function \( \mu \) is clearly onto. On the other hand, if \( S \) is a PAIS, then \( \mu \) is also one-to-one. To see this we will use the following central fact about PAIS.

**Fact 1.** For all states \( s', t' \) and joint actions \( \bar{a}, \bar{b} \), if \( \tau(s', \bar{a}) = \tau(t', \bar{b}) \) then \( \bar{a} = \bar{b} \).

To see this, simply note that the definition of PAIS implies that after a transition is taken, the last joint action is written into the local state of every agent.

**Proposition 1.** In a PAIS \( S \), the function \( \mu : S \times \text{Act}^* \to \text{Hist}(S) \) is one-to-one.
Proof. To show that $\mu$ is one-to-one we suppose that $\mu(s, \alpha) = \mu(t, \beta)$ and show that $s = t$ and $\alpha = \beta$. By definition of $\mu$ we have that $|\alpha| = |\beta|$, call this length $l \geq 0$.

If $l = 0$ then $\alpha = \beta = \epsilon$ and so $s = \mu(s, \epsilon) = \mu(t, \epsilon) = t$, as required. So, suppose $l > 0$. Then, $s = t$ since $s = \mu(s, \epsilon) = \text{first}(\mu(s, \alpha)) = \text{first}(\mu(t, \beta)) = \mu(t, \epsilon) = t$.

Finally, to check that $\alpha = \beta$, repeatedly apply fact 1. \qed

The relevance of $\mu$ being both onto and one-to-one, i.e., a bijection, is that we can treat histories in $\text{Hist}(S)$ and sequences in $S \times \text{Act}^*$ interchangeably. In particular, the following notation is well defined.

**Definition 14.** In a PAIS $S$, for every history $h \in \text{Hist}(S)$ let $\text{state}(h) \in S$ and $\text{actions}(h) \in \text{Act}^*$ denote the unique elements such that $\mu(\text{state}(h), \text{actions}(h)) = h$.

The following lemma characterises $\equiv_i$:

**Lemma 1.** For all agents $i$ and histories $h, h'$, we have that $h \equiv_i h'$ iff $\text{actions}(h) = \text{actions}(h')$ and $\text{state}(h) \sim_i \text{state}(h')$.

Proof. Recall that in an IS two histories are indistinguishable to agent $i$, i.e., $h \equiv_i h'$, if agent $i$ has the same sequence of local states in both $h$ and $h'$. Moreover, in a PAIS, the local state consists of the private state and the tuple of last actions of each agent (with a dummy action in the first state). Thus, if $h \equiv_i h'$ then $\text{actions}(h) = \text{actions}(h')$ (since, in particular, the joint actions are visible) and $\text{state}(h) \sim_i \text{state}(h')$ (since, in particular, the initial states of the histories are indistinguishable to agent $i$). For the direction from right to left, use the fact that agent $i$'s local state at a given point in time is determined by the local transition function $\tau_i$, and thus only depends on its local state in the previous point of time and on the last joint action. \qed

In what follows, all trees will have the same domain, i.e., $\text{dom} \triangleq \text{Act}^*$. We now define labellings over this domain. These labellings will encode the transition function of the PAIS and agent strategies.

**Encoding PAIS**

We now describe the labelling $\text{lab}_S$ that encodes the PAIS $S$; later we will consider labellings $\text{lab}_\nu$ that encode valuations $\nu$. The labelling $\text{lab}_S$ captures relevant information about the system, e.g., it records the current state, given that the history started in state $s$.

**New atoms.** To define $\text{lab}_S$, we introduce the following new sets of atoms:

- $\{\text{cur}(s, s') : s, s' \in S\}$: intuitively, $\text{cur}(s, s')$ holds in a node $\alpha$ if $s'$ is the result of applying the sequence of actions $\alpha$ to the state $s$.

- $\{\text{lastact}(i, a) : i \in \text{Ag}, a \in \text{Act}_i \cup \{\Delta\}\}$: intuitively, $\text{lastact}(i, a)$ holds in a node $\alpha$ if the last action done by agent $i$ is $a$ (here $\Delta$ represents the case that no action has yet been done).

- $\{\text{atom}(s, p) : s \in S, p \in \text{AP}\}$: intuitively, $\text{atom}(s, p)$ holds in a node $\alpha$ if the atom $p \in \text{AP}$ holds after applying the sequence of actions $\alpha$ to the state $s$.

- $\{\text{rel}(i, s, t) : i \in \text{Ag}, s, t, \in S\}$: intuitively, $\text{rel}(i, s, t)$ holds in a node $\alpha$ if the histories, resulting from applying the sequence of actions $\alpha$ starting with $s$ and $t$ are indistinguishable to agent $i$.

24
**Labelling \( \text{lab}_5 \).** We now define \( \text{lab}_5 \) by induction on the length of the elements in \( \text{dom} \triangleq \text{Act}^* \). For the base case, define \( \text{lab}_5(\epsilon) \) to be the union of the following sets of atoms:

1. \( \{ \text{cur}(s, s) : s \in S \} \); intuitively, the empty sequence of actions does not advance the state.
2. \( \{ \text{lastact}(i, \Delta) : i \in \text{Ag} \} \); intuitively, the first action of each agent is the dummy \( \Delta \) action.
3. \( \{ \text{atom}(s, p) : s \in \lambda(p) \text{ and } p \in \text{AP} \} \); intuitively, the atoms of \( \text{AP} \) that hold are those that hold in \( s \).
4. \( \{ \text{rel}(i, s, t) : s, t \in S, i \in \text{Ag}, \text{ and } s \sim_i t \} \); intuitively, the initial indistinguishability relations are the given observability relations.

For the inductive case, let \( \alpha \in \text{Act}^*, \bar{a} \in \text{Act} \), and define \( \text{lab}_5(\alpha \cdot \bar{a}) \) to consist of the union of the following sets of atoms:

1. \( \{ \text{cur}(s, t) : \text{ for some } s', t = \tau(s', \bar{a}) \text{ and } \text{cur}(s, s') \in \text{lab}_5(\alpha) \} \); intuitively, given that \( s \) was the starting state, the current state is \( \tau(s', \bar{a}) \) if in the previous step the state was \( s' \) and the joint action was \( \bar{a} \).
2. \( \{ \text{lastact}(i, a_i) : i \in \text{Ag} \} \); intuitively, if the last direction in the tree was \( \bar{a} \), then \( a_i \) was the last action of agent \( i \).
3. \( \{ \text{atom}(s, p) : \text{ for some } t \in S, t \in \lambda(p) \text{ and } \text{cur}(s, t) \in \text{lab}_5(\alpha \cdot \bar{a}) \} \); intuitively, \( p \) holds now assuming the initial state was \( s \).
4. \( \{ \text{rel}(i, s, t) : \text{ for some } s', t' \in S, \text{rel}(i, s, t) \in \text{lab}_5(\alpha), s' \sim_i t', \text{cur}(s, s'), \text{cur}(t, t') \in \text{lab}_5(\alpha \cdot \bar{a}) \} \); intuitively, the histories starting in \( s \) and \( t \) are indistinguishable to agent \( i \) if they were so one step prior and the current states are indistinguishable to agent \( i \).

The following properties of the labelling \( \text{lab}_5 \) follow from the definitions:

**Lemma 2.** For all \( \alpha \in \text{dom}, s, t \in S, i \in \text{Ag}, p \in \text{AP} \):

1. \( \text{cur}(s, t) \in \text{lab}_5(\alpha) \) iff \( t = \text{last}(\mu(s, \alpha)) \).
2. \( \text{atom}(s, p) \in \text{lab}_5(\alpha) \) iff \( \text{last}(\mu(s, \alpha)) \in \lambda(p) \).
3. \( \text{rel}(i, s, t) \in \text{lab}_5(\alpha) \) iff \( \mu(s, \alpha) \equiv_i \mu(t, \alpha) \).

