Practical Verification of Multi-Agent Systems against Slk Specifications

Petr Čermák[a], Alessio Lomuscio[a], Fabio Mogavero[b], Aniello Murano[c]

[a]*Imperial College London, UK*
[b]*University of Oxford, UK*
[c]*Università degli Studi di Napoli Federico II, Italy*

Abstract

We introduce Strategy Logic with Knowledge, a novel formalism to reason about knowledge and strategic ability in memoryless multi-agent systems with incomplete information. We exemplify its expressive power; we define the model checking problem for the logic and show that it is PSpace-complete. We propose a labelling algorithm for solving the verification problem that we show is amenable to symbolic implementation. We introduce MCMAS$_{\text{SLK}}$, an extension of the open-source model checker MCMAS, implementing the proposed algorithm. We report the benchmarks obtained on a number of scenarios from the literature, including the dining cryptographers protocol.

Keywords: model checking, Strategy Logic, multi-agent systems, formal verification

1. Introduction

Multi-agent systems (MAS) are distributed systems whereby the components, or *agents*, display a high degree of autonomy and interact with their peers in a cooperative or adversarial way in order to maximise their private or common goals [1]. Over the years several logics have been put forward to reason about MAS, including epistemic logics [2], deontic logics [3] and formal languages accounting for the beliefs, desires and intentions of the agents [4]. These formalisms, once combined with temporal logic, are more expressive than logics used to reason about reactive systems, such as Ltl and Ctl [5]. This is because when reasoning about MAS, it is often not sufficient to establish whether a particular temporal statement is realised. Instead, there is an interest in establishing whether high-level properties of the agents hold in the system. These may involve the evolution of their beliefs, the intentions they want to bring about, what regulations they are subjected to and the interplay of all of these.

Knowledge and strategic ability are two particular aspects of agency that are of importance when reasoning about MAS. By means of epistemic specifications, we can, for example, reason about what the agents know about the world, its evolution, their peers, their peers' knowledge, as well as epistemic group notions such as common knowledge [2]. By incorporating strategic abilities in the specifications, we can establish whether particular groups of agents have the ability to bring about certain temporal states of affairs.

There is a relatively long tradition in the development of verification technology, notably model checking, to verify MAS against temporal-epistemic specifications. This includes methods based on bounded [6, 7] verification, logical representation through Binary Decision Diagrams (*BDDs*, for short) [8, 9, 10], symmetry reduction [11] and abstraction [12, 13]. Verification techniques for validating systems against strategic abilities have also been put forward. For example, jMOCHA [14] is a model checker for the verification of systems against specifications in Alternating-time Temporal Logic (Atl). Proposals have also been made to devise methods supporting specifications that account for both the epistemic states of the agents as well as their strategic ability [15, 16, 17].

Also related to the present proposal is [18], where an extension of ATL for imperfect information games, namely CSLP, was introduced. CSLP can express sophisticated strategic and epistemic game properties of coalitions under uncertainty, including solution concepts. However, differently from the proposal made here, CSLP consider the agents' strategies implicitly, which are bound directly to the agents they refer to.

MCMAS [19, 20] is a BDD-based model checker supporting both Atl and epistemic specifications.

An important aspect in combining epistemic and strategic specifications is the information model the agents adhere to, namely whether they have either complete or incomplete information about the world and what form of memory they have. Epistemic analysis of MAS normally assumes that agents have private, incomplete information about the world. It is known that the model checking problem for Atl with incomplete information and perfect recall is undecidable [21]. Given this, the most widely adopted setting is incomplete information with memoryless local states. In turn, this implies that Atl modalities assume memoryless strategies. This raises further issues including whether local strategies should be uniform [22, 19, 17].

In this work we follow this tradition, but extend the strategic dimension of the analysis to a fragment of strategy logic [23]. A limitation of Atl is that while specifications relate to the strategies of the agents to achieve a certain state of affairs, the strategies themselves do not feature in the syntax explicitly; instead, they are treated implicitly through agent modalities that refer to coalitions. The logic Sl was introduced to overcome this by introducing strategies as first-class citizens of the syntax and by allowing explicit quantification and binding over them.

In this paper we define a combination of Sl with the standard epistemic modalities on a memoryless variant of interpreted systems, introduce an algorithm for model checking and present its implementation. A feature of the present work is the relatively low complexity of the model checking problem, which is shown to be PSpace-complete. As discussed in more details below, this is achieved by limiting the scope of the epistemic operators in the language.

The rest of the paper is organised as follows. In Section 2 we introduce the specification language SLK, define its semantics on a variant of interpreted systems, illustrate its use, define the model checking problem and investigate its complexity. In Section 3 we present a labelling algorithm to solve the model checking problem for SLK specifications against interpreted systems. In Section 4 we present MCMAS$_{\text{SLK}}$, a symbolic model checker derived from MCMAS, implementing the labelling algorithm, and present experimental results. We conclude in Section 5 by discussing related work.

2. Strategy Logic with Knowledge

In this section, we introduce Strategy Logic with Knowledge [24] (SLK, for short) as an extension of the original *Strategy Logic* [25, 26] (SL, for short) introduced in [23]. Our aim is to define a formalism combining the ability of SL to express game-theoretic concepts with an epistemic framework for describing the agents' knowledge in the context of incomplete information. In order to avoid the well-known undecidability result of the model-checking problem of multi-agent systems under incomplete information and perfect recall [27], we formalise the new logic by means of imperfect-recall semantics. This implies that agents have no memory of the past, including when planning their strategy to achieve a desired goal. SLK is defined *w.r.t.* interpreted systems [28], whereas SL has been introduced for concurrent game structures [27], since the former is normally used in the context of incomplete information.

In the rest of this section, we give formal definitions of SLK syntax and semantics, together with accessory concepts, such as strategy, profile and play. These provide us with a solid foundation for the development of the model-checking algorithm, which is theoretically presented in Section 3 and then developed as a software tool and benchmarked against several scalable scenarios in Section 4.

2.1. Syntax

SL syntactically extends liner-time temporal logic LTL [29] by introducing two *strategy quantifiers* $\langle\langle x \rangle\rangle$ and $[[x]]$, and an *agent binding* (a, x), where x is a variable and a an agent. Informally, these operators can be read as *"there exists a strategy x"*, *"for all strategies x"* and *"bind agent a to the strategy associated with x"*, respectively. SLK further extends SL with *epistemic modalities* [28] representing *individual knowledge* K_a, *group knowledge* E_{A}, *distributed knowledge* D_{A} and *common knowledge* C_{A}, where a is an agent and A a set of agents. Therefore, this language can be seen as a unique blend of three well-established logic formalisms that allows us to reason about the temporal, strategic and epistemic aspects of a model in a unified way.

SLK formulas are defined as follows.

Definition 1 (Syntax)**.** SLK formulas *are built inductively from the sets of atomic propositions* AP*, variables* Vr *and agents* Ag *using the following context-free grammar, where* $p \in \mathrm{AP}$*,* $x \in \mathrm{Vr}$*,* $a \in \mathrm{Ag}$ *and* $\mathrm{A} \subseteq \mathrm{Ag}$*:*

$$\varphi ::= \bot \mid \top \mid p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \mathsf{X}\,\varphi \mid (\varphi\,\mathsf{U}\,\varphi) \mid (\varphi\,\mathsf{R}\,\varphi) \mid$$
$$\langle\!\langle x\rangle\!\rangle\varphi \mid [\![x]\!]\varphi \mid (a,x)\varphi \mid \mathsf{K}_a\,\varphi \mid \mathsf{E}_{\mathrm{A}}\,\varphi \mid \mathsf{D}_{\mathrm{A}}\,\varphi \mid \mathsf{C}_{\mathrm{A}}\,\varphi$$

where the epistemic operators are applied to sentences only, i.e., $\mathsf{free}(\varphi) = \emptyset$ *in* $\mathsf{K}_a\varphi$*,* $\mathsf{E}_{\mathrm{A}}\varphi$*,* $\mathsf{D}_{\mathrm{A}}\varphi$ *and* $\mathsf{C}_{\mathrm{A}}\varphi$*, where the function* free *is introduced in Definition 2.* SLK *denotes the set of formulas generated by the aforementioned rules.*

As for SL, the *free agents and free variables* $\mathsf{free}(\varphi)$ of an SLK formula φ are the subset of $\mathrm{Ag} \cup \mathrm{Vr}$ containing *(i)* all agents a for which there is no binding (a, x) before the occurrence of a temporal operator and *(ii)* all variables x for which a binding (a, x) is not in the scope of any quantification $\langle\!\langle x\rangle\!\rangle$ or $[\![x]\!]$.

Definition 2 (Free Agents and free Variables)**.** *The set of* free agents and free variables *of an* SLK *formula is given by the function* $\mathsf{free}\colon \mathrm{SLK} \to 2^{\mathrm{Ag}\cup\mathrm{Vr}}$ *defined inductively as follows:*

1. $\mathsf{free}(\bot), \mathsf{free}(\top) \triangleq \emptyset$*;*
2. $\mathsf{free}(p) \triangleq \emptyset$*, where* $p \in \mathrm{AP}$*;*
3. $\mathsf{free}(\neg\varphi) \triangleq \mathsf{free}(\varphi)$*;*
4. $\mathsf{free}(\varphi_1\,\mathsf{Op}\,\varphi_2) \triangleq \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$*, where* $\mathsf{Op} \in \{\wedge, \vee\}$*;*
5. $\mathsf{free}(\mathsf{X}\,\varphi) \triangleq \mathrm{Ag} \cup \mathsf{free}(\varphi)$*;*
6. $\mathsf{free}(\varphi_1\,\mathsf{Op}\,\varphi_2) \triangleq \mathrm{Ag} \cup \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$*, where* $\mathsf{Op} \in \{\mathsf{U}, \mathsf{R}\}$*;*
7. $\mathsf{free}(\mathsf{Qn}\,\varphi) \triangleq \mathsf{free}(\varphi) \setminus \{x\}$*, where* $\mathsf{Qn} \in \{\langle\!\langle x\rangle\!\rangle, [\![x]\!] \mid x \in \mathrm{Vr}\}$*;*
8. $\mathsf{free}((a, x)\varphi) \triangleq \mathsf{free}(\varphi)$*, if* $a \notin \mathsf{free}(\varphi)$*, where* $a \in \mathrm{Ag}$ *and* $x \in \mathrm{Vr}$*;*
9. $\mathsf{free}((a, x)\varphi) \triangleq (\mathsf{free}(\varphi)\setminus\{a\}) \cup \{x\}$*, if* $a \in \mathsf{free}(\varphi)$*, where* $a \in \mathrm{Ag}$ *and* $x \in \mathrm{Vr}$*;*
10. $\mathsf{free}(\mathsf{K}_a\,\varphi), \mathsf{free}(\mathsf{E}_{\mathrm{A}}\,\varphi), \mathsf{free}(\mathsf{D}_{\mathrm{A}}\,\varphi), \mathsf{free}(\mathsf{C}_{\mathrm{A}}\,\varphi) \triangleq \emptyset$*, where* $a \in \mathrm{Ag}$ *and* $\mathrm{A} \subseteq \mathrm{Ag}$.

A formula φ *without free agents (*resp. *variables),* i.e.*, with* $\mathsf{free}(\varphi) \cap \mathrm{Ag} = \emptyset$ *(*resp. $\mathsf{free}(\varphi) \cap \mathrm{Vr} = \emptyset$*), is called* agent-closed *(*resp. variable-closed*). If* φ *is both agent-closed and variable-closed, it is called a* sentence.

Note that strategies can be *shared* among the agents. However, the interpreted systems on which we will later define the semantics allow agents to use different, possibly disjoint, sets of actions and thus also different sets of strategies. Therefore, to determine the set of strategies over which a variable can range, we need to know which agents are bound to it. We thus introduce the set of *sharing agents* that refers, intuitively, to the agents associated with the same given variable within a formula.

Definition 3 (Sharing Agents)**.** *The set of* sharing agents *of an* SLK *formula* w.r.t. *a variable is given by the function* $\mathsf{shr}\colon \mathrm{SLK} \times \mathrm{Vr} \to 2^{\mathrm{Ag}}$ *defined inductively as follows:*

1. $\mathsf{shr}(\bot, x), \mathsf{shr}(\top, x) \triangleq \emptyset$*;*
2. $\mathsf{shr}(p, x) \triangleq \emptyset$*, where* $p \in \mathrm{AP}$*;*
3. $\mathsf{shr}(\mathsf{Op}\,\varphi, x) \triangleq \mathsf{shr}(\varphi, x)$*, where* $\mathsf{Op} \in \{\neg, \mathsf{X}\} \cup \{\langle\!\langle y\rangle\!\rangle, [\![y]\!] \mid y \in \mathrm{Vr}\} \cup \{\mathsf{K}_a \mid a \in \mathrm{Ag}\} \cup \{\mathsf{E}_{\mathrm{A}}, \mathsf{D}_{\mathrm{A}}, \mathsf{C}_{\mathrm{A}} \mid \mathrm{A} \subseteq \mathrm{Ag}\}$*;*

4. $\mathsf{shr}(\varphi_1 \, \mathsf{Op} \, \varphi_2, x) \triangleq \mathsf{shr}(\varphi_1, x) \cup \mathsf{shr}(\varphi_2, x)$, *where* $\mathsf{Op} \in \{\wedge, \vee, \mathsf{U}, \mathsf{R}\}$*;*
5. $\mathsf{shr}((a, x)\varphi, x) \triangleq \{a\} \cup \mathsf{shr}(\varphi, x)$*;*
6. $\mathsf{shr}((a, y)\varphi, x) \triangleq \mathsf{shr}(\varphi, x)$, *where* $y \in \mathrm{Vr} \setminus \{x\}$*;*

Note that, for conciseness, we assume throughout this work that every variable is quantified at most once in a given formula. This can be easily ensured by renaming variables which are not free in the formula (*e.g.*, $(a_1, x)\langle\langle x\rangle\rangle(a_2, x)\mathsf{X}\, p \equiv (a_1, x)\langle\langle y\rangle\rangle(a_2, y)\mathsf{X}\, p \not\equiv (a_1, y)\langle\langle x\rangle\rangle(a_2, x)\mathsf{X}\, p$).

2.2. Multi-Agent Model

As mentioned before, we provide a semantics for SLK based on interpreted systems [2]. Differently from standard interpreted systems, here we take an agent's local states to be composed of private, or *internal*, states and states resulting from portions of the environment that are visible to the agent in question. This is similar to variants of interpreted systems such as broadcasting systems [30].