**Proof.** We prove the first item by induction on \( \alpha \) (the proofs of the other items are similar). Suppose \( \alpha = \epsilon \). Then \( \text{cur}(s, t) \in \text{lab}_5(\alpha) \) iff \( s = t \) (by Definition of \( \text{lab}_5 \)) iff \( t = \text{last}(\mu(s, \alpha)) \), since \( \mu(s, \epsilon) = s \) by definition of \( \mu \). Suppose \( \alpha \neq \epsilon \). Then \( \text{cur}(s, t) \in \text{lab}_5(\alpha \cdot \bar{a}) \) iff there exists \( s' \) such that \( t = \tau(s', \bar{a}) \) and \( \text{cur}(s, s') \in \text{lab}_5(\alpha) \) (by Definition of \( \text{lab}_5 \)) iff there exists \( s' \) such that \( t = \tau(s', \bar{a}) \) and \( s' = \text{last}(\mu(s, \alpha)) \) (by induction hypothesis) iff \( t = \tau(\text{last}(\mu(s, \alpha)), \bar{a}) = \text{last}(\mu(s, \alpha \cdot \bar{a})) \), since \( \mu(s, \alpha \cdot \bar{a}) = \mu(s, \alpha) \cdot \tau(\text{last}(\mu(s, \alpha)), \bar{a}) \), by definition of \( \mu \).
Encoding strategies

Intuitively, a strategy is encoded by a labelling that maps a node \( \alpha \in \text{Act}^* \) to the function that maps the state \( s \) to the action suggested by the strategy on history \( \mu(s, \alpha) \). To capture this we introduce below new atoms \( str(x,s,a) \) and a labelling \( \text{lab}_\nu \) of the domain \( \text{dom} \) (Def. 15).

New atoms. We introduce the following new set of atoms:

\[- \{ str(x,s,a) : x \in \text{Var}, s \in S, a \in \text{Act} \}; \text{intuitively, } str(x,s,a) \text{ holds at a node } \alpha \in \text{Act}^* \text{ if the strategy for variable } x \text{ suggests action } a \text{ in history } \mu(s, \alpha). \]

Recall that \( \text{act} \) denotes \( \bigcup_i \text{Act}_i \), the set of all possible actions.

QCTL* formulas defining strategies, coherence, uniformity. For every \( x \in \text{Var} \) and \( Z \subseteq \text{Ag} \), define the following QCTL* formulas:

\[
\begin{align*}
\text{Uniq}^x & \triangleq \mathbf{AG} \bigwedge_{s \in S} \bigvee_{a \in \text{act}} str(x,s,a) \land \neg str(x,s,b) \\
\text{Cohe}^x_Z & \triangleq \mathbf{AG} \bigwedge_{i \in Z} \bigwedge_{s \in S} \bigvee_{a \in \text{P}_i(s_i)} str(x,s,a) \\
\text{Unif}^x_Z & \triangleq \mathbf{AG} \bigwedge_{i \in Z} \bigwedge_{t,t' \in S} \bigwedge_{a \in \text{Act}_i} \left[ \text{rel}(i,t,t') \land str(x,t,a) \rightarrow str(x,t',a) \right]
\end{align*}
\]

Intuitively, \( \text{Uniq}^x \) expresses that the atoms \( str(x,-,-) \) encode a strategy, i.e., a unique action is associated with every history, and \( \text{Cohe}^x_Z \) (resp., \( \text{Unif}^x_Z \)) expresses that the strategy is coherent (resp., uniform) for the agents in \( Z \). These facts are captured by the following lemma, whose proof follows from the definitions of the formulas and Definition 14.

In what follows, a tree \( T \) with domain \( \text{Act}^* \) will be labeled by \( \mathbb{P}(\text{AP}) \) where \( \text{AP} \) is the set of atoms introduced above. In particular, the labeling of \( T \) can be decomposed into two parts: \( \text{lab}_\nu \) which labels nodes of the tree by the atoms \( \text{cur}(-,-), \text{lastact}(-,-), \text{atom}(-,-), \) and \( \text{rel}(-,-,-) \); and \( \text{lab} \) which labels nodes of the tree by atoms of the form \( str(-,-,-) \).

Lemma 3. Fix a tree \( T = (\text{Act}^*, \mathbb{P}(\text{AP}) \cup \text{lab}) \), a variable \( x \in \text{Var} \), and a set \( Z \subseteq \text{Ag} \) of agents. Consider the relation \( R^x \subseteq \text{Hist} \times \text{Act} \) defined by \( R^x(h,a) \) iff \( str(x, \text{state}(h), a) \in \text{lab}(\text{actions}(h)) \).

1. \( T \models \text{Uniq}^x \) iff \( R^x \) represents a strategy, i.e., for every \( h \in \text{Hist} \) there exists a unique \( a \in \text{Act} \) such that \( R^x(h,a) \).

2. If \( R^x \) represents a strategy (as in item 1), then \( T \models \text{Cohe}^x_Z \) iff the strategy is coherent for the agents in \( Z \).

3. If \( R^x \) represents a strategy (as in item 1), then \( T \models \text{Unif}^x_Z \) iff the strategy is uniform for the agents in \( Z \).
Proof. For the first item, suppose that $T \models \text{Uniq}^z$ and let $h$ be a history. Let $\alpha = \text{actions}(h)$ and $s = \text{state}(h)$. Since $T \models \text{Uniq}^z$, there is a unique action $a$ such that $\text{str}(x, s, a) \in \text{lab}(\alpha)$, as required. Conversely, suppose that $R^z$ represents a strategy and let $\alpha \in \text{Act}^*$. For every $s \in S$ let $a$ be the unique action such that $R^z(\mu(s, a), a)$. By definition of $R^z$, for every $s$ there is a unique action $a$ such that $\text{str}(x, s, a) \in \text{lab}(\alpha)$, as required.

For the second item, we are given that $R^z$ represents the strategy (as in item 1). Suppose $T \models \text{Coh}_{\text{lab}}^z$, let $h$ be a history, and $i \in Z$. Let $\alpha = \text{actions}(h)$ and $s = \text{state}(h)$. Since $T \models \text{Coh}_{\text{lab}}^z$ we have that $\text{str}(x, s, b) \in \text{lab}(\alpha)$ for some $b \in P_i(s_i) \subseteq \text{Act}_i$. Since $R^z$ represents a strategy, this strategy maps $h$ to the unique action $a$ such that $\text{str}(x, s, a) \in \text{lab}(\alpha)$, and so $a = b$. Since $h$ was arbitrary, the strategy is coherent.

Conversely, suppose that the strategy represented by $R^z$ is coherent, let $\alpha \in \text{Act}^*$, $i \in Z$ and $s \in S$. Since $R^z$ represents a strategy, this strategy maps $h$ to the unique action $a$ such that $\text{str}(x, s, a) \in \text{lab}(\alpha)$. By coherency, $a \in P_i(s_i)$, as required.

For the third item, we are given that $R^z$ represents the strategy (as in item 1). Suppose $T \models \text{Uniq}^z$, let $i \in Z$, and let $h, h'$ be two histories such that $h \equiv_i h'$. By Lemma 1, we have that $\alpha \equiv \text{actions}(h) = \text{actions}(h')$ and $t \equiv \text{state}(h) \sim_i t' \equiv \text{state}(h')$. By Lemma 2, we have that $\text{rel}(i, t, t') \in \text{lab}_S(\alpha)$. Say the strategy maps history $h$ to action $a$. Then $\text{str}(x, t, a) \in \text{lab}(\alpha)$. Thus also $\text{str}(x, t', a) \in \text{lab}(\alpha)$, i.e., the strategy maps history $h' = \mu(t', a)$ to $a$. Since $h, h'$ and $i$ were arbitrary, conclude that the strategy is uniform for agents in $Z$. Conversely, suppose the strategy is uniform for agents in $Z$, let $\alpha \in \text{Act}^*$, $i \in Z, t, t' \in S$ and $a \in \text{Act}_i$. Further, suppose that $\text{rel}(i, t, t') \in \text{lab}_S(\alpha)$ and $\text{str}(x, t, a) \in \text{lab}(\alpha)$. By Lemma 2 we have that $h \equiv \mu(t, a) \equiv_i h' \equiv \mu(t', a)$. By uniformity, we have that the strategy maps $h$ and $h'$ to the same action. However $\text{str}(x, t, a) \in \text{lab}(\alpha)$ implies that the strategy maps $h$ to $a$. Thus the strategy also maps $h'$ to $a$, and so $\text{str}(x, t', a) \in \text{lab}(\alpha)$, as required.

\begin{definition} (labelling $\text{lab}_\nu$). To every valuation $\nu : \text{Var} \rightarrow \text{Str}$ we associate the labelling $\text{lab}_\nu$ over the atoms defined as follows:

\begin{itemize}
  \item $\text{str}(x, s, a) \in \text{lab}_\nu(\alpha)$ iff $\nu(x)(\mu(s, a)) = a$.
\end{itemize}

In the introduction to this section we suggested that we would encode strategies in such a way that uniformity becomes a local condition. The next remark explains this.

\begin{remark}
Unifor\emph{mity is a property of $\text{lab}_\nu$ that can be checked at each node independently. More precisely, the strategy $\nu(x)$ is uniform for agent $i$ iff for every node $\alpha \in \text{Act}^*$, states $s \sim_i s'$, and action $a \in \text{act}$, we have that $\text{str}(x, s, a) \in \text{lab}_\nu(\alpha)$ iff $\text{str}(x, s', a) \in \text{lab}_\nu(\alpha)$.
\end{remark}

\begin{reduction} \text{SLi} to \text{QCTL}^\ast \end{reduction}

The following proposition shows how to reduce the model checking problem for \text{SLi} to the same problem for \text{QCTL}^\ast. Intuitively, we reduce checking whether $S$ satisfies $\varphi$ under assignment $(\nu, \beta)$ to verifying if the tree $(\text{dom, \text{lab}}_S \cup \text{lab}_\nu)$ satisfies the translation $\varphi_{\beta, \text{state}(h)}$ in \text{QCTL}^\ast. The domain $\text{dom}$ of the tree is $\text{Act}^*$, and its labelling is $\text{lab}_S \cup \text{lab}_\nu$, where $\text{lab}_S$ and $\text{lab}_\nu$ were defined earlier.
Proposition 2. For every PAIS S, assignment (ν, β), history h ∈ Hist(S), and SLi formula ϕ, there is a QCTL* formula ϕβ,state(h) that depends on ϕ, S, β and state(h), such that

(S, h, ν, β) |= ϕ if and only if ((dom, labS ∪ labν), actions(h)) |= ϕβ,state(h) \[3\]

Moreover, ϕβ,state(h) is computable in polynomial time on the size of ϕ.

Before we prove this proposition, we show how to use it to conclude that model-checking PAIS against SLi specifications is decidable (Theorem 9).

Proof of Theorem 9. Recall that a tree is regular if it is the tree-unwinding of a finite-state system. Given a PAIS S, history h, and SLi sentence ϕ, apply the following steps:

1. Pick an assignment (ν, β) so that (dom, labS ∪ labν) is a regular tree;
2. Form the QCTL* formula ϕβ,state(h) from Proposition 2;
3. Decide if (dom, labS ∪ labν) | h = ϕβ,state(h) using Theorem 12.

Note that since ϕ is a sentence, its truth does not depend on the particular assignment (ν, β) chosen. Thus, we have that the answer in the last step is “Yes” iff (S, h) |= ϕ.

Proof of Proposition 2. To prove this proposition we first describe how to construct the QCTL* formula ϕβ,state(h), and then prove that it is correct, i.e., that equivalence (3) holds.

Constructing ϕβ,state(h). Recall that Act = ∪i Acti is the set of all actions. We define the QCTL* formula ϕβ,s (for β : Ag → Var and s ∈ S) inductively:

− if ϕ is p ∈ AP, define ϕβ,s ≜ atom(s, p).
− if ϕ is x = y define
  \[\phi_{β,s} ≜ \bigwedge_{t \in S} \bigwedge_{a \in Act} AG (str(x, t, a) ↔ str(y, t, a))\]
− if ϕ is ϕ′ ∨ ϕ″ define ϕβ,s ≜ (ϕ′)β,s ∨ (ϕ″)β,s.
− if ϕ is ¬ϕ define ϕβ,s ≜ ¬(ϕ)β,s.
− if ϕ is bind(i, x)ϕ′ define ϕβ,s ≜ ϕ′[i=x].s.
− if ϕ is ∃xϕ′ define
  \[\phi_{β,s} ≜ (∃str(x, t, a))_{t \in S, a \in Act} [Uniq^x ∧ Cohe_{use(x, ϕ)}^x ∧ Unif^x_{use(x, ϕ)} ∧ (ϕ')_{β,s}]\]

where the atoms str(x, t, a) and the formulas Uniq^x, Cohe^x_{use(x, ϕ)}, Unif^x_{use(x, ϕ)} are defined above (for arbitrary x and Z).

\[2\] This is not hard to do, e.g., let β assign each agent a different variable, and for each local state si fix an action ai ∈ Pi(si) and define ν(x)(h) = ai, incase β(i) = x and last(h)i = si.
- If $\varphi$ is $\exists_x \varphi'$ define $\varphi_{\beta,s}$ in a similar way to the $\exists_\varphi$-case except that the last conjunct (i.e., $\varphi'_{\beta,s}$) is replaced by
\[
\bigwedge_{i \in \text{use}(x, \varphi)} \bigwedge_{t \in S} (\text{rel}(i, s, t) \rightarrow \varphi'_t).
\]
- If $\varphi$ is $\chi \varphi'$ then $\varphi_{\beta,s} \triangleq E(\text{IsPath}_{\beta,s} \land \chi \varphi'_{\beta,s})$.
- If $\varphi$ is $\varphi' \cup \varphi''$ then $\varphi_{\beta,s} \triangleq E(\text{IsPath}_{\beta,s} \land \varphi'_{\beta,s} \cup \varphi''_{\beta,s})$.

In the last two items we use a $\text{QCTL}^*$ path formula that depends on the binding $\beta : \text{Ag} \rightarrow \text{Var}$ and state $s \in S$, defined as follows:

\[ \text{IsPath}_{\beta,s} \triangleq G \bigwedge_{i \in \text{Ag}} \bigwedge_{a \in \text{Act}_i} (\text{lastact}(i, a) \rightarrow \text{str}(\beta(i), s, a)) \]

where $\text{lastact}(i, a)$ are new atoms introduced above. Intuitively, $\text{IsPath}_{\beta,s}$ holds of a path $\pi$ starting at node $\alpha \in \text{dom}$ if there is an infinite path extending $\mu(s, \alpha)$ such that agent $i$ follows the strategy associated with variable $\beta(i) \in \text{Var}$.

This completes the construction of $\varphi_{\beta,s}$.

**Proof that the construction is correct.** To prove that equivalence 3 in Proposition 2 holds, we proceed by induction on $\varphi$.

- If $\varphi$ is $p \in \text{AP}$, then we have $(S, h, \mu, \beta) \models p$
  \[ \iff \text{last}(h) \in \lambda(p) \]  dfn of $\models$ in $\text{SLi}$
  \[ \iff \text{last}(\mu(\text{state}(h), \text{actions}(h))) \in \lambda(p) \]  dfn 14
  \[ \iff \text{atom}(\text{state}(h), p) \in \text{lab}(\text{actions}(h)) \]  lemma 2
  \[ \iff (\text{dom}, \text{lab}_5 \cup \text{lab}_v), \text{actions}(h) \models \text{atom}(\text{state}(h), p) \]  dfn of $\models$ in $\text{QCTL}^*$
  \[ \iff (\text{dom}, \text{lab}_5 \cup \text{lab}_v), \text{actions}(h) \models (p)_{\beta, \text{state}(h)} \]  construction

- If $\varphi$ is $x = y$, then we have $(S, h, \mu, \beta) \models x = y$
  \[ \iff \text{for all } h', h \leq_{\text{pref}} h' \text{ implies } \nu(x)(h') = \nu(y)(h') \]  dfn of $\models$ in $\text{SLi}$
  \[ \iff (T, \text{actions}(h)) \models \bigwedge_{t \in S} \bigwedge_{a \in \text{Act}_t} \text{AG}(\text{str}(x, t, a) \leftrightarrow \text{str}(y, t, a)) \]  lemma 3
  \[ \iff (T, \text{actions}(h)) \models (x = y)_{\beta, \text{state}(h)} \]  construction

where $T = (\text{dom}, \text{lab}_5 \cup \text{lab}_v)$.