Definition 4 (Interpreted Systems)**.** *Let* $\mathrm{Ag} \triangleq \{1, \ldots, n\} \cup \{\mathrm{Env}\}$ *be a set of* agents*, or* players*, where* Env *is a distinguished element representing the* environment *and* $\Sigma \triangleq \{1, \ldots, n\}$ *is the set of* proper agents*, and* AP *a finite non-empty set of* atomic propositions*. An* interpreted system *is a tuple* $\mathcal{I} \triangleq \left\langle (\mathrm{St}_a, \mathrm{Ac}_a, \mathsf{P}_a, \mathsf{tr}_a)_{a \in \mathrm{Ag}}, \mathsf{I}, \mathsf{h} \right\rangle$ *whose components are formally defined as follows:*

- St_a *is a finite, non-empty set of local states of agent* $a \in \mathrm{Ag}$*. For each proper agent* $i \in \Sigma$*, we assume that* $\mathrm{St}_i \triangleq \mathrm{St}_i^{\mathrm{p}} \times \mathrm{St}_{\mathrm{Env}}^{\mathsf{vis}_i}$*, where* $\mathrm{St}_i^{\mathrm{p}}$ *is the set of internal states of agent* i *and* $\mathrm{St}_{\mathrm{Env}}^{\mathsf{vis}_i}$ *is an image of the set of the environment states visible to agent* i *via the agent's visibility function* $\mathsf{vis}_i \colon \mathrm{St}_{\mathrm{Env}} \to \mathrm{St}_{\mathrm{Env}}^{\mathsf{vis}_i}$*. For conciseness, set* $\mathrm{St}_{\mathrm{Env}}^{\mathrm{p}} \triangleq \mathrm{St}_{\mathrm{Env}}$*.*

  A tuple $s = (s_1^{\mathrm{p}}, \ldots, s_n^{\mathrm{p}}, s_{\mathrm{Env}}) \in \mathrm{St} \triangleq \mathrm{St}_1^{\mathrm{p}} \times \cdots \times \mathrm{St}_n^{\mathrm{p}} \times \mathrm{St}_{\mathrm{Env}}$ *is called a* global state*. The symbols* $\mathsf{s}_i(s) \triangleq (s_i^{\mathrm{p}}, \mathsf{vis}_i(s_{\mathrm{Env}}))$ *and* $\mathsf{s}_i^{\mathrm{p}}(s) \triangleq s_i^{\mathrm{p}}$ *represent the local and internal state of proper agent* $i \in \Sigma$ *in the global state* s*, respectively. Again, set* $\mathsf{s}_{\mathrm{Env}}(s) \triangleq \mathsf{s}_{\mathrm{Env}}^{\mathrm{p}}(s) \triangleq s_{\mathrm{Env}}$ *for conciseness.*

- Ac_a *is a finite non-empty set of* actions *that agent* $a \in \mathrm{Ag}$ *can perform. By* $\mathrm{Dc} \triangleq \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_n \times \mathrm{Ac}_{\mathrm{Env}}$ *we denote the set of* decisions*, or* joint actions*, of all agents. The symbol* $\mathsf{c}_a(\delta) \in \mathrm{Ac}_a$ *represents the action of agent* $a \in \mathrm{Ag}$ *in the decision* $\delta \in \mathrm{Dc}$*. Furthermore,* $\mathrm{Ac} \triangleq \bigcup_{a \in \mathrm{Ag}} \mathrm{Ac}_a$ *and* $\mathrm{Ac}_{\mathrm{A}} \triangleq \bigcap_{a \in \mathrm{A}} \mathrm{Ac}_a$ *are the sets of* total actions *and* shared actions *of the agents in* $\mathrm{A} \subseteq \mathrm{Ag}$*.*

- $\mathsf{P}_a : \mathrm{St}_a \to 2^{\mathrm{Ac}_a} \setminus \{\emptyset\}$ *is the* protocol *of agent* $a \in \mathrm{Ag}$, i.e.*, a function that associates each local state* $s_a \in \mathrm{St}_a$ *with the non-empty set of actions* $\mathsf{P}_a(s_a)$ *available to the agent* a*. The* global protocol $\mathsf{P} : \mathrm{St} \to 2^{\mathrm{Dc}}$ *is defined as* $\mathsf{P}(s) \triangleq \{\delta \in \mathrm{Dc} \mid \forall a \in \mathrm{Ag} \,.\, \mathsf{c}_a(\delta) \in \mathsf{P}_a(\mathsf{s}_a(s))\}$ *for all global states* $s \in \mathrm{St}$*.*

- $\mathsf{tr}_a : \mathrm{St}_a \times \mathrm{Dc} \to \mathrm{St}_a^{\mathrm{p}}$ *is the* transition function*, or* evolution function*, which maps every local state $s_a \in \mathrm{St}_a$ of an agent $a \in \mathrm{Ag}$ and a decision $\delta \in \mathrm{Dc}$ to a new internal state of the same agent. The resulting new local state of an agent is determined by both her internal state and the visible portion of the amended environment state. The evolution of the whole system is described by a* global transition function $\mathsf{tr}\colon \mathrm{St} \times \mathrm{Dc} \to \mathrm{St}$ *defined as follows:* $\mathsf{tr}(s, \delta) = s'$ *iff, for all agents* $a \in \mathrm{Ag}$*, it holds that* $\mathsf{tr}_a(\mathsf{s}_a(s), \delta) = \mathsf{s}_a^{\mathrm{p}}(s')$*.*

- $\mathrm{I} \subseteq \mathrm{St}$ *is a finite non-empty set of* initial *global states.* W.l.o.g.*, we assume that* St *only contains the global states that are reachable from those in* I *by following the transitions prescribed by the global transition function* tr*.*

- $\mathsf{h}\colon \mathrm{AP} \to 2^{\mathrm{St}}$ *is a* valuation function *that maps each atomic proposition $p \in \mathrm{AP}$ to the set of global states $\mathsf{h}(p)$ in which it is true.*

Modelling agent knowledge in multi-agent systems is a critical aspect in numerous scenarios, since it allows to reason about the decision making process of individual agents, as well as the interactions within a group of agents [28]. The knowledge of (groups of) agents is represented using the four epistemic connectives K, E, D and C already considered for the syntax of SLK. Traditionally, the associated semantics is based on the concept of *epistemic accessibility* [31]. Intuitively, two states are *epistemically accessible* if they are indistinguishable by an agent. This is formalised by means of a binary epistemic accessibility relation on the states of the system. In the case of an interpreted system $\mathcal{I} = \left\langle (\mathrm{St}_a, \mathrm{Ac}_a, \mathsf{P}_a, \mathsf{tr}_a)_{a \in \mathrm{Ag}}, \mathrm{I}, \mathsf{h} \right\rangle$, the individual epistemic accessibility relation $\sim_a \subseteq \mathrm{St} \times \mathrm{St}$ of an agent $a \in \mathrm{Ag}$ is naturally induced by her local states in St_a: two global states $s_1, s_2 \in \mathrm{St}$ are indistinguishable by agent a, in symbols $s_1 \sim_a s_2$, iff $\mathsf{s}_a(s_1) = \mathsf{s}_a(s_2)$, *i.e.*, iff her local states in s_1 and s_2 are the same. This relation provides the meaning for the modality K_a. The remaining modalities E_{A}, D_{A} and C_{A} base their semantics on suitable combinations of the knowledge that the agents in the set $\mathrm{A} \subseteq \mathrm{Ag}$ have:

- The *group* epistemic accessibility relation $\sim_{\mathrm{A}}^{\mathsf{E}}$ is defined as $\sim_{A}^{\mathsf{E}} \triangleq \bigcup_{a \in \mathrm{A}} \sim_a$, *i.e.*, $s_1 \sim_{\mathrm{A}}^{\mathsf{E}} s_2$ iff the local states $\mathsf{s}_a(s_1)$ and $\mathsf{s}_a(s_2)$ of at least one agent $a \in \mathrm{A}$ in the global states $s_1, s_2 \in \mathrm{St}$ are the same.

- The *distributed* epistemic accessibility relation $\sim_{\mathrm{A}}^{\mathsf{D}}$ is defined as $\sim_{A}^{\mathsf{D}} \triangleq \bigcap_{a \in \mathrm{A}} \sim_a$, *i.e.*, $s_1 \sim_{\mathrm{A}}^{\mathsf{D}} s_2$ iff the local states $\mathsf{s}_a(s_1)$ and $\mathsf{s}_a(s_2)$ of all agents $a \in \mathrm{A}$ in the global states $s_1, s_2 \in \mathrm{St}$ are the same.

- The *common* epistemic accessibility relation $\sim_{\mathrm{A}}^{\mathsf{C}}$ is defined as $\sim_{\mathrm{A}}^{\mathsf{C}} \triangleq \left(\sim_{\mathrm{A}}^{\mathsf{E}}\right)^{+}$, *i.e.*, it is the transitive closure of the group accessibility relation.

As standard [28], we say that an agent *knows* a fact iff it is true at all the worlds the agent considers possible, *i.e.*, in all states that the agent cannot distinguish from the current one.

We now recall the standard notions of *strategy*, *profile* and *play*. First observe that an interpreted system $\mathcal{I}$ naturally induces a graph $\mathcal{G}(\mathcal{I}) \triangleq \langle \mathrm{St}, Ed \rangle$, where the edge relation $Ed \triangleq \{(s, \mathsf{tr}(s, \delta)) \in \mathrm{St} \times \mathrm{St} \mid \delta \in \mathrm{Dc}\}$ is obtained by simply deleting all decisions on the transitions. A *path* $\pi \in \mathrm{St}^{\omega}$ in $\mathcal{I}$ is an infinite path in $\mathcal{G}(\mathcal{I})$. By π_i we will denote the i-th state along π. A *strategy* is a function $\sigma \in \mathrm{Str} \triangleq \mathrm{St} \to \mathrm{Ac}$ prescribing which action has to be performed in a certain global state. We say that σ is *coherent w.r.t.* an agent $a \in \mathrm{Ag}$ (a-coherent, for short) if, for each possible global state $s \in \mathrm{St}$, the action that σ prescribes is available to a, *i.e.*, $\sigma(s) \in \mathsf{P}_a(\mathsf{s}_a(s))$. The notion of coherence can be lifted to a set of agents $\mathrm{A} \subseteq \mathrm{Ag}$ (A-coherent, for short) by simply requiring that σ is a-coherent, for every agent $a \in \mathrm{A}$. By $\mathrm{Str}_{\mathrm{A}} \subseteq \mathrm{Str}$ we denote the set of all A-coherent strategies. A strategy σ is also *uniform w.r.t.* the agent a (a-uniform, for short) if, for every pair of global states $s_1, s_2 \in \mathrm{St}$ with $s_1 \sim_a s_2$, it holds that $\sigma(s_1) = \sigma(s_2)$. By $\mathrm{UStr}_{\mathrm{A}} \subseteq \mathrm{Str}_{\mathrm{A}}$ we denote the set of all a-coherent uniform strategies, for the agents $a \in \mathrm{A} \subseteq \mathrm{Ag}$. A strategy *profile* $\xi \in \mathrm{Prf} \subseteq \mathrm{Ag} \to \mathrm{Str}$ specifies for each agent a coherent strategy. Given a profile $\xi \in \mathrm{Prf}$ and an agent $a \in \mathrm{Ag}$, the action $\xi(a)(s) \in \mathsf{P}_a(\mathsf{s}_a(s))$ determines which action a has chosen to perform in a global state $s \in \mathrm{St}$. To identify, instead, the whole decision in s, we apply the flipping function $\widehat{\xi} : \mathrm{St} \to \mathrm{Dc}$ defined as follows: $\widehat{\xi}(s)(a) \triangleq \xi(a)(s)$. A path π is a *play w.r.t.* a profile ξ (ξ*-play*, for short) iff, for all $i \in \mathbb{N}$, there exists a decision $\delta \in \mathsf{P}(\pi_i)$ such that $\delta = \widehat{\xi}(\pi_i)$ and $\pi_{i+1} = \mathsf{tr}(\pi_i, \delta)$, *i.e.*, π_{i+1} is the successor of π_i induced by the agent decision δ prescribed by the profile ξ in the same state. Note that, given a state s, the deterministic structure of the interpreted system ensures that there exists exactly one ξ-play π starting in s. Such a play is called (ξ, s)*-play* and is denoted by $\mathsf{play}(\xi, s)$.

As an example, consider a generalised version of the cake-cutting problem [32], in which n agents share a cake of size d have to slice it in a fairly way. In particular, we would like to formalise a protocol allocating the slices fairly. The idea is a slight generalisation of the procedure proposed in [32] based on a *divide et impera* approach, where the agents take turns to slice the cake, while the environment responds by trying to ensure the cake is divided fairly. More precisely, we assume that, at each even round, the n agents concurrently choose the size of the piece they would each like to receive and, at each odd round, the environment makes the cuts and assigns each piece to a subset of the agents. The environment does this based on agent choices made in the previous turn. However, its action is not necessarily dictated by what agents want, since their willing may represent a non-fair partitioning of the cake. Due to the structure-by-rounds of our approach, the problem of cutting a cake of size d among n agents is divided into several simpler problems, in which pieces of size $d' < d$ have to be split among $n' < n$ agents. The resulting multi-player game terminates once each agent receives a different slice. For simplicity, we assume that we can only cut the cake into pieces of integer size. The formalisation of the corresponding interpreted system $\mathcal{I}$ follows:

- The internal states of the environment have to maintain information about the pieces of cake already sliced, their assignment to the agents, as well as the associated sizes. When it is the environment's turn to move, they also have to record the requests for the sizes of the pieces made by the agents in the previous turn. To distinguish between the turn of the agents and that of the environment, we make use of a flag $\mathtt{A}$ that identifies those states in which the agents need to make a choice. Formally, we have

$$\mathrm{St}_{\mathrm{Env}} = \{((\mathsf{f},\mathsf{g}),\alpha)\in \mathrm{S}\times\mathrm{A} \mid \alpha \neq \mathtt{A} \Rightarrow \forall a \in [1,n]\,.\,\alpha(a) \leq \mathsf{g}(\mathsf{f}(a))\}.$$

where $\mathrm{A} = \{\mathtt{A}\} \cup [1,n] \rightarrow [0,d]$, $\mathrm{S} = \{(\mathsf{f},\mathsf{g}) \in \mathrm{Z} \mid \exists h \in [1,n]\,.\,\mathsf{rng}(\mathsf{f}) = \mathsf{dom}(\mathsf{g}) = [1,h] \wedge \sum_{i\in\mathsf{dom}(\mathsf{g})} \mathsf{g}(i) \leq d\}$ and $\mathrm{Z} = ([1,n] \rightarrow [1,n]) \times ([1,n] \rightharpoonup [0,d])$. Intuitively, the function f assigns to each proper agent $a \in [1,n]$ the $\mathsf{f}(a)$-th piece of cake having size $\mathsf{g}(\mathsf{f}(a))$. Obviously, the sum of all sizes cannot be greater than d, but it can be strictly less than this value, since the environment can remove parts of the cake that cannot be split evenly among the agents. In addition, if it is not the agents' turn, *i.e.*, $\alpha \neq \mathtt{A}$, the function α returns, for each agent $a \in [1,n]$, her own desired size $\alpha(a)$ for the piece of cake to cut from the $\mathsf{f}(a)$-th piece assigned to her. The actions of the environment are all possible splittings and assignments of the cake together with a nothing-to-do operation, *i.e.*, $\mathrm{Ac}_{\mathrm{Env}} = \mathrm{S} \cup \{\bot\}$. To complete the definition of the environment, we first need to define a partial order on the first component of a state, which allows to identify all possible continuations of a given state. Formally, for all $(\mathsf{f},\mathsf{g}), (\mathsf{f}',\mathsf{g}') \in \mathrm{S}$, we write $(\mathsf{f}',\mathsf{g}') \preccurlyeq (\mathsf{f},\mathsf{g})$ iff the following two conditions hold:

– If $\mathsf{f}'(a_1) = \mathsf{f}'(a_2)$, then $\mathsf{f}(a_1) = \mathsf{f}(a_2)$, *i.e.*, if agents $a_1, a_2 \in [1,n]$ share the same piece of cake in f' then they do the same in f.

– $\sum_{a\in\mathsf{f}^{-1}[\{i\}]} \mathsf{g}'(\mathsf{f}'(a)) \leq \mathsf{g}(i)$ for all $i \in \mathsf{dom}(\mathsf{g})$, *i.e.*, the agents sharing the same piece i *w.r.t.* f, have pieces *w.r.t.* f' whose total size in g' does not exceed that of i in g.

Intuitively, $(\mathsf{f}',\mathsf{g}') \preccurlyeq (\mathsf{f},\mathsf{g})$ if f' is a refinement of f, *i.e.*, f' is a finer partition than f, and g' is coherent with the size of the pieces of the previous cut. At this point, we can formalise the protocol for Env as follows: $\mathsf{P}_{\mathrm{Env}}(((\mathsf{f},\mathsf{g}),\mathtt{A})) = \{\bot\}$ and $\mathsf{P}_{\mathrm{Env}}(((\mathsf{f},\mathsf{g}),\alpha)) = \{(\mathsf{f}',\mathsf{g}') \in \mathrm{S} \mid (\mathsf{f}',\mathsf{g}') \preccurlyeq (\mathsf{f},\mathsf{g})\}$ if $\alpha \neq \mathtt{A}$. This means that, when it is the environment's turn, it can choose one of the possible finer continuations of a given state. Finally, for the transition function we have $\mathsf{tr}(((\mathsf{f},\mathsf{g}),\mathtt{A}),\delta) = ((\mathsf{f},\mathsf{g}),\alpha)$, where $\alpha(a) = \mathsf{c}_a(\delta)$, for all $a \in [1,n]$, and $\mathsf{tr}(((\mathsf{f},\mathsf{g}),\alpha),\delta) = (\mathsf{c}_{\mathrm{Env}}(\delta),\mathtt{A})$ when $\alpha \neq \mathtt{A}$. Informally, when it is the agents' turn, the environment just registers the agents' choices in its internal state. When it is its turn instead, it modifies its state by executing its own action $\mathsf{c}_{\mathrm{Env}}(\delta)$.

• All information about the cake required by an agent $a \in [1,n]$ is already contained in the environment state and visible to to all agents, *i.e.*, $\mathsf{vis}_a(s_{\mathrm{Env}}) = s_{\mathrm{Env}}$ for all environment states $s_{\mathrm{Env}} \in \mathrm{St}_{\mathrm{Env}}$. Hence, the agent does not need any internal state, *i.e.*, $\mathrm{St}_a^{\mathrm{p}} = \{\mathtt{I}\}$. The actions are simply the dimensions of the possible pieces of cake together with a nothing-to-do operation, *i.e.*, $\mathrm{Ac}_a = [1, d] \cup \{\perp\}$. In the environment's turn, *i.e.*, when $\alpha \neq \mathtt{A}$, the only possible executable action is $\perp$, *i.e.*, $\mathsf{P}_a((\mathtt{I}, ((\mathsf{f}, \mathsf{g}), \alpha))) = \{\perp\}$. Otherwise, we have $\mathsf{P}_a((\mathtt{I}, ((\mathsf{f}, \mathsf{g}), \alpha))) = [0, \mathsf{g}(\mathsf{f}(a))]$, *i.e.*, a can choose a size from 0 to the current dimension of the $\mathsf{f}(a)$-th piece. Because of the singleton internal state $\mathtt{I}$, we have $\mathsf{tr}_a((\mathtt{I}, s_{\mathrm{Env}}), \delta) = \mathtt{I}$ for all environment states $s_{\mathrm{Env}} \in \mathrm{St}_{\mathrm{Env}}$ and decisions $\delta \in \mathrm{Dc}$.

- The set of initial global states is the singleton $\mathrm{I} = \{(\mathtt{I}, \ldots, \mathtt{I}, ((\mathsf{f}, \mathsf{g}), \mathtt{A}))\}$, where the entire cake is assigned to all agents, *i.e.*, $\mathsf{f}(a) = 1$ for all $a \in [1, n]$, and its size is $\mathsf{g}(1) = d$.
- The set of atomic propositions, which represent all possible sizes of a piece of cake an agent can receive, is $\mathrm{AP} = [1, n] \times [1, d] \cup \{\#\}$. For example, $\langle a, i \rangle$ is true iff agent $a \in [1, n]$ receives a piece of cake of size $i \in [1, d]$, *i.e.*, $\mathsf{g}(\mathsf{f}(a)) = i$. In addition, the symbol # is used to label the terminal states in which each agent obtains a different piece. The labelling is defined as follows: $\mathsf{h}(\langle a, i \rangle) = \{(\mathtt{I}, \ldots, \mathtt{I}, ((\mathsf{f}, \mathsf{g}), \alpha)) \in \mathrm{St} \mid i = \mathsf{g}(\mathsf{f}(a))\}$ for all $\langle a, i \rangle \in \mathrm{AP}, \alpha \in \mathrm{A}$ and $\mathsf{h}(\#) = \{(\mathtt{I}, \ldots, \mathtt{I}, ((\mathsf{f}, \mathsf{g}), \alpha)) \in \mathrm{St} \mid \forall a_1, a_2 \in [1, n] \,.\, a_1 \neq a_2 \Rightarrow \mathsf{f}(a_1) \neq \mathsf{f}(a_2)\}$ for all $\alpha \in \mathrm{A}$.

2.3. Semantics

We can now continue with the formalisation of SLK semantics. Similarly to first-order logic, the interpretation of a formula makes use of an assignment function which generally associates placeholders with some elements of the quantification domain. For SLK, as in SL, an *assignment* is a partial function $\chi \in \mathrm{Asg} \triangleq (\mathrm{Vr} \cup \mathrm{Ag}) \rightharpoonup \mathrm{Str}$ mapping variables and agents to strategies such that $\chi(a)$ is a-coherent for every agent $a \in \mathsf{dom}(\chi) \cap \mathrm{Ag}$. By $\mathrm{Asg}_{\mathrm{P}}$ we denote the set of assignments defined over the placeholders in $\mathrm{P} \subseteq \mathrm{Ag} \cup \mathrm{Vr}$. An assignment χ is *complete* iff it is defined on all agents, *i.e.*, $\mathrm{Ag} \subseteq \mathsf{dom}(\chi)$. In this case, it directly identifies the profile $\chi_{\restriction \mathrm{Ag}}$ given by the restriction of χ to Ag. In addition, $\chi[e \mapsto \sigma]$, with $e \in \mathrm{Vr} \cup \mathrm{Ag}$ and $\sigma \in \mathrm{Str}$, is the assignment defined on $\mathsf{dom}(\chi[e \mapsto \sigma]) = \mathsf{dom}(\chi) \cup \{e\}$ which differs from χ only in the fact that e is associated with σ. Formally, $\chi[e \mapsto \sigma](e) \triangleq \sigma$ and $\chi[e \mapsto \sigma](e') \triangleq \chi(e')$ for all $e' \in \mathsf{dom}(\chi) \setminus \{e\}$. Obviously, in case e is an agent, the previous operation is well-defined only if σ is e-coherent.

Given a complete assignment $\chi \in \mathrm{Asg}_{\mathrm{P}}$ over a set of placeholders $\mathrm{P} \subseteq \mathrm{Ag} \cup \mathrm{Vr}$ and a state $s \in \mathrm{St}$, we define the i-th translation of (χ, s) *w.r.t.* an index $i \in \mathbb{N}$ as the pair $(\chi, s)^i \triangleq (\chi, s') \in \mathrm{Asg}_{\mathrm{P}} \times \mathrm{St}$, where $s' = \pi_i$ is the i-th state along the corresponding play $\pi = \mathsf{play}(\chi_{\restriction \mathrm{Ag}}, s)$.

We can now formally introduce the semantics of SLK.

Definition 5 (Semantics)**.** *Given an interpreted system $\mathcal{I}$, for all* SLK *formulas $\varphi \in$* SLK*, states $s \in \mathrm{St}$ and assignments $\chi \in \mathrm{Asg}$ such that $\mathsf{free}(\varphi) \subseteq \mathsf{dom}(\chi)$ and $\chi(x)$ is $\mathsf{shr}(\varphi, x)$-coherent for all $x \in \mathsf{dom}(\chi) \cap \mathrm{Vr}$, the satisfaction relation $\mathcal{I}, (\chi, s) \models \varphi$ is inductively defined as follows:*

1. *$\mathcal{I}, (\chi, s) \models p$ if $s \in \mathsf{h}(p)$, with $p \in \mathrm{AP}$.*
2. *Truth values and Boolean operators are interpreted as usual.*
3. *For all variables $x \in \mathrm{Vr}$, we have that:*
   (a) *$\mathcal{I}, (\chi, s) \models \langle\!\langle x \rangle\!\rangle \varphi$ if there exists a $\mathsf{shr}(\varphi, x)$-coherent uniform strategy $\sigma \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}$ such that $\mathcal{I}, (\chi[x \mapsto \sigma], s) \models \varphi$;*
   (b) *$\mathcal{I}, (\chi, s) \models [\![x]\!] \varphi$ if, for all $\mathsf{shr}(\varphi, x)$-coherent uniform strategies $\sigma \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}$, it holds that $\mathcal{I}, (\chi[x \mapsto \sigma], s) \models \varphi$.*
4. *For an agent $a \in \mathrm{Ag}$ and a variable $x \in \mathrm{Vr}$, we have that: $\mathcal{I}, (\chi, s) \models (a, x)\varphi$ if $\mathcal{I}, (\chi[a \mapsto \chi(x)], s) \models \varphi$.*
5. *For an agent $a \in \mathrm{Ag}$, we have that: $\mathcal{I}, (\chi, s) \models \mathsf{K}_a \varphi$ if, for all states $s' \in \mathrm{St}$ with $s \sim_a s'$, it holds that $\mathcal{I}, (\emptyset, s') \models \varphi$.*
6. *For a set of agents $\mathrm{A} \subseteq \mathrm{Ag}$, we have that:*
   (a) *$\mathcal{I}, (\chi, s) \models \mathsf{E}_{\mathrm{A}} \varphi$ if, for all states $s' \in \mathrm{St}$ with $s \sim^{\mathsf{E}}_{\mathrm{A}} s'$, it holds that $\mathcal{I}, (\emptyset, s') \models \varphi$;*
   (b) *$\mathcal{I}, (\chi, s) \models \mathsf{D}_{\mathrm{A}} \varphi$ if, for all states $s' \in \mathrm{St}$ with $s \sim^{\mathsf{D}}_{\mathrm{A}} s'$, it holds that $\mathcal{I}, (\emptyset, s') \models \varphi$;*
   (c) *$\mathcal{I}, (\chi, s) \models \mathsf{C}_{\mathrm{A}} \varphi$ if, for all states $s' \in \mathrm{St}$ with $s \sim^{\mathsf{C}}_{\mathrm{A}} s'$, it holds that $\mathcal{I}, (\emptyset, s') \models \varphi$.*
7. *Finally, if the assignment χ is also complete, we have that:*
   (a) *$\mathcal{I}, (\chi, s) \models \mathsf{X}\, \varphi$ if $\mathcal{I}, (\chi, s)^1 \models \varphi$;*
   (b) *$\mathcal{I}, (\chi, s) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ if there is an index $i \in \mathbb{N}$ such that $\mathcal{I}, (\chi, s)^i \models \varphi_2$ and, for all indices $j \in \mathbb{N}$ with $j < i$, it holds that $\mathcal{I}, (\chi, s)^j \models \varphi_1$;*
   (c) *$\mathcal{I}, (\chi, s) \models \varphi_1 \,\mathsf{R}\, \varphi_2$ if, for all indices $i \in \mathbb{N}$, it holds that $\mathcal{I}, (\chi, s)^i \models \varphi_2$ or there is an index $j \in \mathbb{N}$ with $j < i$ such that $\mathcal{I}, (\chi, s)^j \models \varphi_1$.*

As the satisfaction of a sentence φ does not depend on the assignments, we omit them and write $\mathcal{I}, s \models \varphi$ for an arbitrary state $s \in \mathrm{St}$, and $\mathcal{I} \models \varphi$ when $\mathcal{I}, s \models \varphi$ holds for all initial states $s \in \mathrm{I}$ in $\mathcal{I}$.

Slk semantics differs from Sl's one in the following two aspects, both of which are consequences of using incomplete information: *(i)* it is defined on uniform memoryless strategies; *(ii)* it supports epistemic operators. Note that, while epistemic modalities increase the expressive power of Slk *w.r.t.* Sl, the imperfect recall feature limits the agents' behaviours.

As it is clear from the interpretation of the epistemic operators and the associated syntactic restriction on free placeholders (*i.e.*, all the epistemic operators can only predicate on sentences), we assume that agents are not aware of the current strategy assignment. In fact, in as far as the epistemic accessibility relation is concerned, they are not aware of the strategy chosen at a state. This assumption is implemented by the empty assignments in the inductive definitions of the epistemic cases.

Also observe that Slk strictly subsumes all logics in the Atl* hierarchy with incomplete information and imperfect recall imposed on all agents, including Ctlk and Atlk. Moreover, it allows us to express properties that cannot be formalised in any of the previous logics, including the original Sl. For example, given an interpreted system with agents $\mathrm{Ag} = \{a, b\}$, the Slk sentence

$$\mathsf{E}_{\{a,b\}}\, [\![e]\!] [\![x]\!] (\mathrm{Env}, e)(a, x)(b, x)\, [\mathsf{G}\,\mathsf{F}\, p \wedge \mathsf{G}\,\mathsf{F}\, \neg p]$$

expresses that a and b both know that if they use the same strategy, the atomic proposition p will be infinitely often true and infinitely often false.

Lastly, note that in the formalisation above all agents in the system are assumed to adhere to memoryless strategies. This is in contrast with memoryless ATL formalisms whereby agents the coalition in the formula are allowed to follow memoryfull strategies. We leave combinations of memoryless and memoryfull strategies for further work.

We now illustrate SLK's expressive power to express *Nash equilibria* by continuing our analysis of the cake cutting problem. Recall that we use atomic propositions $\langle a, i\rangle \in [1, n] \times [1, d]$ to indicate that agent a gets a piece of cake of size i. In addition, the symbol # is used to denote the terminal states in which a final division of the cake is reached. The existence of a protocol for the cake-cutting problem can be expressed by the following SLK specification

$$\varphi = \langle\langle x\rangle\rangle\,(\varphi_{\mathrm{F}} \wedge \varphi_{\mathrm{S}})$$

where:

- $\varphi_{\mathrm{F}} = [\![y_1]\!]\cdots[\![y_n]\!](\psi_{\mathrm{NE}} \rightarrow \psi_{\mathrm{E}})$ ensures that the protocol x is *fair*, *i.e.*, all possible Nash equilibria $(y_1, \ldots, y_n)$ of the agents guarantee equality of splitting;

- $\varphi_{\mathrm{S}} = \langle\langle y_1\rangle\rangle\cdots\langle\langle y_n\rangle\rangle\psi_{\mathrm{NE}}$ ensures that the protocol has a *solution*, *i.e.*, there is at least one Nash equilibrium;

- $\psi_{\mathrm{NE}} = \flat \bigwedge_{a=1}^{n}\left(\bigwedge_{i=1}^{d}\left(\langle\langle z\rangle\rangle\,(a,z)p_a^i\right) \rightarrow \left(\bigvee_{j=i}^{d} p_a^j\right)\right)$ ensures that, if agent a has a strategy z that allows her to obtain a piece of cake of size i once the strategies of the other agents are fixed, she is already able to obtain a slice of size $j \geq i$ by means of her original strategy y_a; in other words, $(y_1, \ldots, y_n)$ is a *Nash equilibrium*;

- $\psi_{\mathrm{E}} = \flat \bigwedge_{a=1}^{n} p_a^{\lfloor d/n\rfloor}$ ensures that agent a is able to obtain a piece of size $\lfloor d/n\rfloor$;

- $\flat = (\mathrm{Env}, x)(1, y_1)\cdots(n, y_n)$ and $p_a^i = \mathsf{F}(\# \wedge \langle a, i\rangle)$ are auxiliary abbreviations.

Observe that we can perform the synthesis of cake-cutting protocols satisfying specific desired properties, by simply conjoining the corresponding SLK formalisation with the formulas φ_{F} and φ_{S} in φ.

Note, that, as above, when reasoning about Nash equlibria all agents in the system are assumed to follow memoryless strategies.

2.4. Model Checking

We now introduce the model-checking problem for SLK. In particular, we show that its imperfect-recall semantics allows us to obtain a decision procedure whose complexity is, in comparison to that of SL with complete information, considerably simpler *w.r.t.* the length of specification, but notably harder *w.r.t.* the size of the system. Indeed, the perfect-recall semantics of Sl [23] induces the related model checking problem to be polynomial in data complexity and non-elementary in formula complexity [26]. On the contrary, we prove that Slk model checking is PSpace-complete *w.r.t.* both input dimensions.

Definition 6 (Model Checking)**.** *Given an interpreted system $\mathcal{I}$, a state $s \in \mathrm{St}$, an assignment χ and an* Slk *sentence $\varphi \in$* Slk*, the* model-checking problem *involves determining whether $\mathcal{I}, (\chi, s) \models \varphi$ holds.*

The original algorithmic procedure for Sl follows an automata-theoretic approach [33], reducing the decision problem for the logic to the emptiness problem of a tree automaton, where the Ltl properties are verified by means of an embedded word automaton obtained by a variation of the classic Vardi-Wolper construction [34]. The non-elementariness *w.r.t.* the length of the Sl specification is due to the alternation of the memoryful strategy quantifiers that requires alternating projection operations on the automaton, each of which induces an exponential blow-up. This construction is, however, independent of the underlying model, a key feature that is crucial for the exhibited data complexity.