- The case that $\varphi$ is a Boolean combination is immediate from the induction hypothesis. For instance, if $\varphi$ is $\neg \varphi'$ then $(S, h, \mu, \beta) \models \varphi$
  \[ \iff (S, h, \nu, \beta) \not\models \varphi' \]  dfn of $\models$ in $\text{SLi}$
  \[ \iff ((\text{dom}, \text{lab}_5 \cup \text{lab}_v), \text{actions}(h)) \not\models \varphi'_{\beta, \text{state}(h)} \]  induction
  \[ \iff ((\text{dom}, \text{lab}_5 \cup \text{lab}_v), \text{actions}(h)) \models \neg \varphi'_{\beta, \text{state}(h)} \]  dfn of $\models$ in $\text{QCTL}^*$
  \[ \iff ((\text{dom}, \text{lab}_5 \cup \text{lab}_v), \text{actions}(h)) \models \varphi_{\beta, \text{state}(h)} \]  construction
We mention that we can provide a slight but useful optimisation in the Remark 8.

This completes the proof of Proposition 2.

Remark 8. We mention that we can provide a slight but useful optimisation in the translation from SLi to QCTL*. Instead of dealing with consecutive temporal operators separately, we treat them as a single LTL formula. That is, we can view the syntax of SLi so that besides including the terms $X\varphi \ | \ \varphi U \varphi$ we also include arbitrary LTL formulas whose atoms are SLi formulas, e.g., we include $(\varphi \lor \varphi) U(XX \varphi)$. Then, in the reduction we add the corresponding items, e.g., if $\varphi\in (\varphi' \lor \varphi'')(XX \varphi')$, then $\varphi^\beta_s \triangleq E(\text{IsPath}_{\beta,s} \land ((\varphi')^\beta_s \lor (\varphi'')^\beta_s)) U(XX(\varphi'')^\beta_s)$. The consequence of this is that $\varphi^\beta_s$ has the form $E(\text{LTL}(\cdot))$ rather than the more general form $\text{CTL}^*(\cdot)$ where $\cdot$ stands for the translation of SLi formulas.

Similarly, we can treat sequences of quantifiers of the same type in a single step of the translation. That is, there are two types of quantifiers in SLi, i.e., $\exists_o$ and $\exists_s$. So, for instance, instead of treating a sequence of quantifiers $\exists_o x_1 \exists_o x_2 \ldots \exists_o x_k$ in $k$ separate
steps of the translation, we can treat them in one step and thus get a translated formula of the form
\[
(\exists \text{str}(x_1, t, a) \exists \text{str}(x_2, t, a) \ldots \exists \text{str}(x_k, t, a))_{t \in S, a \in \text{act}} \ldots
\]

These optimisations have consequences for the complexity of model-checking SLi, as we see next.

3.5. Computational Complexity of model-checking PAIS

In this section we provide upper and lower bounds on the computational complexity of model checking PAIS against SLi (recall that we use an explicit representation of the inputs $S$ and $\varphi$ to the model-checking problem; see Section 2.4).

We start with upper bounds.

**Upper Bound**

The complexity of the algorithm in the previous section depends on a) the complexity of model-checking QCTL* formulas, and b) the complexity of the translation of SLi formulas to QCTL* formulas. We analyse these components in turn.

The finest published upper bound for model checking QCTL* is based on the quantifier-block depth of a formula, i.e., the maximum, over all paths in the parse tree of the formula, of the number of consecutive sequences of quantifiers. Formally, for a QCTL* formula $\varphi$ define the quantifier-block depth, denoted $\text{depth}(\varphi)$, inductively as follows:

- $\text{depth}(p) = 0$
- $\text{depth}(\varphi_1 \lor \varphi_2) = \max_i(\text{depth}(\varphi_i))$
- $\text{depth}(\neg \varphi) = \text{depth}(\varphi)$
- $\text{depth}(\exists p \varphi) = \text{depth}(\varphi) + m$ where $m = 0$ if $\varphi$ starts with $\exists$, and $m = 1$ otherwise.
- $\text{depth}(E \psi) = \max_i(\text{depth}(\varphi_i))$, where $\varphi_i$ varies over the maximal state subformulas of $\psi$.

Recall that the complexity class $k$-EXPTIME consists of the decision problems that can be solved by a deterministic Turing machine running in time $O(\text{exp}_k(P(n)))$ where $P$ is a polynomial and $\text{exp}_k$ is defined inductively as follows: $\text{exp}_0(n) \equiv n$ and $\text{exp}_{k+1}(n) \equiv 2^{\text{exp}_k(n)}$.

**Theorem 16.** [80] The complexity of model-checking QCTL* formulas of quantifier-block depth $k \geq 1$ is $(k + 1)$-EXPTIME-complete.\(^3\)

To apply this result, we similarly define the quantifier-block depth of an SLi formula. Here, however, we have two types of quantifiers, $\exists_o$ and $\exists_s$, which are treated separately. Formally, for $\varphi \in \text{SLi}$, define $\text{depth}(\varphi)$ inductively:

- $\text{depth}(p) = \text{depth}(x = y) = 0$

\(^3\)The definition of quantifier-block depth given above coincides with that given in [80]. To see this, note that $\text{depth}(\varphi) = 0$ iff $\varphi \in \text{CTL}^*$, and that $\text{depth}(\varphi) \leq k + 1$ iff $\varphi$ is of the form $\text{CTL}^*(\exists p_1 \ldots \exists p_n \varphi')$ where $n$ varies over the positive integers and $\varphi'$ varies over the QCTL* formulas with $\text{depth}(\varphi') \leq k$.\)
- \text{depth}(\neg \varphi) = \text{depth}(X\varphi) = \text{depth}(\text{bind}(i, x)\varphi) = \text{depth}(\varphi)
- \text{depth}(\varphi_1 \lor \varphi_2) = \text{depth}(\varphi_1 \cup \varphi_2) = \max_i(\text{depth}(\varphi_i))
- \text{depth}(\exists x.\varphi) = \text{depth}(\varphi) + m \text{ where } m = 0 \text{ if } \varphi \text{ starts with } \exists x, \text{ and } m = 1 \text{ otherwise.}
- \text{depth}(\exists_\alpha x.\varphi) = \text{depth}(\varphi) + m \text{ where } m = 0 \text{ if } \varphi \text{ starts with } \exists_\alpha, \text{ and } m = 1 \text{ otherwise.}

The next lemma reports the size and quantifier-block depth of the QCTL* formula $\varphi_{\beta,s}$ in terms of those of the SLi formula $\varphi$ and the system $S$.

**Lemma 4.** Using translation $(\cdot)_{\beta,s}$ from the previous section, optimised as in Remark 8, we have:

1. The quantifier-block depth of $\varphi_{\beta,s}$ is the same as that of $\varphi$.

2. The size of the QCTL* formula $\varphi_{\beta,s}$ is polynomial in $|S|$ and $|\varphi|$, and exponential in the $\exists_\alpha$-block depth of $\varphi$.

**Proof.** For the first item, note that a block of $\exists$ quantifiers is introduced only in place of $\exists_\alpha$ or $\exists_\beta$.