Unfortunately, due to the undecidability result proved for Atl with memoryful strategies under incomplete information [27], we cannot hope to use the same approach of standard Sl when agents only have partial information about the global states. In fact, Slk avoids the undecidability of the model-checking problem by precisely exploiting its imperfect recall semantics. Effectively, we are removing agent memory and forcing them to choose their actions purely based on their current local states. In this way, the domain of strategies over which the quantifications have to range is necessarily finite, due to the finiteness of the underlying model. This fact allows us to apply a completely different approach, where one simply iterates over all possible memoryless strategies. Once the strategies are assigned to all corresponding variables in the Slk formula, we can then project them onto the interpreted structure obtaining a labelled graph, where the Ltl property is verified recursively on the structure of the formula, by mimicking the semantics definition. It is quite immediate to observe that this procedure only requires a polynomial amount of space *w.r.t.* both the size of the model and the length of the Sl specification. However, it is important to note that this approach is not independent of the model under verification. Even more, there is no way to avoid this unless PTime = PSpace. We indeed prove that the Slk model-checking problem is hard for the latter complexity class.

Since the model checking of Atl* with imperfect recall is known to be PSpace-complete [35], one may be tempted to immediately conclude that the same holds for Slk. However, there is an important difference between Atl* and Slk semantics: the former assigns memoryless strategies only to the existentially quantified agents, whereas the latter assigns memoryless strategies to all agents. Informally, an Atl* expression $\langle\!\langle \mathrm{A} \rangle\!\rangle \psi$ means that there exist memoryless strategies for agents in A such that, no matter how the other agents behave, the property ψ holds. The corresponding Slk formula $\{\langle\!\langle x_a \rangle\!\rangle (a, x_a)\}_{a \in \mathrm{A}} \{[\![y_a]\!]$

$(a, y_a)\}_{a \in \text{Ag} \setminus \text{A}} \varphi$ has a slightly different meaning: there exist memoryless strategies for agents in A such that, for all memoryless strategies of the remaining agents, ψ holds. Intuitively, Slk restricts the universally quantified agents more than Atl* does[1]. Therefore, Slk cannot inherit the hardness result from Atl*, which is, in its turn, inherited from that of Ltl. Instead, we show a direct reduction from the satisfiability problem of quantified Boolean formulas (QBF, for short), which was proved to be complete for PSpace in [36]. This reduction is loosely inspired by the one used to prove a similar result for Substructure Temporal Logic over finite models [37].

Before concluding this section with a proof of the model-checking result for Slk, we stress that although memoryless strategies are less powerful than the memoryful ones, they are more compact and, therefore, easier to handle. These features are desirable in several scenarios. Consider, for example, the situation in which we want to synthesise the correct behaviour of a simple hardware device with a fixed amount of memory. In this case, we can model each possible memory settings of the device as a local state and allow arbitrary transitions between them. Informally, we let the agent representing the device decide what she wants to remember. Thus, it is appropriate to treat this agent as being memoryless, since her real memory is already encoded in her local state space. We then model check the property describing the desired behaviour. If we succeed, we can extract from the procedure a memoryless strategy as a witness, which ensures that the fixed-size memory is sufficient for the agent to achieve the required goal.

Theorem 1 (Model Checking). *The model-checking problem for* Slk *is* PSpace-complete w.r.t. *both the size of the model and the length of the specification.*

Proof. The proof is divided into two parts. In the first one we give the PSpace decision procedure; in the second we provide the PSpace hardness proof by means of a reduction from QBF satisfiability.

To verify that a given Slk formula φ is satisfied over an interpreted system $\mathcal{I}$ at a state $s \in \text{St}$ under a memoryless assignments $\chi \in \text{Asg}$, we make use of the PSpace computable recursive function $\texttt{Verify}_{\mathcal{I}} : \text{Sl} \times \text{Asg} \times \text{St} \to \{\bot, \top\}$ defined as follows, for which it holds that $\mathcal{I}, (\chi, s) \models \varphi$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi, s) = \top$:

- $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi, s) \triangleq \varphi$, with $\varphi \in \{\bot, \top\}$.
- $\texttt{Verify}_{\mathcal{I}}(p, \chi, s) \triangleq \top$ iff $s \in \mathsf{h}(p)$, with $p \in \text{AP}$.
- $\texttt{Verify}_{\mathcal{I}}(\neg\varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi, s) = \bot$.
- $\texttt{Verify}_{\mathcal{I}}(\varphi_1 \wedge \varphi_2, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi_1, \chi, s) = \top$ and $\texttt{Verify}_{\mathcal{I}}(\varphi_2, \chi, s) = \top$.

[1]Note that this subtle difference in the semantics has no effect on the relationship between Sl and Atl* with perfect recall since enforcing memoryful strategies on agents does not constrain their behaviour in any way.

- $\texttt{Verify}_{\mathcal{I}}(\varphi_1 \vee \varphi_2, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi_1, \chi, s) = \top$ or $\texttt{Verify}_{\mathcal{I}}(\varphi_2, \chi, s) = \top$.
- $\texttt{Verify}_{\mathcal{I}}(\langle\langle x\rangle\rangle\varphi, \chi, s) \triangleq \top$ iff there exists a memoryless $\mathsf{shr}(\varphi, x)$-coherent uniform strategy $\sigma \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}$ such that $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi[x \mapsto \sigma], s) = \top$.
- $\texttt{Verify}_{\mathcal{I}}([\![x]\!]\varphi, \chi, s) \triangleq \top$ iff for all memoryless $\mathsf{shr}(\varphi, x)$-coherent uniform strategies $\sigma \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}$ it holds that $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi[x \mapsto \sigma], s) = \top$.
- $\texttt{Verify}_{\mathcal{I}}((a, x)\varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi[a \mapsto \chi(x)], s) = \top$.
- $\texttt{Verify}_{\mathcal{I}}(\mathsf{K}_a\, \varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \varnothing, s') = \top$, for all states $s' \in \mathrm{St}$ with $s \sim_a s'$.
- $\texttt{Verify}_{\mathcal{I}}(\mathsf{E}_{\mathrm{A}}\, \varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \varnothing, s') = \top$, for all states $s' \in \mathrm{St}$ with $s \sim^{\mathsf{E}}_{\mathrm{A}} s'$.
- $\texttt{Verify}_{\mathcal{I}}(\mathsf{D}_{\mathrm{A}}\, \varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \varnothing, s') = \top$, for all states $s' \in \mathrm{St}$ with $s \sim^{\mathsf{D}}_{\mathrm{A}} s'$.
- $\texttt{Verify}_{\mathcal{I}}(\mathsf{C}_{\mathrm{A}}\, \varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \varnothing, s') = \top$, for all states $s' \in \mathrm{St}$ with $s \sim^{\mathsf{C}}_{\mathrm{A}} s'$.
- $\texttt{Verify}_{\mathcal{I}}(\mathsf{X}\, \varphi, \chi, s) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi, \chi, s') = \top$, where $s' = \mathsf{tr}(s, \widehat{\chi_{\restriction \mathrm{Ag}}}(s))$ is the successor of the state s following the decision $\widehat{\chi_{\restriction \mathrm{Ag}}}(s)$ obtained by evaluating the flipping of the profile $\chi_{\restriction \mathrm{Ag}}$ on s itself.
- To compute $\texttt{Verify}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s)$, we use the auxiliary bounded-recursive function $\texttt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s, \mathrm{C})$, where $\mathrm{C} \subseteq \mathrm{St}$ is the set of states already visited during the verification of the property $\varphi_1 \,\mathsf{U}\, \varphi_2$. We set $\texttt{Verify}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s) \triangleq \texttt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s, \emptyset)$ and define the latter as follows, by exploiting the classic one-step unfolding property of the until operator:
  - if $s \in \mathrm{C}$, then $\texttt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s, \mathrm{C}) \triangleq \bot$;
  - if $s \notin \mathrm{C}$ and $\texttt{Verify}_{\mathcal{I}}(\varphi_2, \chi, s) = \top$, then $\texttt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s, \mathrm{C}) \triangleq \top$;
  - if $s \notin \mathrm{C}$ and $\texttt{Verify}_{\mathcal{I}}(\varphi_2, \chi, s) = \bot$, then $\texttt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s, \mathrm{C}) \triangleq \top$ iff $\texttt{Verify}_{\mathcal{I}}(\varphi_1, \chi, s) = \top$ and $\texttt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s', \mathrm{C} \cup \{s\}) = \top$, where $s' = \mathsf{tr}(s, \widehat{\chi_{\restriction \mathrm{Ag}}}(s))$ is the successor of the state s following the decision $\widehat{\chi_{\restriction \mathrm{Ag}}}(s)$.
- Similarly to the previous case, to compute $\texttt{Verify}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s)$, we use the auxiliary bounded-recursive function $\texttt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s, \mathrm{C})$, where $\mathrm{C} \subseteq \mathrm{St}$ is the set of states already visited during the verification of the property $\varphi_1 \,\mathsf{R}\, \varphi_2$. We set $\texttt{Verify}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s) \triangleq \texttt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s, \emptyset)$ and define the latter as follows, by exploiting the classic one-step unfolding property of the release operator:

- if $s \in \mathrm{C}$, then $\mathtt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s, \mathrm{C}) \triangleq \top$;
- if $s \notin \mathrm{C}$ and $\mathtt{Verify}_{\mathcal{I}}(\varphi_2, \chi, s) = \bot$, then $\mathtt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s, \mathrm{C}) \triangleq \bot$;
- if $s \notin \mathrm{C}$ and $\mathtt{Verify}_{\mathcal{I}}(\varphi_2, \chi, s) = \top$, then $\mathtt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s, \mathrm{C}) \triangleq \bot$ iff $\mathtt{Verify}_{\mathcal{I}}(\varphi_1, \chi, s) = \bot$ and $\mathtt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s', \mathrm{C} \cup \{s\}) = \bot$, where $s' = \mathsf{tr}(s, \widehat{\chi_{\restriction \mathrm{Ag}}}(s))$ is the successor of the state s following the decision $\widehat{\chi_{\restriction \mathrm{Ag}}}(s)$.

The correctness of the construction can be shown by induction on the syntactic structure of the Slk formulas, by observing that a formula $\varphi_1 \,\mathsf{U}\, \varphi_2$ is not satisfied on $\mathcal{I}$ starting at s under χ, when the same state is found twice on the path induced by χ without proving that φ_2 holds somewhere on it before φ_1 is falsified. A similar reasoning can be used to establish the case for $\varphi_1 \,\mathsf{R}\, \varphi_2$ as well. As far as the complexity is concerned, a single $\mathtt{Verify}_{\mathcal{I}}(\varphi, \chi, s)$ function call (excluding recursion) requires $|\varphi| + |\mathrm{St}| \cdot \lceil \log_2 |\mathrm{Ac}| \rceil \cdot |\mathsf{free}(\varphi)| + \lceil \log_2 |\mathrm{St}| \rceil$ space to maintain its parameters. Similar reasoning can be done for $\mathtt{Verify}^{\mathsf{U}}_{\mathcal{I}}(\varphi_1 \,\mathsf{U}\, \varphi_2, \chi, s, \mathrm{C})$ and $\mathtt{Verify}^{\mathsf{R}}_{\mathcal{I}}(\varphi_1 \,\mathsf{R}\, \varphi_2, \chi, s, \mathrm{C})$ function calls, which require $|\varphi| + |\mathrm{St}| \cdot \lceil \log_2 |\mathrm{Ac}| \rceil \cdot |\mathsf{free}(\varphi)| + \lceil \log_2 |\mathrm{St}| \rceil + |\mathrm{St}|$ space. Furthermore, in each recursive call (of any of the three checking functions), either φ is decomposed or C is augmented. Hence, the recursion depth cannot be more than $\mathrm{O}(|\varphi| \cdot |\mathrm{St}|)$ at any point in time. It follows that the model checking procedure requires only $\mathrm{O}(|\varphi|^2 \cdot |\mathrm{St}|^2 \cdot \lceil \log_2 |\mathrm{Ac}| \rceil)$ space.

We now turn to the PSpace lower bound, which is proved by means of a direct reduction from QBF satisfiability. Let $\varphi = \wp\psi$ be a QBF formula over the Boolean variables in $\mathrm{X} = \{x_1, \ldots, x_k\}$, where $\wp$ is the quantification prefix of the form $\mathsf{Qn}_1 x_1 \cdots \mathsf{Qn}_k x_k$, with $\mathsf{Qn}_a \in \{\exists, \forall\}$ for $a \in [1, k]$, and the matrix $\psi = \bigwedge_{i=1}^{m} \mathrm{D}_i$ is a Boolean formula in conjunctive normal form over the variables in X, where each clause D_i can be seen as a subset of the literals in $\mathrm{AP} = \{x, \overline{x} \mid x \in \mathrm{X}\}$. We transform φ into a suitable Slk formula $\widetilde{\varphi}$ over the set of atomic propositions AP, where, for each $x \in \mathrm{X}$, we need two atomic propositions, one for the positive literal x and one for the negative literal $\neg x$. In the formula $\widetilde{\varphi}$, the k propositional quantifications are replaced by corresponding strategy quantifications for k different proper agents. These quantifications are applied to a suitable encoding $\widetilde{\psi}$ of the matrix ψ, where the verification of the conjunction of clauses is assigned to a distinguished agent, the $k+1$-th, while that of each clause to the agent embodying the environment. Formally, we have $\widetilde{\varphi} = \mathsf{Qn}'_1 x_1 \cdots \mathsf{Qn}'_k x_k [\![x_{k+1}]\!] \langle\!\langle x_{\mathrm{Env}} \rangle\!\rangle (1, x_1) \cdots (k, x_k)(k+1, x_{k+1})(\mathrm{Env}, x_{\mathrm{Env}})\widetilde{\psi}$, where the Ltl formula $\widetilde{\psi}$ is defined later in the proof and $\mathsf{Qn}'_a x_a = \langle\!\langle x_a \rangle\!\rangle$, if $\mathsf{Qn}_a = \exists$, and $\mathsf{Qn}'_a x_a = [\![x_a]\!]$ otherwise. The formula $\widetilde{\varphi}$ is then checked against the interpreted system $\mathcal{I}$ described in the following paragraph, which ensures that φ is satisfiable iff $\mathcal{I} \models \widetilde{\varphi}$. The correctness of this construction can be shown by induction on the number k of quantifications occurring in φ.

The idea behind the construction of $\mathcal{I}$ is to have a global state for each clause D_i, each variable in X and each of its two possible Boolean valuations. Every proper agent a from 1 to k determines the valuation of her associated variable x_a. Agent $k+1$ determines which clause D_i we have to focus on for the verification task. Finally, the environment agent Env tries to satisfy the chosen clause, by selecting the variable x_a with the appropriate valuation. In Figure 1, we exemplify the reduction for the QBF formula $\varphi = \forall p \exists q \exists r \, . \, (p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee \neg p)$, by showing the internal graph structure of the environment. The formal description of $\mathcal{I}$ follows:

- The system consists of the $k+1$ proper agents in $\Sigma = \{1, \ldots, k+1\}$ together with the environment Env.

- The set of internal states of the environment is $\mathrm{St}_{\mathrm{Env}} = \{\mathtt{I}\} \cup \{\mathrm{D}_1, \ldots, \mathrm{D}_m\} \cup \{x_1, \ldots, x_k\}$, where $\mathtt{I}$ represents the initial state. Since the task assigned to Env is to verify a given clause D_i by selecting one of its literals, we have $\mathrm{Ac}_{\mathrm{Env}} = \{\bot\} \cup \{1, \ldots, k\}$ as set of actions, where $\bot$ represents the idle action. Accordingly, Env employs the following protocol: $\mathsf{P}_{\mathrm{Env}}(s_{\mathrm{Env}}) = \{a \in \{1, \ldots, k\} \mid \{x_a, \overline{x}_a\} \cap \mathrm{D}_i \neq \emptyset\}$, if $s_{\mathrm{Env}} = \mathrm{D}_i$ for some $i \in \{1, \ldots, m\}$, and $\mathsf{P}_{\mathrm{Env}}(s_{\mathrm{Env}}) = \{\bot\}$ otherwise. In this way, we are sure that Env can only select one of the indices a of the variables occurring in the chosen clause D_i. Finally, the transition function is defined as follows: $\mathsf{tr}_{\mathrm{Env}}(\mathtt{I}, \delta) = \mathrm{D}_{\mathsf{c}_{k+1}(\delta)}$, $\mathsf{tr}_{\mathrm{Env}}(\mathrm{D}_i, \delta) = x_{\mathsf{c}_{\mathrm{Env}}(\delta)}$ and $\mathsf{tr}_{\mathrm{Env}}(x_j, \delta) = x_j$, for $j \in [1, k]$. Intuitively, when Env is at the initial state, it transits to the clause of index $\mathsf{c}_{k+1}(\delta)$ chosen by the agent $k+1$. If the state is a clause, the environment can decide to move to the variable of index $\mathsf{c}_{\mathrm{Env}}(\delta)$. Once a state representing a variable is reached, it remains there forever.