For the second item, we calculate an upper bound $Bd(\varphi)$ on the size of the QCTL* formulas $\varphi_{\beta,s}$:

- if $\varphi$ is an atom $p \in AP$, then $Bd(\varphi) = O(|S|)$ (note that an atom $p$ is replaced by the formula $\text{atom}(s, p)$ which mentions $s$).
- if $\varphi$ is $x = y$ then $Bd(\varphi) = \text{Poly}(|S|)$.
- if $\varphi$ is an LTL formula $\psi$ over SLi formulas $\varphi_1, \ldots, \varphi_h$, then $Bd(\varphi) = \text{Poly}(|S|) + |\psi| + \sum |\varphi_i|$ (use Remark 8).
- if $\varphi$ is $\varphi' \lor \varphi''$ then $Bd(\varphi) = O(Bd(\varphi') + Bd(\varphi''))$.
- if $\varphi$ is $\neg \varphi'$ then $Bd(\varphi) = O(Bd(\varphi'))$.
- if $\varphi$ is $\text{bind}(i, x)\varphi'$ then $Bd(\varphi) = O(Bd(\varphi'))$ (since $|\varphi'_{\beta,s}| = |\varphi'_{\beta',s}|$ for all $\beta, \beta'$).
- if $\varphi$ is $\exists_\alpha x.\varphi'$ then $Bd(\varphi) = \text{Poly}(|S|) + Bd(\varphi')$.
- if $\varphi$ is $\exists_\beta x.\varphi'$ then $Bd(\varphi) = \text{Poly}(|S|) \times Bd(\varphi')$ (i.e., multiplicative), since $|\varphi'_{\beta,s}| = |\varphi'_{\beta',s,t'}|$ for all $t, t'$.

Note that each item, except the last, involves (at most) an additive factor of $\text{Poly}(|S|)$. Thus, we have the following invariant of all steps except the last: $|\varphi_{\beta,s}| = \text{Poly}(|S||\varphi|)$.

Conclude that the size of $\varphi_{\beta,s}$ is $\text{Poly}(|S||\varphi|)^9$ where $g$ is the $\exists_\beta$-block depth of $\varphi$. \qed

Following all considerations above, we conclude that deciding whether $(S, h) \models \varphi$ can be solved in time $\text{exp}_{d+1}(|\varphi|)|S|)$, where $d$ is the quantifier-block depth of $\varphi$ and $g$ is the $\exists_\beta$-block depth of $\varphi$. In particular, we obtain that:

**Theorem 17.** Model-checking PAIS against SLi formulas of quantifier-block depth $k \geq 1$ can be carried out in time $\text{exp}_{k+2}(|S||\phi|)$. In particular, the problem is in $(k+2)$-EXPTIME.
Lower Bound

We now provide a lower bound for the computational complexity of model checking PAIS against SLi specifications by reducing model checking concurrent game structures (CGS) against SL specifications under perfect information to the problem under analysis (we refer to Appendix B for a formal definition of CGS).

First, consider the class of interpreted systems with perfect information, i.e., for all $i, j \in \text{Ag}$, $L_i = L_j$ and $\tau_i = \tau_j$. We can show that every IS with perfect information can be seen as an IS with public actions only. Indeed, given an IS $S$ with perfect information, define the corresponding PAIS $S'$ by assuming that for all $i \in \text{Ag}$, $\text{Act}'_i = \text{Pb}_i \text{Act}'_i = \text{Act}_i$.

Next, we show that states $s = (l, \ldots, l) \in S$ and $(s, a) = ((l, a), \ldots, (l, a)) \in S'$ satisfy the same formulas $\phi$ in SLi. The proof is by induction on the structure of $\phi$ by using the following induction hypothesis:

$$(S, h, \nu, \beta) \models \phi \iff (S', h', \nu, \beta) \models \phi$$

where $h'$ is any history $(s_1, a_1), \ldots, (s_k, a_k)$ such that $h = s_1, \ldots, s_k$.

The case of interest is obviously for strategy operators. In particular, we have that for $s = (l, \ldots, l), s' = (l', \ldots, l') \in S$, $i \in \text{Ag}$, $s \sim_i s'$ iff $l = l'$ iff $(l, a) = (l', a)$ iff $((l, a), \ldots, (l, a)) \sim_i ((l', a), \ldots, (l', a))$. Hence, we have a (uniform) strategy in $S$ witnessing a strategy operator $\exists_{\nu}$ (equivalently, $\exists_\nu$) iff we have such a strategy in $S'$.

Next, observe that every CGS $S$ with perfect information (that is, $\sim_{\nu}$ is the identity relation for every $i \in \text{Ag}$) is square: for every states $s, s'$, agent $i \in \text{Ag}$, and joint action $J \in \text{Act}$,

1. if $s \sim_i s'$ then $T(s, J) \sim_i T(s', J)$;
2. if $s \sim_j s'$ for every $j \in \text{Ag}$, then $\lambda(s) = \lambda(s')$.

Therefore, by Corollary 1 in Appendix B, there is an IS $\theta(S)$ that can be computed in PTIME, and that agrees with $S$ on formulas of SLi (again, we refer to Appendix B for the details of mapping $\theta$). Moreover, $\theta(S)$ enjoys perfect information as well. In particular, by the discussion above, $\theta(S)$ can be seen as a PAIS. Thus, $\theta$ can be seen as a PTIME-computable truth-preserving mapping from CGS with perfect information to PAIS. To summarise:

Lemma 5. There is a PTIME-computable map $\theta : \text{CGS} \rightarrow \text{PAIS}$ that preserves the truth of SL formulas, i.e., for all $\varphi \in \text{SL}$ and $S \in \text{CGS}$, we have that $S \models \varphi$ iff $\theta(S) \models \varphi$.

The best-known lower bound on the complexity of model-checking CGS against SL specifications is $k$-EXPSPACE-hard for formulas of quantifier-block depth at most $k \geq 1$. This follows from a known reduction from the satisfiability problem for quantified linear-temporal logic (QLTL) [82]:

Theorem 18. [10] There is a linear-time reduction from the satisfiability problem for QLTL to the model-checking problem for CGS against SL specifications. Moreover, the reduction preserves the quantification structure of the formulas, and the quantifier-block depth in particular.

Thus, we conclude that the model-checking problem for CGS against SL specifications of quantifier-block depth at most $k$ is as least as hard as the satisfiability problem for
QLTL formulas of quantifier block-depth at most \( k \). The latter is known, for \( k \geq 1 \), to be \((k - 1)\)-EXPSPACE-complete for QLTL formulas in prenex-normal form consisting of \( k - 1 \) alternations of quantifiers \([82]\). Note that a formula in prenex-normal-form has \( k - 1 \) alternations of quantifiers if and only if its quantifier-block depth is \( k \).

In view of the above, we obtain the following.

**Theorem 19.** The model-checking problem for PAIS against SLi specifications of quantifier-block depth \( k \geq 2 \) is \((k - 1)\)-EXPSPACE-hard.

Note that there is a gap of three exponentials between the best known upper and lower bounds. This is mainly due to the fact that the lower bound is rather naïve, in the sense that it already holds for a special case, i.e., that of perfect information. We further discuss this gap in the next section.

4. Conclusions

One of the key difficulties in automated reasoning about strategic abilities in MAS lies in the fact that the corresponding model checking and synthesis problems are undecidable under the assumptions, common in AI, of incomplete information and perfect recall. This is the case even for relatively weak languages such as ATL. Yet, MAS applications often require specifications that are more expressive than ATL, e.g., capable of expressing game-theoretical solution concepts such as Nash equilibria. Identifying classes of systems for which these two desiderata can be combined remains a challenge. In this paper we contributed towards this aim.

Specifically, we introduced SLi, an extension of Strategy Logic for agents with imperfect information and synchronous perfect-recall. SLi includes quantifiers for both the objective and the subjective interpretation of strategy modalities. It also includes equality, which allows us to express uniqueness of equilibria. After observing that the model checking problem is undecidable, we showed that for a noteworthy subclass of systems, those in which all agents may only use public actions, admit decidable model checking and synthesis, and identified upper and lower bounds for the model-checking problem. We illustrated the expressivity of the formalism by phrasing rational synthesis under incomplete information, a set-up only recently explored \([51, 55]\), as an instance of model checking for SLi. This has the noteworthy consequence that rational synthesis is decidable in the framework of PAIS.

We showed that the verification problem is decidable, and established non-elementary upper and lower bounds. The exact complexity of model-checking MAS against SLi specifications remains open at this stage. We note that there are significant fragments, i.e., SL, for which the exact complexity is also similarly open. More sophisticated methods for resolving these and other bounds are needed.