- Agent $k+1$ only needs to decide which clause will be verified. Therefore, she has only one internal state, $\mathrm{St}^{\mathrm{p}}_{k+1} = \{\mathtt{I}\}$, sees the whole environment state, $\mathsf{vis}_{k+1}(s_{\mathrm{Env}}) = s_{\mathrm{Env}}$, her transition function is trivial, $\mathsf{tr}_{k+1}(s_{k+1}, \delta) = \mathtt{I}$, her actions are $\mathrm{Ac}_{k+1} = \{\bot\} \cup \{1, \ldots, m\}$ and the protocol is set as follows: $\mathsf{P}_{k+1}(s_{k+1}) = \{1, \ldots, m\}$, if $s_{k+1} = (\mathtt{I}, \mathtt{I})$, and $\mathsf{P}_{k+1}(s_{k+1}) = \{\bot\}$ otherwise.

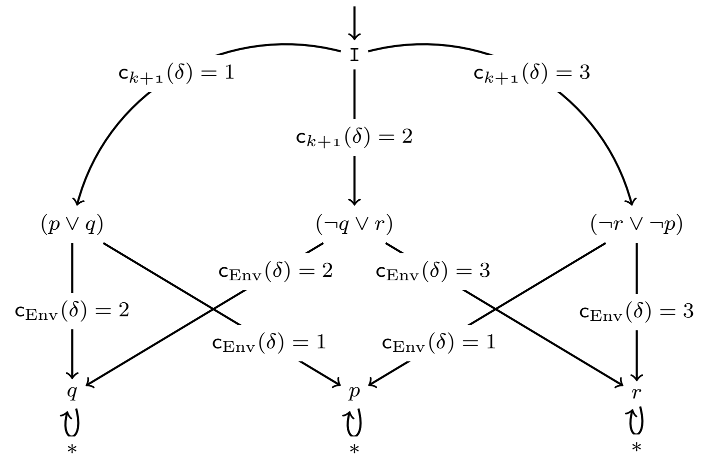
- Each agent $a \in \{1, \ldots, k\}$ has a set of internal states equal to $\mathrm{St}^{\mathrm{p}}_a = \{\mathtt{I}, 0, 1\}$, where $\mathtt{I}$ represents the initial state, while 0 and 1 are the Boolean valuations to be assigned to the variable x_a. The agent sees the whole environment state, *i.e.*, $\mathsf{vis}_a(s_{\mathrm{Env}}) = s_{\mathrm{Env}}$ for all $s_{\mathrm{Env}} \in \mathrm{St}_{\mathrm{Env}}$. The set of actions is $\mathrm{Ac}_a = \{\bot, 0, 1\}$, where the idle action $\bot$ is used by a when she does not need to set the value of the variable. The protocol and the transition function are defined as follows: $\mathsf{P}_a(s_a) = \{0, 1\}$ and $\mathsf{tr}_a(s_a, \delta) = \mathsf{c}_a(\delta)$, if $s_a = (\mathtt{I}, x_a)$, and $\mathsf{P}_a(s_a) = \{\bot\}$ and $\mathsf{tr}_a(s_a, \delta) = s_a$, if $s_a \neq (\mathtt{I}, x_a)$. Intuitively, agent a can choose the value for her own variable x_a only when she is in her initial state and the environment is in its internal state associated with the variable x_a.

- There is only one global initial state represented by the $(k+2)$-tuple whose components are all equal to $\mathtt{I}$, $\mathrm{I} = \{(\mathtt{I}, \ldots, \mathtt{I})\}$.

- Finally, the valuation function for the atomic propositions is set as follows:

$$\begin{aligned}\mathsf{h}(x_a) = \{s \in \mathrm{St} \mid [\exists i \in \{1,\ldots,m\}\,.\,\mathsf{s}_{\mathrm{Env}}(s) = \mathrm{D}_i \wedge x_a \in \mathrm{D}_i] \vee \\ [\mathsf{s}_{\mathrm{Env}}(s) = x_a \wedge \mathsf{s}^{\mathrm{p}}_a(s) = 1]\} \quad \text{for all } x_a \in \mathrm{AP} \cap \mathrm{X} \\ \mathsf{h}(\overline{x}_a) = \{s \in \mathrm{St} \mid [\exists i \in \{1,\ldots,m\}\,.\,\mathsf{s}_{\mathrm{Env}}(s) = \mathrm{D}_i \wedge \overline{x}_a \in \mathrm{D}_i] \vee \\ [\mathsf{s}_{\mathrm{Env}}(s) = x_a \wedge \mathsf{s}^{\mathrm{p}}_a(s) = 0]\} \quad \text{for all } \overline{x}_a \in \mathrm{AP} \setminus \mathrm{X}\end{aligned}$$

  Intuitively, we label with x_a (*resp.* $\overline{x}_a$) all global states corresponding to clauses containing this proposition. We also label those states corresponding to the valuation of the variable set to 1 (*resp.* 0).



I
$\mathsf{c}_{k+1}(\delta) = 1$
$\mathsf{c}_{k+1}(\delta) = 2$
$\mathsf{c}_{k+1}(\delta) = 3$
$(p \vee q)$
$(\neg q \vee r)$
$(\neg r \vee \neg p)$
$\mathsf{c}_{\mathrm{Env}}(\delta) = 2$
$\mathsf{c}_{\mathrm{Env}}(\delta) = 2$
$\mathsf{c}_{\mathrm{Env}}(\delta) = 3$
$\mathsf{c}_{\mathrm{Env}}(\delta) = 3$
$\mathsf{c}_{\mathrm{Env}}(\delta) = 1$
$\mathsf{c}_{\mathrm{Env}}(\delta) = 1$
q
p
r
*
*
*

Figure 1: Structure underlying the environment.

Now note that the number of reachable global states in this interpreted system is $1+m+3k$. Indeed, at the initial global state $(\mathtt{I},\ldots,\mathtt{I},\mathtt{I})$, the only agent that can execute a non-idle action is agent $(k+1)$, choosing one of the m possible clauses D_i. In the resulting global state $(\mathtt{I},\ldots,\mathtt{I},\mathrm{D}_i)$, the environment is the only agent capable of changing its internal state, thereby reaching one of the k variables, say x_j. The global state resulting from this action is $(\mathtt{I},\ldots,\mathtt{I},x_j)$. In this situation, only agent j can change her own state, by choosing one of the two possibles actions $\alpha \in \{0,1\}$, which represents the valuation for the variable x_j. Consequently, the system reaches the global state $(\mathtt{I},\ldots,\alpha,\ldots,\mathtt{I},x_j)$, where α is the j-th component of the tuple. The system will then remain in this state forever. Observe that, since only one agent at a time can perform a non-idle action, the number of possible transitions in the system, *i.e.*, the number of decisions declared in the global protocol, is linear in the size of the input QBF formula as well, precisely $m + \sum_{i=1}^{m} |\mathrm{D}_i| + 2k$.

To conclude the reduction, we have to define the LTL formula $\widetilde{\psi}$. The idea here is to ensure that, when the environment chooses a given variable to satisfy a clause D_i selected by agent $k+1$, the polarity of the variable in the clause and that of its valuation agree. In other words, for each proposition x_a, we have to ensure that either the proposition itself, or its dual $\overline{x}_a$ occurs along the play identified by the strategies. Therefore, to ensure the above, we take $\widetilde{\psi} = \bigwedge_{a=1}^{k} \neg\left((\mathsf{F}\, x_a) \wedge (\mathsf{F}\, \overline{x}_a)\right)$. $\square$

3. Labelling Algorithm for Model Checking

In this section we describe our novel labelling algorithm for Slk which solves the model-checking problem described in Definition 6.

3.1. Algorithm

The model checking algorithm $\mathtt{Check}_{\mathcal{I}}$ for Slk, which calculates the set of global states in which a given formula is true, is an extension of the existing ones for temporal logics. It differs from those in two ways:

1. It has an extra input parameter which represents the *binding* of agents to variables. When model checking a formula, we start with an empty binding and augment it whenever an agent binding operator $(a, x)\varphi$ is encountered.
2. Unlike the original algorithm, which merely returns the set of states in which a given formula holds, the present algorithm returns a set of *extended states*. Intuitively, an extended state is a pair of *(i)* a global state and *(ii)* a variable assignment (mapping variables to strategies) subject to which the formula holds in that state.

We will now define the aforementioned concepts of a binding and variable assignment more formally. For simplicity, we will fix Vr to always be the set of variables quantified in the Slk formula we are considering (*e.g.*, if the formula to be checked is $\varphi = \langle\!\langle x \rangle\!\rangle [\![y]\!] (a, x)(b, y) \mathsf{X}\, p$, then we set $\mathrm{Vr} = \{x, y\}$). This will allow us to define variable assignments as total functions, which will make the theory and proofs much simpler.

Definition 7 (Bindings)**.** *Let $\mathcal{I}$ be an interpreted system. Then a* binding *is a partial function $b \colon \mathrm{Ag} \rightharpoonup \mathrm{Vr}$ which maps agents in its domain to variables.* $\mathrm{Bnd} \triangleq \mathrm{Ag} \rightharpoonup \mathrm{Vr}$ *denotes the set of all bindings.*

Definition 8 (Variable Assignments)**.** *Let $\mathcal{I}$ be an interpreted system. Then a* variable assignment *is a function $v \colon \mathrm{Vr} \to \mathrm{UStr}$ which maps variables in its domain to uniform shared memoryless strategies.* $\mathrm{VAsg} \triangleq \mathrm{Vr} \to \mathrm{UStr}$ *denotes the set of all variable assignments.*

Note that variable assignments are also assignments, *i.e.*, $\mathrm{VAsg} \subseteq \mathrm{Asg}$. As an example of an extended state, consider the tuple $(s_1, \{(x, \sigma_x), (y, \sigma_y)\})$, representing the global state s_1 in which the agents bound to variables x and y act according to the strategies σ_x and σ_y, respectively. We formalise this below.

Definition 9 (Extended States)**.** *Let $\mathcal{I}$ be an interpreted system, $s \in \mathrm{St}$ a global state and $v \in \mathrm{VAsg}$ a variable assignment. Then an* extended state *is a pair $\langle s, v\rangle \in \mathrm{St} \times \mathrm{VAsg}$. $\mathrm{Ext} \triangleq \mathrm{St} \times \mathrm{VAsg}$ denotes the set of all extended states.*

Intuitively, an extended state $\langle s, v\rangle \in \mathrm{Ext}$ *guarantees* a formula $\varphi \in \textsc{Slk}$ iff all assignments which agree with v make the formula true in the state s.

Definition 10 (Guarantees)**.** *Let $\mathcal{I}$ be an interpreted system, $\langle s, v\rangle \in \mathrm{Ext}$ an extended state, $b \in \mathrm{Bnd}$ a binding and $\varphi \in \textsc{Slk}$ an* Slk *formula with $\mathsf{free}(\varphi) \cap \mathrm{Ag} \subseteq \mathsf{dom}(b)$. We say that $\langle s, v\rangle$* guarantees *φ in $\mathcal{I}$ under b iff for the assignment $\chi_v = v \cup \{(a, v(b(a))) \mid a \in \mathsf{dom}(b)\} \in \mathrm{Asg}$, we have $\mathcal{I}, (\chi_v, s) \models_{\textsc{Slk}} \varphi$.*

Now that we have covered the basic structures, we can define the concepts of negation and predecessors of extended states. These will be necessary for the model checking algorithm presented at the end of this subsection. Let us start with negation. Assume that we have calculated the set of extended states E which guarantee the formula φ under a binding b. We want to find the set E' of extended states which guarantee the formula $\neg\varphi$. Intuitively, E' should contain all extended states that somehow disagree with E. E' is calculated as follows:

$$\mathrm{E}' = \mathrm{Ext} \setminus \mathrm{E}$$

We will now prove that our claim is correct.

Lemma 1. *Let $\mathcal{I}$ be an interpreted system, $b \in \mathrm{Bnd}$ a binding and $\varphi \in \textsc{Slk}$ an* Slk *formula with $\mathsf{free}(\varphi) \cap \mathrm{Ag} \subseteq \mathsf{dom}(b)$. Let $\mathrm{E} \subseteq \mathrm{Ext}$ be the set of all extended states which guarantee φ in $\mathcal{I}$ under b. Then $\mathrm{Ext} \setminus \mathrm{E}$ is the set of all extended states which guarantee $\neg\varphi$ in $\mathcal{I}$ under b.*

Proof. Let $\mathrm{E}' \subseteq \mathrm{Ext}$ be the set of all extended states which guarantee $\neg\varphi$ in $\mathcal{I}$ under b. We show that $\mathrm{E}' = \mathrm{Ext} \setminus \mathrm{E}$:

$\Rightarrow$: Take an arbitrary extended state $\langle s, v\rangle \in \mathrm{E}'$. Since $\langle s, v\rangle$ guarantees $\neg\varphi$, we have $\mathcal{I}, (\chi_v, s) \models_{\textsc{Slk}} \neg\varphi$. By Slk semantics (Definition 5), we have $\mathcal{I}, (\chi_v, s) \not\models_{\textsc{Slk}} \varphi$. Thus, $\langle s, v\rangle \notin \mathrm{E}$, so we have $\langle s, v\rangle \in \mathrm{Ext} \setminus \mathrm{E}$.

$\Leftarrow$: Take an arbitrary extended state $\langle s, v\rangle \in \mathrm{Ext} \setminus \mathrm{E}$. Since $\langle s, v\rangle$ does not guarantee φ (otherwise, we would have $\langle s, v\rangle \in \mathrm{E}$), we have $\mathcal{I}, (\chi_v, s) \not\models_{\textsc{Slk}} \varphi$. By Slk semantics (Definition 5), $\mathcal{I}, (\chi_v, s) \models_{\textsc{Slk}} \neg\varphi$. Thus, $\langle s, v\rangle \in \mathrm{Ext}$ guarantees $\neg\varphi$, so we have $\langle s, v\rangle \in \mathrm{E}'$. □

Having covered negation, it remains to define how to calculate the set of previous extended states, *i.e.*, given a set of extended states $\mathrm{E} \subseteq \mathrm{Ext}$ which guarantee φ under a binding b, determine the set of extended states $\mathrm{E}' \subseteq \mathrm{Ext}$ which guarantee $\mathsf{X}\,\varphi$ under the same binding.

We first define the transition relation on global states implied by a binding and a variable assignment. Intuitively, there is a transition from state $s_1 \in \mathrm{St}$ to a state $s_2 \in \mathrm{St}$ (given the binding and variable assignment) if there exists a decision between them such that each agent acts according to the strategy assigned to the variable she is bound to.

Definition 11 (Implied Transition Relation). *Let $\mathcal{I}$ be an interpreted system, $s_1, s_2 \in \mathrm{St}$ two global states, $b \in \mathrm{Bnd}$ a binding and $v \in \mathrm{VAsg}$ a variable assignment. Then the* transition relation $\rightarrow_v^b \subseteq \mathrm{St} \times \mathrm{St}$ implied *by b and v is defined by $s_1 \rightarrow_v^b s_2$, iff $\mathsf{dom}(b) = \mathrm{Ag}$ and there exists a decision $\delta \in \mathrm{Dc}$ such that $\mathsf{tr}(s_1, \delta) = s_2$ and, for all agents $a \in \mathrm{Ag}$, it holds that $\mathsf{c}_a(\delta) = v(b(a))(s_1)$.*

We are now ready to explain how the set of previous extended states is calculated. Intuitively, given an extended state $\langle s, v\rangle \in \mathrm{Ext}$, the previous extended states are pairs $\langle s', v\rangle$ where s is the successor of s' when all agents act according to their strategies in v.