Our framework does have some limitations. For instance, although we can model fair behaviour (using the LTL part of SLi), explicit probabilities are important for describing optimal play in games like Poker. Also, epistemic operators are important for describing how agent knowledge changes in rational distributed computing and rational verification scenarios like rational secret sharing. Although we can express agent knowledge using the derived epistemic operator \( K_a \) (see Remark 2), in our framework an agent is not informed of its own strategy, i.e., what an agent can know is not constrained by any
agent strategies; see [15] for a discussion about the relevance of informed semantics. That said, we view our framework as the first step towards even richer frameworks which include features such as probabilities and informedness.

References

[24] C. Dima, F. Tiplea, Model-checking ATL under imperfect information and perfect recall semantics is undecidable, CoRR abs/1102.4225.
Appendix A. SLi has well-defined semantics

Here we prove that the satisfaction relation in Definition 4 is well defined in the sense that at every step of the inductive definition, valuation-binding pairs introduced on the right-hand sides are indeed assignments, and the valuations are compatible.

We recall the relevant definitions. An assignment is a pair $(\nu, \beta)$ such that for all $i \in A g$, the strategy $\nu(\beta(i))$ is coherent and uniform for $i$. A valuation $\nu$ is $\varphi$-compatible if for every variable $x \in \text{Var}$, the strategy $\nu(x)$ is coherent and uniform for every agent in $\text{use}(x, \varphi)$. By an abuse of terminology we say that an assignment $(\nu, \beta)$ is $\varphi$-compatible, whenever $\nu$ is.

Lemma 6. Suppose that $\chi = (\nu, \beta)$ is a $\varphi$-compatible assignment. Then,

- if $\varphi = \neg \varphi_1$ then $\chi$ is $\varphi_1$-compatible.
- if $\varphi = \varphi_1 \lor \varphi_2$ then $\chi$ is $\varphi_1$-compatible, for $i = 1, 2$.
- if $\varphi = X \varphi_1$ then $\chi$ is $\varphi_1$-compatible.
- if $\varphi = \varphi_1 \cup \varphi_2$ then $\chi$ is $\varphi_1$-compatible, for $i = 1, 2$.
- if $\varphi = \exists_x \varphi_1$ or $\varphi = \exists_x \varphi_1$, then $\chi[x \mapsto \sigma] = (\nu[x \mapsto \sigma], \beta)$ is an assignment that is $\varphi_1$-compatible for every $\sigma$ that is coherent and uniform for every agent in $\text{use}(x, \varphi_1)$.
- if $\varphi = \text{bind}(i, x) \varphi_1$ then $\chi[i \mapsto \chi(x)] = (\nu, \beta[i \mapsto \nu(x)])$ is an assignment that is $\varphi_1$-compatible.

Proof. We will make use of the following basic facts about $\text{use}(\cdot, \cdot)$. For all variables $x, y$:

- $\text{use}(x, \neg \varphi_1) = \text{use}(x, X \varphi_1) = \text{use}(x, \varphi_1)$.
- $\text{use}(x, \varphi_1 \lor \varphi_2) = \text{use}(x, \varphi_1 \cup \varphi_2) = \bigcup_{i \in \{1, 2\}} \text{use}(x, \varphi_i)$.
- $\text{use}(y, \exists_x \varphi_1) = \text{use}(y, \varphi_1)$ for all variables $y$ (where $\exists \in \{\exists_0, \exists_1\}$).
- $\text{use}(y, \text{bind}(i, x) \varphi_1) = \begin{cases} \text{use}(y, \varphi_1) & \text{if } x \neq y \\ \text{use}(y, \varphi_1) \cup \{i\} & \text{if } x = y \end{cases}$

The cases for the Boolean and temporal operators are similar to each other. We illustrate the case of disjunction $\varphi = \varphi_1 \lor \varphi_2$. To show that $\chi$ is $\varphi_1$-compatible, for $i = 1, 2$, take $x \in \text{Var}$. We need to prove that $\chi(x)$ is coherent and uniform for all agents in $\text{use}(x, \varphi)$, for $i = 1, 2$. By assumption $\chi(x)$ is coherent and uniform for every agent in $\text{use}(x, \varphi)$. Then, simply note that $\text{use}(x, \varphi_i) \subseteq \text{use}(x, \varphi)$, and therefore $\chi(x)$ is also coherent and uniform for every agent in each $\text{use}(x, \varphi_i)$.

Consider the case that $\varphi = \exists_x \varphi_1$ or $\varphi = \exists_x \varphi_1$. We show that $\chi[x \mapsto \sigma] = (\nu[x \mapsto \sigma], \beta)$ is $\varphi_1$-compatible, where $\sigma$ is coherent and uniform for every agent in $\text{use}(x, \varphi_1)$. To this end, take any $y \in \text{Var}$. We need to show that $\chi[x \mapsto \sigma](y)$ is coherent and uniform for all agents in $\text{use}(y, \varphi_1)$. There are two cases to consider. If $x \neq y$ we are done as $\chi[x \mapsto \sigma](y) = \chi(y)$ was assumed to be coherent and uniform for every agent...
in use(y, ϕ), which is equal to use(y, ϕ₁). If x = y then χ[x → σ](y) = σ, which was assumed to be coherent and uniform for every agent in use(x, ϕ₁).

For the binding operator, we first show that χ[i ↦ χ(x)] = (ν, β[i ↦ ν(x)]) is an assignment. To see this, we show that for every j ∈ Ag the strategy assigned to j by variant χ[i ↦ χ(x)] is coherent and uniform for j. If j ≠ i this follows from the fact that this strategy is χ(j), which was assumed to be coherent and uniform for j. If j = i, the strategy assigned by the variant is ν(x). Since χ is ϕ-compatible, ν(x) is coherent and uniform for every agent in use(x, bind(i, x)ϕ₁), and therefore for i as well. This shows that the variant χ[i ↦ χ(x)] is an assignment.

Second, we show that the variant χ[i ↦ χ(x)] is ϕ₁-compatible. To this end, let y ∈ Var. There are two cases to consider. If x ≠ y then the strategy χ[i ↦ χ(x)](y) is equal to χ(y), which was assumed to be coherent and uniform for every agent in use(y, ϕ) = use(y, ϕ₁). If x = y then the strategy χ[i ↦ χ(x)](y) is equal to χ(x), which was assumed to be coherent and uniform for every agent in use(x, ϕ) ⊇ (x, ϕ₁).

**Appendix B. Comparing IS and iCGS**

In this section we compare interpreted systems with a class of models for multi-agent systems that is mainstream in the literature on formal methods, i.e., concurrent game structures with imperfect information (iCGS) [2, 59]. In particular, we show that there exist truth-preserving polynomial translation between IS and iCGS. This result is used in Appendix C to prove Theorem 6.

**Definition 20 (iCGS).** An iCGS M is a tuple

\[
G = (Ag, AP, \{Act_i\}_{i \in Ag}, G, G_0, \{V_i\}_{i \in Ag}, T, \{\sim_i\}_{i \in Ag}, \lambda)
\]

where

1. Ag, AP, and Act_i, for every i ∈ Ag, are defined as in Definition 3. That is, Ag is the finite non-empty set of agent names; AP is the finite non-empty set of atoms; and Act_i is the finite non-empty set of actions of agent i.

2. G is the finite non-empty set of global states, with G_0 ⊆ G the non-empty set of initial states.

3. \(V_i : G \rightarrow 2^{Act_i} \setminus \{\emptyset\}\) maps state s to the non-empty set \(V_i(s)\) of available actions for i in s, so that \(V_i(g) = V_i(g')\) whenever \(g \sim_i g'\).

4. Writing \(ACT = \prod_{i \in Ag} Act_i\) for the set of joint actions, \(T : G \times ACT \rightarrow G\) is the partial transition function such that for \(s \in G\) and \(J \in ACT\), \(T(s, J)\) is defined iff \(J(i) \in V_i(s)\) for all \(i \in Ag\).

5. For every \(i \in Ag\), the indistinguishability relation \(\sim_i \subseteq G \times G\) is an equivalence relation.

6. \(\lambda : AP \rightarrow 2^G\) is the labelling function.
Observe that in iCGS global states and indistinguishability relations are primitive, rather than being defined via local states as for IS.

We define histories in iCGS as finite sequences respecting the transition function \( T \). We define the semantics of SLi on iCGS by adapting from Def 4, as follows, where \( h \) is a history, \( \varphi \) is an SLi formula, \( \nu \) is a valuation of the variables that is \( \varphi \)-compatible, and \( \beta \) is a binding of the agents, such that \( (\nu, \beta) \) is an assignment:

\[
\begin{align*}
(G, h, \nu, \beta) \models p & \quad \text{if last}(h) \in \lambda(p) \\
(G, h, \nu, \beta) \models x = y & \quad \text{if for every } h' \text{ extending } h, \text{ we have that } \nu(x)(h') = \nu(y)(h') \\
(G, h, \nu, \beta) \models \neg \psi & \quad \text{if } (G, h, \nu, \beta) \not\models \psi \\
(G, h, \nu, \beta) \models \psi_1 \lor \psi_2 & \quad \text{if } (G, h, \nu, \beta) \models \psi_1 \text{ or } (G, h, \nu, \beta) \models \psi_2 \\
(G, h, \nu, \beta) \models X \psi & \quad \text{if } (G, \pi_{\leq|\!|_1}(h, \nu, \beta), \nu, \beta) \models \psi \\
(G, h, \nu, \beta) \models \exists_i x \psi & \quad \text{if for some } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } use(x, \psi), \text{ we have that } (G, h, \nu[x \mapsto \sigma], \beta) \models \psi \\
(G, h, \nu, \beta) \models \exists_i x \psi & \quad \text{if for some } \sigma \in \text{Str} \text{ that is coherent and uniform for every agent in } use(x, \psi), \text{ we have that } (G, h', \nu[x \mapsto \sigma], \beta) \models \psi \\
& \quad \text{for every } h' \sim_i h \text{ and every } i \in use(x, \psi) \\
(G, h, \nu, \beta) \models \text{bind}(i, x) \psi & \quad \text{if } (G, h, \nu, \beta[i \mapsto x]) \models \psi
\end{align*}
\]

Next, we show that IS can be considered as a subclass of iCGS.

**Definition 21.** An iCGS is called square, collectively denoted \( \text{iCGS}_\square \), if for every \( s, s' \in G, i \in Ag \) and \( J \in \text{Act} \),

1. if \( s \sim_i s' \) then \( T(s, J) \sim_i T(s', J) \);

2. if \( s \sim_j s' \) for every \( j \in Ag \), then \( \lambda(s) = \lambda(s') \).

We can now prove the main theoretical result of this section.

**Theorem 22.** There are PTIME-computable maps \( \Theta : \text{IS} \to \text{iCGS}_\square \) and \( \theta : \text{iCGS}_\square \to \text{IS} \) that preserve SLi formulas, i.e.,

1. For every SLi formula \( \varphi \) and \( S \in \text{IS} \), \( S \models \varphi \) iff \( \Theta(S) \models \varphi \).

2. For every SLi formula \( \varphi \) and \( S \in \text{iCGS}_\square \), \( S \models \varphi \) iff \( \theta(S) \models \varphi \).

**Proof.** For the first item, every IS \( S = (Ag, \{L_i, Act_i, P_i, \tau_i\}_{i \in Ag}, S_0, AP, \lambda) \) induces the iCGS \( \Theta(S) = (Ag, AP, \{Act_i\}_{i \in Ag}, S, S_0, \{P_i\}_{i \in Ag}, \tau, \{\sim_i\}_{i \in Ag}, \lambda) \), where

1. \( Ag, AP, Act_i \) (for every \( i \in Ag \)), and \( \lambda \) are the same as in \( S \).

2. \( S \) (resp \( S_0 \)) is the set of global states (resp. initial global states) of \( S \).

3. For every \( i \in Ag, P_i \) is the local protocol in \( S \).

4. \( \tau \) is the global transition function of \( S \).

5. each \( \sim_i \) is the indistinguishability relation for agent \( i \in Ag \), such that \( s \sim_i s' \) iff \( s_i = s_i' \).
Corollary 1. The model-checking problems for IS against

by adapting the proof in [24] that model-checking

SLi is undecidable (Theorem 6). Since ATL is a fragment of SLi (see Remark 4), it is sufficient to prove that model checking IS against ATL is undecidable. We do this by adapting the proof in [24] that model-checking ATL on iCGS is undecidable. The

Appendix C. Undecidability of model checking IS against SLi

This section is devoted to proving that model checking interpreted systems against SLi is undecidable (Theorem 6). Since ATL is a fragment of SLi (see Remark 4), it is sufficient to prove that model checking IS against ATL is undecidable. We do this by adapting the proof in [24] that model-checking ATL on iCGS is undecidable. The
latter result is shown by a reduction of the halting problem for Turing machines (TM), i.e., for every TM $M$ one can construct an iCGS $S_M$ with three agents 1, 2, and 3 such that $M$ does not halt on the empty word as input iff $S_M \models \langle \{1, 2\}\rangle G \text{ok}$. Moreover, we can make a minor modification to the definition of $S_M$ in [24] so that the resulting iCGS is square (Definition 21). Thus, we can conclude (by Corollary 1) that model checking IS over ATL$_{IR}$ specifications is undecidable as well. We here outline the adapted proof, in particular we describe the modification in the definition of $S_M$ w.r.t. [24] and explain why the undecidability result still holds, without providing all partial results, which are immediate adaptations from [24]. We first need to introduce some notions.

A deterministic Turing machine can be defined as a tuple $M = (Q, \Sigma, q_0, B, \delta)$, where $Q$ is a finite set of states, $\Sigma$ is a finite alphabet, $q_0$ is the initial state, $B \in \Sigma$ is the blank symbol, and $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}$ is a partial transition function, where “L” specifies a “left move” and “R” specifies a “right move”.

Given a Turing machine $M$ as above, in [24] the authors define the iCGS $S_M = \langle Ag, AP, \{Act_j\} \in Ag, G, G_0, T, \{\sim_j\} \in Ag, \lambda \rangle$ simulating the computations of $M$, for $Ag = \{1, 2, 3\}$. Intuitively, the configurations of the Turing machine are encoded horizontally in the levels of the computation tree of $S_M$. To encode such configurations, [24] considers designated states to encode the left border of the tape, the separators between cells, the cell content, as well as the current position of the reading head. Here we present a slightly modified version of $S_M$ to fit our purposes.

First, the set $S$ of states, together with their intuitive meaning, includes

- $s_{init}$ the only initial state in $G_0$, with copy $s'_{init}$;
- $s_{lb}$ specifies the left border of $M$’s tape, with copy $s'_{lb}$;
- $s_{gen}$ initiates the generation of a new blank cell of $M$’s tape;
- $s_{tr}$ initiates the generation of a new cell separator;
- $s'_{tr}$ used for transferring information between two equivalent runs;
- $s_a$ for any $a \in \Sigma$, specifies that some tape cell holds $a$;
- $s_{q,a}$ for any state $q \in Q$ and $a \in \Sigma$, specifies that $M$ is in state $q$ and the read/write head points a cell holding symbol $a$;
- $s_{q,q',X}$ for any $q, q' \in Q$ and $X \in \{L, R\}$ such that $\delta(q, a) = (q', a', X)$ for some $a$ and $a'$, specifies that the machine $M$ enters state $q'$ from state $q$ by an $X$-move;
- $s_{err}$ “error” state used to collect all “unwanted” transitions the agents must avoid.

The set of atomic propositions is $AP = \{p_1, p_2, \text{ok}\}$ and the labelling function $\lambda$ is defined as:

$$\lambda(s) = \begin{cases} \{\text{ok}\} & \text{if } s \in G \setminus \{s_{gen}, s_{tr}, s_{err}\} \\ \{p_1, \text{ok}\} & \text{if } s = s_{gen} \\ \{p_2, \text{ok}\} & \text{if } s = s_{tr} \\ \emptyset & \text{if } s = s_{err} \end{cases}$$

In particular, all states but $s_{err}$ are labelled by $\text{ok}$. Hence, formula $\langle \{1, 2\}\rangle G \text{ok}$ basically means that agents 1 and 2 have a strategy to always avoid state $s_{err}$.