Definition 12 (Previous Extended States). *Let $\mathcal{I}$ be an interpreted system, $\mathrm{E} \subseteq \mathrm{Ext}$ a set of extended states and $b \in \mathrm{Bnd}$ a binding such that $\mathsf{dom}(b) = \mathrm{Ag}$. Then the function $\mathsf{pre}\colon 2^{\mathrm{Ext}} \times \mathrm{Bnd} \to 2^{\mathrm{Ext}}$ returning the set of* previous extended states *is defined as $\mathsf{pre}(\mathrm{E}, b) \triangleq \left\{\langle s, v\rangle \in \mathrm{Ext} \;\middle|\; \exists s' \in \mathrm{St}.\, \langle s', v\rangle \in \mathrm{E} \wedge s \rightarrow_v^b s'\right\}$.*

Again, we will show that the function pre is correct.

Lemma 2. *Let $\mathcal{I}$ be an interpreted system, $b \in \mathrm{Bnd}$ a binding with $\mathsf{dom}(b) = \mathrm{Ag}$ and $\varphi \in \mathrm{SLK}$ an* SLK *formula. Let $\mathrm{E} \subseteq \mathrm{Ext}$ be the set of all extended states which guarantee φ in $\mathcal{I}$ under b. Then $\mathsf{pre}(\mathrm{E}) \subseteq \mathrm{Ext}$ is the set of all extended states which guarantee $\mathsf{X}\,\varphi$ in $\mathcal{I}$ under b.*

Proof. Let $\mathrm{E}' \subseteq \mathrm{Ext}$ be the set of all extended states which guarantee $\mathsf{X}\,\varphi$ in $\mathcal{I}$ under b. We show that $\mathsf{pre}(\mathrm{E}) = \mathrm{E}'$:

$\Rightarrow$: Take an arbitrary extended state $\langle s, v\rangle \in \mathsf{pre}(\mathrm{E}, b)$. By construction, there is a global state $s' \in \mathrm{St}$ such that $\langle s', v\rangle \in \mathrm{E}$ and $s \rightarrow_v^b s'$. Since $\langle s', v\rangle \in \mathrm{E}$ guarantees φ, we have $\mathcal{I}, (\chi_v, s') \models_{\mathrm{SLK}} \varphi$. The implied transition relation implies that there exists a decision $\delta \in \mathrm{Dc}$ such that $s' = \mathsf{tr}(s, \delta)$ where $\mathsf{c}_a(\delta) = v(b(a))(s)$ for all agents $a \in \mathrm{Ag}$. This can be also rewritten as $s' = \mathsf{tr}(s, \langle \chi_v(a)(s) : a \in \mathrm{Ag}\rangle)$ because $\chi_v(a) = v(b(a))$ for all agents $a \in \mathsf{dom}(b) = \mathrm{Ag}$ (see Definition 10).

We want to show that $\langle s, v\rangle$ guarantees $\mathsf{X}\,\varphi$, *i.e.*, $\mathcal{I}, (\chi_v, s) \models_{\mathrm{SLK}} \mathsf{X}\,\varphi$. This is the case iff $\mathcal{I}, (\chi_v, \pi_1) \models_{\mathrm{SLK}} \varphi$ where $\pi = \mathsf{play}(\chi_v, s)$ (see Definition 5). From the definition of a play (see Subsection 2.1), we obtain $\pi_1 = \mathsf{tr}(s, \langle \chi_v(a)(s) : a \in \mathrm{Ag}\rangle)$, so $\pi_1 = s'$. Since we have already shown that $\mathcal{I}, (\chi_v, s') \models_{\mathrm{SLK}} \varphi$, we have $\mathcal{I}, (\chi_v, s) \models_{\mathrm{SLK}} \mathsf{X}\,\varphi$, so $\langle s, v\rangle \in \mathrm{E}'$ as required.

$\Leftarrow$: Take an arbitrary extended state $\langle s, v\rangle \in \mathrm{E}'$. Thus, we have $\mathcal{I}, (\chi_v, s) \models_{\mathrm{SLK}} \mathsf{X}\,\varphi$. By Definition 5, this means that $\mathcal{I}, (\chi_v, \pi_1) \models_{\mathrm{SLK}} \varphi$ where $\pi = \mathsf{play}(\chi_v, s)$. From the definition of a play (see Subsection 2.1, we obtain $\pi_1 = \mathsf{tr}(s, \langle \chi_v(a)(s) : a \in \mathrm{Ag}\rangle)$. This can be equivalently written as $\pi_1 = \mathsf{tr}(s, \delta)$ for some $\delta \in \mathrm{Dc}$ where $\mathsf{c}_a(\delta) = \chi_v(a)(s) = v(b(a))(s)$ for all agents $a \in \mathrm{Ag}$.

As $\mathsf{dom}(b) = \mathrm{Ag}$ by assumption, we have $s \rightarrow_v^b \pi_1$. Moreover, since $\mathcal{I}, (\chi_v, \pi_1) \models_{\mathrm{SLK}} \varphi$, we have $\langle \pi_1, v\rangle \in \mathrm{E}$ (because it guarantees φ). Therefore, we have $\langle s, v\rangle \in \mathsf{pre}(\mathrm{E}, b)$ as required. □

Finally, we have all the ingredients to define the model checking algorithm $\mathtt{Check}_{\mathcal{I}}(\cdot,\cdot)$ for SLK.

Definition 13 (SLK Model Checking Algorithm)**.** *Let $\mathcal{I}$ be an interpreted system, $\varphi \in$ SLK an SLK formula and $b \in \mathrm{Bnd}$ a binding, such that $\mathsf{free}(\varphi) \cap \mathrm{Ag} \subseteq \mathsf{dom}(b)$. Then the model checking function $\mathtt{Check}_{\mathcal{I}} : \mathrm{SLK} \times \mathrm{Bnd} \to 2^{\mathrm{Ext}}$ is inductively defined as follows:*

1. $\mathtt{Check}_{\mathcal{I}}(\top, b) \triangleq \mathrm{Ext}$.
2. $\mathtt{Check}_{\mathcal{I}}(p, b) \triangleq \{\langle s, v\rangle \mid s \in \mathsf{h}(p)\}$, *with* $p \in \mathrm{AP}$.
3. *For all formulas $\varphi, \varphi_1, \varphi_2 \in$ SLK, it is defined as:*
   (a) $\mathtt{Check}_{\mathcal{I}}(\neg\varphi, b) \triangleq \mathrm{Ext} \setminus \mathtt{Check}_{\mathcal{I}}(\varphi, b)$;
   (b) $\mathtt{Check}_{\mathcal{I}}(\varphi_1 \wedge \varphi_2, b) \triangleq \mathtt{Check}_{\mathcal{I}}(\varphi_1, b) \cap \mathtt{Check}_{\mathcal{I}}(\varphi_2, b)$;
   (c) $\mathtt{Check}_{\mathcal{I}}(\varphi_1 \vee \varphi_2, b) \triangleq \mathtt{Check}_{\mathcal{I}}(\varphi_1, b) \cup \mathtt{Check}_{\mathcal{I}}(\varphi_2, b)$.
4. *For an agent $a \in \mathrm{Ag}$, a variable $x \in \mathrm{Vr}$ and a formula $\varphi \in$ SLK,* $\mathtt{Check}_{\mathcal{I}}((a, x)\varphi, b) \triangleq \mathtt{Check}_{\mathcal{I}}(\varphi, b[a \mapsto x])$.
5. *For a variable $x \in \mathrm{Vr}$ and an SLK formula $\varphi \in$ SLK, it is defined as:*
   (a) $\mathtt{Check}_{\mathcal{I}}(\langle\langle x\rangle\rangle\varphi, b) \triangleq \big\{\langle s, v\rangle \in \mathrm{Ext} \bigm| \exists f \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}. \langle s, v[x \mapsto f]\rangle \in \mathtt{Check}_{\mathcal{I}}(\varphi, b)\big\}$;
   (b) $\mathtt{Check}_{\mathcal{I}}([\![x]\!]\varphi, b) \triangleq \big\{\langle s, v\rangle \in \mathrm{Ext} \bigm| \forall f \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}. \langle s, v[x \mapsto f]\rangle \in \mathtt{Check}_{\mathcal{I}}(\varphi, b)\big\}$.
6. *For all formulas $\varphi, \varphi_1, \varphi_2 \in$ SLK, it is defined as:*
   (a) $\mathtt{Check}_{\mathcal{I}}(\mathsf{X}\,\varphi, b) \triangleq \mathsf{pre}(\mathtt{Check}_{\mathcal{I}}(\varphi, b), b)$;
   (b) $\mathtt{Check}_{\mathcal{I}}(\mathsf{F}\,\varphi, b) \triangleq \mathtt{Check}_{\mathcal{I}}(\top\,\mathsf{U}\,\varphi, b)$;
   (c) $\mathtt{Check}_{\mathcal{I}}(\mathsf{G}\,\varphi, b) \triangleq \mathtt{Check}_{\mathcal{I}}(\neg\mathsf{F}\,\neg\varphi, b)$;
   (d) $\mathtt{Check}_{\mathcal{I}}(\varphi_1\,\mathsf{U}\,\varphi_2, b) \triangleq \mathrm{lfp}_{\mathrm{X}}\big[\mathtt{Check}_{\mathcal{I}}(\varphi_2, b) \cup (\mathtt{Check}_{\mathcal{I}}(\varphi_1, b) \cap \mathsf{pre}(\mathrm{X}, b))\big]$ *where $\mathrm{lfp}_{\mathrm{X}} f$ is the least fixed point of function f;*
   (e) $\mathtt{Check}_{\mathcal{I}}(\varphi_1\,\mathsf{R}\,\varphi_2, b) \triangleq \mathtt{Check}_{\mathcal{I}}(\neg\left((\neg\varphi_1)\,\mathsf{U}\,(\neg\varphi_2)\right), b)$.
7. *For an agent $a \in \mathrm{Ag}$, a set of agents $\mathrm{A} \subseteq \mathrm{Ag}$ and a formula $\varphi \in$ SLK, it is defined as:*
   (a) $\mathtt{Check}_{\mathcal{I}}(\mathsf{K}_a\,\varphi, b) \triangleq \mathsf{acc}(\varphi, \sim_a)$;
   (b) $\mathtt{Check}_{\mathcal{I}}(\mathsf{E}_{\mathrm{A}}\,\varphi, b) \triangleq \mathsf{acc}(\varphi, \sim^{\mathsf{E}}_{\mathrm{A}})$;
   (c) $\mathtt{Check}_{\mathcal{I}}(\mathsf{D}_{\mathrm{A}}\,\varphi, b) \triangleq \mathsf{acc}(\varphi, \sim^{\mathsf{D}}_{\mathrm{A}})$;
   (d) $\mathtt{Check}_{\mathcal{I}}(\mathsf{C}_{\mathrm{A}}\,\varphi, b) \triangleq \mathsf{acc}(\varphi, \sim^{\mathsf{C}}_{\mathrm{A}})$;
   where $\mathsf{acc}(\varphi, R) \triangleq \{\langle s, v\rangle \in \mathrm{Ext} \mid \forall\, \langle s', v'\rangle \in \mathtt{Check}_{\mathcal{I}}(\neg\varphi, \emptyset). \neg R(s', s)\}$.

The correctness of the algorithm is asserted in the following theorem.

Theorem 2. *Let $\mathcal{I}$ be an interpreted system and $\varphi \in$ SLK an SLK sentence. Then the set of all states at which φ holds is such that:*

$$\{s \in \mathrm{St} \mid \mathcal{I}, s \models_{\mathrm{SLK}} \varphi\} = \{s \in \mathrm{St} \mid \exists v \in \mathrm{VAsg}. \langle s, v\rangle \in \mathtt{Check}_{\mathcal{I}}(\varphi, \emptyset)\}\,.$$

Proof (Sketch). We prove by induction that for an arbitrary interpreted system $\mathcal{I}$, SLK formula $\varphi \in$ SLK and binding $b \in \mathrm{Bnd}$ such that $\mathsf{free}(\varphi) \subseteq \mathsf{dom}(b)$, $\mathtt{Check}_{\mathcal{I}}(\varphi, b)$ is the set of all extended states that guarantee φ in $\mathcal{I}$ under b.

Lemma 1 and Lemma 2 report the proofs for negation and the temporal operators, respectively.

Since the topmost formula φ is a sentence, it will either hold, or not hold in each state (regardless of the variable assignment). Therefore, we existentially quantify over variable assignments. □

Observe that all cases of the SLK model checking algorithm are well-defined because the sets UStr, Vr, VAsg, Ag, Dc, Bnd, St and, consequently, Ext are finite. An efficient symbolic implementation of the algorithm using BDDs is presented in Subsection 3.3.

3.2. Strategy Synthesis

One of the features of SLK is that it may encode *non-behavioural strategies* [38], where an agent's action in a particular scenario may depend on actions in counterfactual scenarios. Consequently, SLK strategies are difficult to synthesise. To see why this is the case, consider the SLK sentence $\varphi = [\![x]\!][\![y]\!]\langle\!\langle z\rangle\!\rangle\psi$. Assume that φ holds at a particular state $s \in \mathrm{St}$ in an interpreted system $\mathcal{I}$, *i.e.*, $\mathcal{I}, s \models_{\mathrm{SLK}} \varphi$. We would now like to synthesise a uniform shared strategy $f_z : \mathrm{St} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,z)}$ for the variable z depending on the uniform shared strategies $f_x : \mathrm{St} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,x)}$ and $f_y : \mathrm{St} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,y)}$ for the variables x and y, respectively. If SLK strategies were behavioural, there *would* exist a mapping m_1 (or an *elementary dependence map* as in [38]) from the next actions of f_x and f_y in s to the next action of f_z in s:

$$m_1 : \underbrace{\mathrm{St} \to \left(\mathrm{Ac}_{\mathsf{shr}(\varphi,x)} \times \mathrm{Ac}_{\mathsf{shr}(\varphi,y)} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,z)}\right)}_{(s, f_x(s), f_y(s)) \mapsto f_z(s)}$$

There are at most $\left|\mathrm{Ac}_{\mathsf{shr}(\varphi,x)}\right| \times \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,y)}\right|$ possible inputs to m_1 to determine $f_z(s)$ as it depends only on $f_x(s)$ and $f_y(s)$. Unfortunately, such a mapping does not exist in general because SLK strategies are non-behavioural. Instead, a more general mapping m_2 (also referred to as *dependence map* [38]) from strategies f_x and f_y to the strategy f_z must be considered:

$$m_2 : \underbrace{\left(\mathrm{St} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,x)}\right) \times \left(\mathrm{St} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,y)}\right) \to \left(\mathrm{St} \to \mathrm{Ac}_{\mathsf{shr}(\varphi,z)}\right)}_{(f_x, f_y, s) \mapsto f_z(s)}$$

Informally, determining $f_z(s)$ requires the same amount of information as constructing the whole strategy f_z. In order to synthesise the action $f_z(s)$ or the strategy f_z, we possibly need to know the complete strategies f_x and f_y. More importantly, the maximum number of *entries* in the mappings m_1 and m_2 are:

$$\begin{aligned} m_1 &: |\mathrm{St}| \times \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,x)}\right| \quad\;\; \times \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,y)}\right| \\ m_2 &: |\mathrm{St}| \times \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,x)}\right|^{|\mathrm{St}|} \times \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,y)}\right|^{|\mathrm{St}|} \end{aligned}$$

Assume that the interpreted system $\mathcal{I}$ is relatively small; e.g., assume $|\mathrm{St}| = 10$ global states and $\left|\mathrm{Ac}_{\mathsf{shr}(\varphi,x)}\right| = \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,y)}\right| = \left|\mathrm{Ac}_{\mathsf{shr}(\varphi,z)}\right| = 10$ actions. While the mapping m_1 for *behavioural* strategies would require at most 1000 entries of the form $(s, f_x(s), f_y(s)) \mapsto f_z(s)$, the mapping m_2 for *non-behavioural* strategies might have up to 10^{21} entries of the form $(f_x, f_y, s) \mapsto f_z(s)$. Furthermore, if we encode each output of f_z using only $\lceil \log_2 |\mathrm{Ac}_{\mathsf{shr}(\varphi,z)}| \rceil = 4$ bits and store the whole mapping in a large array, m_1 will use at most 500 *bytes* while m_2 might need up to 434 *exabytes*.

While SLK strategy synthesis is infeasible in general, it can be performed efficiently on certain types of formulas. We will now explain the concepts of *witness* and *counterexample strategies* and describe how these can be synthesised using the model checking algorithm discussed in Subsection 3.1. Let $\mathcal{I}$ be an interpreted system, $s \in \mathrm{St}$ a global state and $\varphi_w = \langle\!\langle x \rangle\!\rangle \psi_w$ and $\varphi_c = [\![y]\!]\psi_c$ two SLK sentences. Furthermore, assume that φ_w holds at s while φ_c does not, *i.e.*, $\mathcal{I}, (\emptyset, s) \models_{\mathrm{SLK}} \varphi_w$ and $\mathcal{I}, (\emptyset, s) \not\models_{\mathrm{SLK}} \varphi_c$. By SLK semantics (see Definition 5), there is a memoryless uniform shared strategy f_w for x which makes ψ_w true at s. Conversely, there must be a memoryless shared strategy f_c for y which makes ψ_c false at s. f_w and f_c are referred to as a *witness* and a *counterexample strategy*, respectively. Intuitively, the strategy f_w is a "witness" to φ_w being true at s while f_c is a "counterexample" for φ_c at s. A slightly more general form of the two concepts is provided in the following definition.

Definition 14 (Witness and Counterexample Strategies)**.** *Let $\mathcal{I}$ be an interpreted system, $s \in \mathrm{St}$ a global state and $\varphi_w = \langle\!\langle x_0 \rangle\!\rangle \dots \langle\!\langle x_{m-1} \rangle\!\rangle \psi_w$ and $\varphi_c = [\![y_0]\!] \dots [\![y_{n-1}]\!]\psi_c$ two* SLK *sentences. Then:*

- *Memoryless uniform shared strategies $f_{w0}, \dots, f_{w(m-1)}$ are* witness strategies *for φ_w at s iff* (i) *$f_{wi} \in \mathrm{UStr}_{\mathsf{shr}(\psi_w, x_i)}$ for all $0 \leq i < m$ and* (ii) *$\mathcal{I}, (\chi_w, s) \models_{\mathrm{SLK}} \psi_w$ where $\chi_w = \{(x_i, f_{wi}) \mid 0 \leq i < m\}$.*

- *Memoryless uniform shared strategies $f_{c0}, \dots, f_{c(n-1)}$ are* counterexample strategies *for φ_c at s iff* (i) *$f_{ci} \in \mathrm{UStr}_{\mathsf{shr}(\psi_c, y_i)}$ for all $0 \leq i < n$ and* (ii) *$\mathcal{I}, (\chi_c, s) \not\models_{\mathrm{SLK}} \psi_c$ where $\chi_c = \{(y_i, f_{ci}) \mid 0 \leq i < n\}$.*

The two concepts are *duals* of each other in the sense that if f is a witness strategy for $\langle\!\langle x \rangle\!\rangle \psi$ at s, then it is a counterexample strategy for $[\![x]\!]\neg\psi$ at s (and vice versa).

Lemma 3. *Let $\mathcal{I}$ be an interpreted system, $s \in \mathrm{St}$ a global state, ψ an agent-closed* SLK *formula with $\mathsf{free}(\psi) = \{x_0, \dots, x_{n-1}\}$ and $f_0, \dots, f_{n-1}$ memoryless uniform shared strategies such that $f_i \in \mathrm{UStr}_{\mathsf{shr}(x_i, \psi)}$ for $0 \leq i < n$. Then $f_0, \dots, f_{n-1}$ are witness strategies of $\varphi_w = \langle\!\langle x_0 \rangle\!\rangle \dots \langle\!\langle x_{n-1} \rangle\!\rangle \psi$ at s iff $f_0, \dots, f_{n-1}$ are counterexample strategies of $\varphi_c = [\![x_0]\!] \dots [\![x_{n-1}]\!]\neg\psi$ at s.*

Proof. We prove both directions of the equivalence separately:

$\Rightarrow$: Assume that $f_0, \dots, f_{n-1}$ are witness strategies of φ_w at s. By Definition 14, we have $f_i \in \mathrm{UStr}_{\mathsf{shr}(\psi, x_i)}$ for all $0 \leq i < n$ and $\mathcal{I}, (\chi, s) \models \psi$ where $\chi = \{(x_i, f_i) \mid 0 \leq i < n\}$. By Definition 3, we have $\mathsf{shr}(\neg\varphi, x) = \mathsf{shr}(\varphi, x)$ for all SLK formulas $\varphi \in$ SLK and variables $x \in \mathrm{Vr}$ so $f_i \in \mathrm{UStr}_{\mathsf{shr}(\neg\psi, x_i)}$

for all $0 \leq i < n$. Since $\mathcal{I}, (\chi, s) \models_{\text{SLK}} \psi$, $\mathcal{I}, (\chi, s) \not\models_{\text{SLK}} \neg\psi$ holds by SLK semantics (Definition 5). Hence, by Definition 14, $f_0, \ldots, f_{n-1}$ are counterexample strategies for φ_c at s.

$\Leftarrow$: Assume that $f_0, \ldots, f_{n-1}$ are counterexample strategies of φ_c at s. By Definition 14, we have $f_i \in \text{UStr}_{\mathsf{shr}(\neg\psi, x_i)}$ for all $0 \leq i < n$ and $\mathcal{I}, (\chi, s) \not\models_{\text{SLK}} \neg\psi$ where $\chi = \{(x_i, f_i) \mid 0 \leq i < n\}$. Again, we have $f_i \in \text{UStr}_{\mathsf{shr}(\psi, x_i)}$ for all $0 \leq i < n$ by Definition 3. Since $\mathcal{I}, (\chi, s) \not\models_{\text{SLK}} \neg\psi$, $\mathcal{I}, (\chi, s) \models_{\text{SLK}} \psi$ holds by SLK semantics (Definition 5). Hence, by Definition 14, $f_0, \ldots, f_{n-1}$ are witness strategies of φ_w at s. □

This duality will allow us to focus on witness strategies and their synthesis only. Before describing how witness strategies can be retrieved, we need to show that they always exist when an SLK formula holds.

Lemma 4. *Let $\mathcal{I}$ be an interpreted system, ψ an agent-closed* SLK *formula such that $\mathsf{free}(\psi) = \{x_0, \ldots, x_{n-1}\}$ and $\varphi = \langle\!\langle x_0 \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle \psi$ an* SLK *sentence. Then the following holds: $\mathcal{I}, (\emptyset, s) \models_{\text{SLK}} \varphi$ iff there exist witness strategies $f_0, \ldots, f_{n-1}$ for φ at s.*

Proof. We prove both directions of the equivalence separately:

$\Rightarrow$: Assume that $\mathcal{I}, (\emptyset, s) \models_{\text{SLK}} \varphi$. By SLK semantics (Definition 5), there exist strategies $f_0, \ldots, f_{n-1}$ such that $f_i \in \text{UStr}_{\mathsf{shr}(\langle\!\langle x_{i+1} \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle \psi, x_i)}$ for all $0 \leq i < n$ and $\mathcal{I}, (\chi, s) \models_{\text{SLK}} \psi$ where $\chi = \{(x_i, f_i) \mid 0 \leq i < n\}$. By Definition 3, we have $\mathsf{shr}(\langle\!\langle x_{i+1} \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle \psi, x_i) = \mathsf{shr}(\psi, x_i)$ for all $0 \leq i < n$. Hence $f_0, \ldots, f_{n-1}$ satisfy both conditions for being witness strategies of φ at s (see Definition 14).

$\Leftarrow$: Assume that there exist witness strategies $f_0, \ldots, f_{n-1}$ for φ at s. By Definition 14, we have $f_i \in \text{UStr}_{\mathsf{shr}(\psi, x_i)}$ for all $0 \leq i < n$ and $\mathcal{I}, (\chi, s) \models_{\text{SLK}} \psi$ where $\chi = \{(x_i, f_i) \mid 0. \leq i < n\}$. By Definition 3, we have $\mathsf{shr}(\psi, x_i) = \mathsf{shr}(\langle\!\langle x_{i+1} \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle \psi, x_i)$. Hence, for all $0 \leq i < n$, we get $f_i \in \text{UStr}_{\mathsf{shr}(\langle\!\langle x_{i+1} \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle \psi, x_i)}$. Therefore, $\mathcal{I}, (\emptyset, s) \models_{\text{SLK}} \varphi$ by SLK semantics (Definition 5). □

It remains to explain how witness strategies can be synthesised using the SLK model checking algorithm we introduced in Subsection 3.1. Consider an SLK sentence $\varphi = \langle\!\langle x_0 \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle\, \psi$ that holds at a global state $s \in \text{St}$ in some interpreted system $\mathcal{I}$. Since $\mathcal{I}, (\emptyset, s) \models_{\text{SLK}} \varphi$, by Lemma 4, there must be some witness strategies $f_0, \ldots, f_{n-1}$ for φ at s, which we want to synthesise. The corresponding assignment $\chi \in \text{Asg}$ on the variables $x_0, \ldots, x_{n-1}$ satisfies $\mathcal{I}, (\chi, s) \models_{\text{SLK}} \psi$. Let $\text{E} = \texttt{Check}_{\mathcal{I}}(\psi, \emptyset)$ be the set of extended states which guarantee ψ in $\mathcal{I}$ under the empty binding. E contains all possible extended states $\langle s', v' \rangle$ such that $\mathcal{I}, (s', v') \models_{\text{SLK}} \psi$. Hence, it must be the case that $\langle s, v \rangle \in \text{E}$ where v is some variable assignment which extends χ.

Therefore, in order to synthesise witness strategies for φ at s, it suffices to pick an arbitrary extended state $\langle s, v \rangle \in \texttt{Check}_{\mathcal{I}}(\psi, \emptyset)$. The witness strategies for φ at s are then $v(x_0), \ldots, v(x_{n-1})$.

Lemma 5. *Let $\mathcal{I}$ be an interpreted system, ψ an agent-closed* SLK *formula such that* $\mathsf{free}(\psi) = \{x_0, \ldots, x_{n-1}\}$ *and* $\varphi = \langle\!\langle x_0 \rangle\!\rangle \ldots \langle\!\langle x_{n-1} \rangle\!\rangle \psi$ *an* SLK *sentence. Then the following properties hold:*

1. *For all variable assignments* $\langle s, v\rangle \in \mathtt{Check}_{\mathcal{I}}(\psi, \emptyset)$, $v(x_0), \ldots, v(x_{n-1})$ *are witness strategies for* φ *at* s.
2. *If there exist witness strategies* $f_0, \ldots, f_{n-1}$ *for* φ *at* s, *then there exists a variable assignment* $v \in \mathrm{VAsg}$ *such that* $v(x_i) = f_i$ *for* $0 \leq i < n$ *and* $\langle s, v\rangle \in \mathtt{Check}_{\mathcal{I}}(\psi, \emptyset)$.

Proof. We prove both properties separately:

1. Take an arbitrary extended state $\langle s, v\rangle \in \mathtt{Check}_{\mathcal{I}}(\psi, \emptyset)$. Observe that the SLK model checking algorithm (see item 5 in Definition 13) ensures uniformity of v wrt $x_0, \ldots, x_{n-1}$, *i.e.*, $v(x_i) \in \mathrm{UStr}_{\mathsf{shr}(\psi, x_i)}$ for $0 \leq i < n$. As in the proof of Theorem 2, $\langle s, v\rangle$ guarantees ψ in $\mathcal{I}$ under $\emptyset$. By Definition 10, $\mathcal{I}, (v, s) \models_{\mathrm{SLK}} \psi$. By SLK semantics (Definition 5), we have $\mathcal{I}, (\chi, s) \models_{\mathrm{SLK}} \psi$ where $\chi = \{(x_i, v(x_i)) \mid 0 \leq i < n\}$ since $\mathsf{free}(\psi) \subseteq \{x_0, \ldots, x_{n-1}\} \subseteq \mathsf{dom}(v)$. Hence, $v(x_0), \ldots, v(x_{m-1})$ satisfy both conditions for being witness strategies for φ at s (see Definition 14).
2. Assume that $f_0, \ldots, f_{n-1}$ are witness strategies for φ at s. By Definition 14, $f_i \in \mathrm{UStr}_{\mathsf{shr}(\psi, x_i)}$ for all $0 \leq i < n$ and $\mathcal{I}, (\chi, s) \models_{\mathrm{SLK}} \psi$ where $\chi = \{(x_i, f_i) \mid 0 \leq i < n\}$. Since $\mathsf{dom}(\chi) = \mathsf{free}(\psi)$, by SLK semantics (Definition 5), we have $\mathcal{I}, (v, s) \models_{\mathrm{SLK}} \psi$ for all $v \in \mathrm{VAsg}$ such that $\chi \subseteq v$. There must exist at least one such variable assignment v because $\mathsf{dom}(\chi) \subseteq \mathrm{Vr}$ (simply set $v(x) = \chi(x)$ for $x \in \mathsf{dom}(\chi)$ and assign arbitrary strategies to variables $y \in \mathrm{Vr} \setminus \mathsf{dom}(\chi)$). By Definition 10, $\langle s, v\rangle \in \mathrm{Ext}$ guarantees ψ in $\mathcal{I}$ under $\emptyset$. Therefore, $\langle s, v\rangle \in \mathtt{Check}_{\mathcal{I}}(\psi, \emptyset)$ (see proof of Theorem 2). □

Informally, the first property in Lemma 5 expresses *soundness* of the approach, *i.e.*, that it will return only witness strategies for φ at s. Conversely, the second property asserts *completeness* of the approach, *i.e.*, that it will return witness strategies for φ at s, if they exist.

3.3. Symbolic Implementation

In this subsection we discuss how the algorithm presented in Subsection 3.1 can be implemented symbolically using binary decision diagrams. BDDs are an efficient representation for Boolean formulas and are used by many existing model checkers including MCMAS for model checking temporal logic formulas [39]. We here present a modification of the relevant algorithms for SLK.

We start by representing the parameters of the interpreted system by means of Boolean formulas [10]. Given an interpreted system:

$$\mathcal{I} = \left\langle (\mathrm{St}_a, \mathrm{Ac}_a, \mathsf{P}_a, \mathsf{tr}_a)_{a \in \mathrm{Ag}}, \mathsf{I}, \mathsf{h} \right\rangle$$

we can represent global states and decisions as follows [39, 10]:

- For every agent $a \in \mathrm{Ag}$, we can encode her set of internal states $\mathrm{St}^{\mathrm{p}}_a$ with $\mathsf{nv}(a) = \lceil \log_2 |\mathrm{St}_a| \rceil$ Boolean variables. Thus, a *global state* $s = (s^{\mathrm{p}}_1, \ldots, s^{\mathrm{p}}_n, s_{\mathrm{Env}}) \in \mathrm{St}$ can be encoded as a conjunction $s[\overline{v}]$ of the variables in a Boolean vector $\overline{v} = \overline{v}^{\mathrm{p}}_1 \ldots \overline{v}^{\mathrm{p}}_n \overline{v}_{\mathrm{Env}} = (v_0, \ldots, v_{N-1})$, where $N = \sum_{a \in \mathrm{Ag}} \mathsf{nv}(a)$.
- For every agent $a \in \mathrm{Ag}$, we can encode her set of actions Ac_a with $\mathsf{na}(a) = \lceil \log_2 |\mathrm{Ac}_a| \rceil$ Boolean variables. Thus, a *decision* $\delta = (c_1, \cdots, c_n, c_{\mathrm{Env}}) \in \mathrm{Dc}$ can be encoded as a conjunction $\delta[\overline{w}]$ of the variables in a Boolean vector $\overline{w} = \overline{w}_1 \ldots \overline{w}_n \overline{w}_{\mathrm{Env}} = (w_0, \ldots, w_{M-1})$, where $M = \sum_{a \in \mathrm{Ag}} \mathsf{na}(a)$.

3.3.1. Running Example

Consider a variant of the classical Nim game [40], where two players take turns to remove one or two objects from a single heap which initially contains four objects. The player who removes the last object wins the game.