The indistinguishability relation $\sim_3$ is the identity (that is, agent 3 has perfect information as to the current state of the system); while relations $\sim_1$ and $\sim_2$ are defined as: $s \sim_1 s'$ iff $(p_i \in \lambda(s) \Leftrightarrow p_i \in \lambda(s'))$. Moreover, for agent 2, state $s_{init}$ is only indistinguishable from itself. We stress that this latter condition on $s_{init}$ is the only difference...
between our definition of $S_M$ and the one in [24]. Also with this modification, the undecidability result holds. Moreover, the modified $S_M$ allows us to show that condition ($\square$) is satisfied, and therefore it can be represented as an interpreted system. Furthermore, the indistinguishability relations for agents 1 and 2 allow information to be transferred consistently between runs, at the various levels of computations.

The set $Act$ of actions includes:

- the idle action $i$;
- $q_0$, which is an action meant to set up the initial state of $M$;
- $(q,q',X)$, for $q, q' \in Q$ and $X \in \{L, R\}$ with $\delta(q, a) = (q', a', X)$ for some $a, a' \in \Sigma$;
- $br_1$ and $br_2$, which are two “branching” actions.

In particular, $Act_1 = Act_2 = Act \setminus \{br_1, br_2\}$ and $Act_3 = \{i, br_1, br_2\}$.

Finally, we report the definition of the transition relation $T$ in $S_M$ as defined in [24]. This definition is meant to be used in the proof of Lemma 8, whereby the iCGS $S_M$ is indeed square.

- $s_{init} \xrightarrow{\text{\tiny $(i,br_1)$}} s_{init}'$ and $s_{init} \xrightarrow{\text{\tiny $(i,br_2)$}} s_{gen}'$ and $s_{init} \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- $s_{init}' \xrightarrow{\text{\tiny $(1,1)$}} s_{lb}$ and $s_{init}' \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- $s_{lb} \xrightarrow{\text{\tiny $(1, (q_0), 1)$}} s_{lb}'$ and $s_{lb} \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- $s_{lb}' \xrightarrow{\text{\tiny $(1,1)$}} s_{lb}'$ and $s_{lb}' \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- $s_{gen} \xrightarrow{\text{\tiny $(1,br_1)$}} s_{B}$ and $s_{gen} \xrightarrow{\text{\tiny $(1,br_2)$}} s_{tr}$ and $s_{gen} \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- $s_{tr} \xrightarrow{\text{\tiny $(1,br_1)$}} s_{tr}'$ and $s_{tr} \xrightarrow{\text{\tiny $(1,br_2)$}} s_{gen}$ and $s_{tr} \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- for every $a \in \Sigma$, the transitions at $s_a$ are:
  - $s_a \xrightarrow{\text{\tiny $(1,1)$}} s_a$;
  - $s_B \xrightarrow{\text{\tiny $(1, (q_0), 1)$}} s_{q_0,B}$
  - $s_a \xrightarrow{\text{\tiny $(1, (q,q',X),1)$}} s_{q,a}$ for any action $(q,q',X)$ with $X \in \{R, L\}$;
  - $s_a \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- for every $q \in Q$ and $a \in \Sigma$, the transitions at $s_{q,a}$ are:
  - $s_{q,a} \xrightarrow{\text{\tiny $(1, (q,q',X),1)$}} s_{q,a}'$ whenever $\delta(q, a) = (q', a', X)$ with $X \in \{R, L\}$;
  - $s_{q,a} \xrightarrow{\text{c}} s_{err}$ for any other joint action $c$;
- the transitions at $s_{tr}'$ are:
  - $s_{tr}' \xrightarrow{\text{\tiny $(1,1)$}} s_{tr}'$. 

44
the transition relation $T$ represented as an interpreted system. To prove this result, we refer to the definition of Lemma 8. The iCGS $S$ now show that the iCGS information and perfect recall is undecidable.

As regards (1), the result is immediate for $s$. As for agent 1, notice that $s$ still works for our slightly modified version of iCGS with imperfect information and perfect recall to the halting problem for Turing machines $\langle\langle\{h\}\rangle\rangle$. We state this result formally in the following lemma.

**Lemma 7.** For all histories $h$ and $h'$ starting in the initial state $s_{init}$, $h \sim_2 h'$ iff $h$ and $h'$ are indistinguishable to agent 2 in the iCGS $S_M$ from [24].

**Proof.** We distinguish the cases for $|h| = |h'| = 1$ and $|h| = |h'| > 1$. In the former case, we have $h = h' = s_{init}$ and the result immediately holds. In the latter case, notice that $h_1 = h'_1 = s_{init}$ and for all $m > 1$, $h_m \neq s_{init}$ and $h'_m \neq s_{init}$. Hence, $h \sim_2 h'$ iff $h$ and $h'$ are indistinguishable to agent 2 according to [24].

As a consequence of Lemma 7, all partial results in [24] still hold for our iCGS $S_M$, including Claims 1-4. In particular, we are still able to prove that the ATL formula $\langle\langle\{1,2\}\rangle\rangle G ok$ is true in $S_M$ iff the Turing machine $M$ does not halt on the empty word. As an immediate consequence, we obtain the following undecidability result.

**Proposition 3.** The model checking problem for iCGS with respect to ATL with imperfect information and perfect recall is undecidable.

Finally, since we want to transfer the undecidability result to interpreted systems, we now show that the iCGS $S_M$ is square. As a consequence, by Theorem 22 $S_M$ can be represented as an interpreted system. To prove this result, we refer to the definition of the transition relation $T$ above.

**Lemma 8.** The iCGS $S_M$ is square. That is, for every agent $j \in Ag = \{1,2,3\}$, if $s \sim_1 s'$ then for every joint action $J \in Act$, $T(s,J) \sim_1 T(s',J)$. Moreover, if $s \sim_2 s'$ for every $j \in Ag$, then $\lambda(s) = \lambda(s')$.

**Proof.** As regards (1), the result is immediate for $a = 3$, as she has perfect information. As for agent 1, notice that $s \sim_1 s'$ iff either $s = s' = s_{gen}$ or both $s \neq s_{gen}$ and $s' \neq s_{gen}$. In the former case, the result holds as transition $\tau(s_{gen},J)$ is deterministic for a given joint action $J$. As to the latter, to obtain a contradiction suppose that $\tau(s,J) \neq_1 \tau(s',J)$. Then, one of the two states has to be equal to $s_{gen}$, while the other is different. Assume that $\tau(s,J) = s_{gen}$ and $\tau(s',J) \neq s_{gen}$. By definition of $T$, the only possible cases are that $J = (i,i,br_2)$, $s' = s_{gen}$, and either $s = s_{init}$ or $s = s_{tr}$. A contradiction, as we assumed $s' \neq s_{gen}$. Further, for agent 2 we have $s \sim_2 s'$ iff either $s = s' = s_{tr}$, or $s = s' = s_{init}$, or both $s \notin \{s_{tr},s_{init}\}$ and $s' \notin \{s_{tr},s_{init}\}$. The first two cases follow as above by determinism of the transition function. As to the latter, suppose for contradiction that $\tau(s,J) \neq_2 \tau(s',J)$. Then, assume $\tau(s,J) \in \{s_{tr},s_{init}\}$ and $\tau(s',J) \neq \tau(s,J)$. By definition of $T$, the only possible cases are that $J = (i,i,br_2)$,
$s = s_{gen}$, and either $s' = s_{init}$ or $s' = s_{tr}$. In both cases we obtain a contradiction with the assumption $s' \notin \{s_{tr}, s_{init}\}$.

Finally, since agent 3 has perfect information, $s \sim_j s'$ for every $j \in Ag$ iff $s = s'$, and therefore $\lambda(s) = \lambda(s')$.

To sum up, we modified the definition of the indistinguishability relation for agent 2 in the iCGS $S_M$ for a Turing machine $M$ so that it is square (Lemma 8). This modification does not affect the proof of the results in [24], as both definitions of indistinguishability for agent 2 coincide on histories starting from the initial state $s_{init}$. In particular, model checking $\text{ATL}_{IR}$ on iCGS is still undecidable, but since $S_M$ is now square, it can be represented as an IS, and therefore by Theorem 22 the undecidability result applies to IS as well.