We model the scenario as an interpreted system $\mathcal{I}_{\mathrm{Nim}}$ with agents $\mathrm{Ag} = \{\mathrm{A}, \mathrm{B}, \mathrm{Env}\}$, where the proper agents $\Sigma = \{\mathrm{A}, \mathrm{B}\}$ correspond to the two players:

- We fix the *environment states* $\mathrm{St}_{\mathrm{Env}} = \{ s^{\mathrm{Env}}_{rq} \mid 0 \le r \le 4 \wedge q \in \Sigma \}$, where r is the number of objects remaining on the heap and q is the active player, whose turn it is to remove objects. Both players have a single internal state, $\mathrm{St}^{\mathrm{p}}_{\mathrm{A}} = \mathrm{St}^{\mathrm{p}}_{\mathrm{B}} = \{\mathtt{I}\}$, and see the whole environment state, *i.e.*,$\mathsf{vis}_p(s_{\mathrm{Env}}) = s_{\mathrm{Env}}$ for all $p \in \Sigma$, $s_{\mathrm{Env}} \in \mathrm{St}_{\mathrm{Env}}$. Hence:

$$\begin{aligned} \mathrm{St}_p &= \{ s^p_{rq} = (\mathtt{I}, s^{\mathrm{Env}}_{rq}) \mid s^{\mathrm{Env}}_{rq} \in \mathrm{St}_{\mathrm{Env}} \} \quad \text{for } p \in \Sigma \\ \mathrm{St} &= \{ s_{rq} = (\mathtt{I}, \mathtt{I}, s^{\mathrm{Env}}_{rq}) \mid s^{\mathrm{Env}}_{rq} \in \mathrm{St}_{\mathrm{Env}} \} \end{aligned}$$

- A player can perform three possible *actions*: do nothing, remove one object, or remove two objects, which is formally represented as $\mathrm{Ac}_{\mathrm{A}} = \mathrm{Ac}_{\mathrm{B}} = \{\perp, 1, 2\}$. The environment always does nothing, *i.e.*, $\mathrm{Ac}_{\mathrm{Env}} = \{\perp\}$.
- The *protocols* of the agents require that the active player removes one or two objects:

$$\mathsf{P}_p(s^p_{rq}) = \begin{cases} \{1,2\} & \text{if } p = q \wedge r > 1 \\ \{1\} & \text{if } p = q \wedge r > 0 \\ \{\perp\} & \text{otherwise} \end{cases} \quad \text{for } p \in \Sigma, s^p_{rq} \in \mathrm{St}_p$$
$$\mathsf{P}_{\mathrm{Env}}(s^{\mathrm{Env}}_{rq}) = \{\perp\} \quad \text{for } s^{\mathrm{Env}}_{rq} \in \mathrm{St}_{\mathrm{Env}}$$

- The *evolution functions* simply reflect the active player's action for $\delta \in \mathrm{Dc}$:

$$\mathsf{tr}_p(s^p_{rq}, \delta) = \mathtt{I} \quad \text{for } p \in \Sigma, s^p_{rq} \in \mathrm{St}$$
$$\mathsf{tr}_{\mathrm{Env}}(s^{\mathrm{Env}}_{rq}, \delta) = \begin{cases} s^{\mathrm{Env}}_{(r - \mathsf{c}_{\mathrm{A}}(\delta))\mathrm{B}} & \text{if } q = \mathrm{A} \wedge r > 0 \\ s^{\mathrm{Env}}_{(r - \mathsf{c}_{\mathrm{B}}(\delta))\mathrm{A}} & \text{if } q = \mathrm{B} \wedge r > 0 \\ s^{\mathrm{Env}}_{rq} & \text{otherwise} \end{cases} \quad \text{for } s^{\mathrm{Env}}_{rq} \in \mathrm{St}_{\mathrm{Env}}$$

- The set of *initial states* is $\mathrm{I} = \{s_{4\mathrm{A}}\} = \{(\mathtt{I}, \mathtt{I}, s_{4\mathrm{A}}^{\mathrm{Env}})\}$, *i.e.*, the heap initially contains 4 objects and player A plays first.
- Finally, we use two *atomic propositions* $\mathrm{AP} = \{\mathrm{win}_\mathrm{A}, \mathrm{win}_\mathrm{B}\}$ and reflect the winning condition in the *valuation function*: $\mathsf{h}(\mathrm{win}_\mathrm{A}) = \{s_{0\mathrm{B}}\}$ and $\mathsf{h}(\mathrm{win}_\mathrm{B}) = \{s_{0\mathrm{A}}\}$.

To encode the global states St, the symbolic implementation uses a Boolean vector $\overline{v} = (v_0, v_1, v_2, v_3)$ with $\mathsf{nv}(\mathrm{A}) + \mathsf{nv}(\mathrm{B}) + \mathsf{nv}(\mathrm{Env}) = 0 + 0 + 4 = 4$ variables. Variables v_0, v_1, v_2 are used to encode the number of remaining objects on the heap as a binary number and variable v_3 is true iff B is the active player. For example, $s_{1\mathrm{A}}[\overline{v}] = \neg v_0 \wedge \neg v_1 \wedge v_2 \wedge \neg v_3$.

Similarly, the decisions Dc are represented using a Boolean vector $\overline{w} = (w_0, w_1, w_2, w_3)$ with $\mathsf{na}(\mathrm{A}) + \mathsf{na}(\mathrm{B}) + \mathsf{na}(\mathrm{Env}) = 2 + 2 + 0 = 4$ variables. Variables w_0, w_1 and w_2, w_3 encode the actions of players A and B, respectively, using binary encoding (treating $\perp$ as zero). For example, $(2, \perp, \perp)[\overline{w}] = w_0 \wedge \neg w_1 \wedge \neg w_2 \wedge \neg w_3$.

3.3.2. Encoding of Extended States

In order to implement the SLK model checking algorithm, we need to represent sets of *extended states* (see Definition 9), which consist of a global state and a variable assignment. We thus represent the variable assignment explicitly using Boolean variables as well.

Since the number of strategy variables (and hence the domain of a variable assignment) depends on the SLK formula we are checking, the total number of Boolean variables also depends on the formula. Let $\varphi \in \mathrm{SLK}$ be an SLK sentence and $\mathsf{vars}(\varphi) \subseteq \mathrm{Vr}$ the set of variables quantified in φ. For each variable $x \in \mathsf{vars}(\varphi)$, we represent the strategy associated with x using a number of Boolean variables.

Let us now consider an arbitrary variable $x \in \mathsf{vars}(\varphi)$. In the SLK model checking algorithm (Definition 13), the variable quantifies over strategies $f \in \mathrm{UStr}_{\mathsf{shr}(\varphi,x)}$. The domain and the range of a strategy $f \in \mathrm{UStr}_\mathrm{A}$ are St and Ac_A, respectively (see Subsection 2.3). Hence, we could represent it using $|\mathrm{St}| \times \lceil \log_2 |\mathrm{Ac}_\mathrm{A}| \rceil$ Boolean variables by encoding the action associated with each global state. By considering the protocols[2] of all agents in A, we can reduce the number of Boolean variables to $\sum_{s \in \mathrm{St}} \left\lceil \log_2 \left| \bigcap_{a \in \mathrm{A}} \mathsf{P}_a(\mathsf{s}_a(s)) \right| \right\rceil$.

Furthermore, we can exploit the fact that the strategies are *uniform*. Intuitively, if we have a strategy $f \in \mathrm{UStr}_{\{a\}}$ for an agent $a \in \mathrm{Ag}$, then for all states $s_1, s_2 \in \mathrm{St}$, $s_1 \sim_a s_2$ implies $f(s_1) = f(s_2)$. Hence, there is no point in storing both actions $f(s_1)$ and $f(s_2)$. More generally, the epistemic accessibility relations $\sim_a$ with $a \in \mathrm{A}$ induce regions of the global state space, to which the uniform strategies $f \in \mathrm{UStr}_\mathrm{A}$ must assign the same action. It turns out that these regions are equivalence classes with respect to the common epistemic accessibility relation $\sim_\mathrm{A}^{\mathsf{C}}$.

[2] $|\cap_{a \in \mathrm{A}} \mathsf{P}_a(\mathsf{s}_a(s))| \leq |\mathrm{Ac}_\mathrm{A}|$ for each global state $s \in \mathrm{St}$.

Lemma 6. *Let $\mathcal{I}$ be an interpreted system and* $\mathrm{A} \subseteq \mathrm{Ag}$ *a set of agents. Then a memoryless strategy* $f : \mathrm{St} \to \mathrm{Ac}_{\mathrm{A}}$ *is* uniform *iff for each set of global states in the quotient set* $\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}$*, we have* $f(s_1) = f(s_2)$ *for all* $s_1, s_2 \in \mathrm{S}$.

Proof. We will prove both directions of the equivalence separately:

$\Rightarrow$: Assume that f is uniform and take an arbitrary set $\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}$. Furthermore, take arbitrary global states $s_1, s_2 \in \mathrm{S}$. By the definition of quotient set, we have $s_1 \sim^{\mathsf{C}}_{\mathrm{A}} s_2$. There are two cases:

- $s_1 = s_2$. Trivially, $f(s_1) = f(s_2)$.
- By the definition of common epistemic accessibility relation (see Subsection 2.3), there is a chain of global states $s'_1, s'_2, \ldots, s'_n \in \mathrm{St}$ and agents $i_1, i_2, \ldots, i_{n+1} \in \mathrm{A}$ with $n \geq 0$ such that $s_1 \sim_{i_1} s'_1 \sim_{i_2} s'_2 \ldots s'_n \sim_{i_{n+1}} s_2$. By uniformity of f, we get $f(s_1) = f(s'_1) = \cdots = f(s'_n) = f(s_2)$.

$\Leftarrow$: Assume that for each $\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}$, we have $f(s_1) = f(s_2)$ for all $s_1, s_2 \in \mathrm{S}$. Take an arbitrary agent $a \in \mathrm{A}$ and global states $s_1, s_2 \in \mathrm{St}$ such that $s_1 \sim_a s_2$. To prove uniformity of f, we need to show that $f(s_1) = f(s_2)$. Since $s_1 \sim_a s_2$, we also have $s_1 \sim^{\mathsf{C}}_{\mathrm{A}} s_2$. Hence, s_2 belongs to the equivalence class $[s_1]_{\sim^{\mathsf{C}}_{\mathrm{A}}}$. By definition of an equivalence class, it must be the case that $s_1 \in [s_1]_{\sim^{\mathsf{C}}_{\mathrm{A}}}$ and $[s_1]_{\sim^{\mathsf{C}}_{\mathrm{A}}} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}$. Since $s_1, s_2 \in [s_1]_{\sim^{\mathsf{C}}_{\mathrm{A}}}$ and $[s_1]_{\sim^{\mathsf{C}}_{\mathrm{A}}} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}$, we get $f(s_1) = f(s_2)$ by the initial assumption as required. □

This allows us to present an even more compact representation of a strategy $f \in \mathrm{UStr}_{\mathrm{A}}$ with $\mathrm{A} \subseteq \mathrm{Ag}$ in an interpreted system $\mathcal{I}$. We only need to store one action for each *shared local state* $\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}$. Thus, we can represent f using $\sum_{\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathrm{A}}}} \left\lceil \log_2 \left| \bigcap_{s \in \mathrm{S}} \bigcap_{a \in \mathrm{A}} \mathsf{P}_a(\mathsf{s}_a(s)) \right| \right\rceil$ Boolean variables. Finally, a variable assignment $v \in \mathrm{VAsg}$ for a formula $\varphi \in \textsc{Slk}$ can be represented using a Boolean vector $\overline{u} = (u_0, \ldots, u_{K-1})$ such that[3]:

$$K = \sum_{x \in \mathsf{vars}(\varphi)} \sum_{\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathsf{shr}(\varphi,x)}}} \left\lceil \log_2 \left| \bigcap_{s \in \mathrm{S}} \bigcap_{a \in \mathsf{shr}(\varphi,x)} \mathsf{P}_a(\mathsf{s}_a(s)) \right| \right\rceil$$

as follows:

$$v[\overline{u}] := \bigwedge_{x \in \mathsf{vars}(\varphi)} \bigwedge_{\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathsf{shr}(\varphi,x)}}} v(x)(\mathrm{S})[\overline{u}_{\mathrm{S},x}]$$

[3]Assuming that each strategy and action is stored separately in $\overline{u}$, the provided representation is optimal. If we relaxed this assumption, the number of Boolean variables could be reduced to $\left\lceil \log_2 \left| \prod_{x \in \mathsf{vars}(\varphi)} \prod_{\mathrm{S} \in \mathrm{St}/{\sim^{\mathsf{C}}_{\mathsf{shr}(\varphi,x)}}} \bigcap_{s \in \mathrm{S}} \bigcap_{a \in \mathsf{shr}(\varphi,x)} \mathsf{P}_a(\mathsf{s}_a(s)) \right| \right\rceil$. However, we do not pursue this approach any further because it would complicate the symbolic implementation of the labelling algorithm.

where $v(x)(\mathrm{S})[\overline{u}_{\mathrm{S},x}]$ is the Boolean formula representing that the single action assigned by the uniform strategy mapped to variable x in all states $s \in \mathrm{S}$ is $v(x)(s)$ for any $s \in \mathrm{S}$.

Note that despite both optimisations, the worst case still remains $K = |\mathsf{vars}(\varphi)| \times |\mathrm{St}| \times \lceil \log_2 (\max_{a \in \mathrm{Ag}} |\mathrm{Ac}_a|) \rceil$, *i.e.*, we need polynomially many Boolean variables with respect to both the size of the model $|\mathcal{I}|$ and the number of strategy variables in the formula $|\mathsf{vars}(\varphi)|$.

An extended state $\langle s, v \rangle \in \mathrm{Ext}$ can be represented by the conjunction $\langle s, v \rangle[\overline{v}, \overline{u}] := s[\overline{v}] \wedge v[\overline{u}]$ over the vectors $\overline{v}$ and $\overline{u}$. A set of extended states $\mathrm{E} \subseteq \mathrm{Ext}$ can in turn be expressed as the disjunction $\mathrm{E}[\overline{v}, \overline{u}] := \bigvee_{\langle s,v \rangle \in \mathrm{E}} s[\overline{v}] \wedge v[\overline{u}]$.

Consider the SLK sentence $\varphi_{\mathrm{win}_\mathrm{A}} = \langle\!\langle x \rangle\!\rangle \, [\![y]\!] \, [\![e]\!] \, (\mathrm{A}, x)(\mathrm{B}, y)(\mathrm{Env}, e) \mathsf{F}\, \mathrm{win}_\mathrm{A}$, which specifies that there is a strategy for player A to win regardless of the strategies followed by the other player and the environment. Our Boolean encoding represents variable assignments for $\varphi_{\mathrm{win}_\mathrm{A}}$ in $\mathcal{I}_{\mathrm{Nim}}$ using a Boolean vector $\overline{u}$ with $K = 6$ variables:

$$\begin{aligned}
K = & \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{Env}}(s^{\mathrm{Env}}_{0\mathrm{A}}) \right| \right\rceil + \cdots + \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{Env}}(s^{\mathrm{Env}}_{4\mathrm{A}}) \right| \right\rceil + \\
& \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{Env}}(s^{\mathrm{Env}}_{0\mathrm{B}}) \right| \right\rceil + \cdots + \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{Env}}(s^{\mathrm{Env}}_{4\mathrm{B}}) \right| \right\rceil + \\
& \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{A}}(s^{\mathrm{A}}_{0\mathrm{A}}) \right| \right\rceil + \cdots + \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{A}}(s^{\mathrm{A}}_{4\mathrm{A}}) \right| \right\rceil + \\
& \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{A}}(s^{\mathrm{A}}_{1\mathrm{B}}) \right| \right\rceil + \cdots + \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{A}}(s^{\mathrm{A}}_{4\mathrm{B}}) \right| \right\rceil + \\
& \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{B}}(s^{\mathrm{B}}_{0\mathrm{A}}) \right| \right\rceil + \cdots + \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{B}}(s^{\mathrm{B}}_{4\mathrm{A}}) \right| \right\rceil + \\
& \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{B}}(s^{\mathrm{B}}_{0\mathrm{B}}) \right| \right\rceil + \cdots + \left\lceil \log_2 \left| \mathsf{P}_{\mathrm{B}}(s^{\mathrm{B}}_{4\mathrm{B}}) \right| \right\rceil \\
= & \lceil \log_2 |\{\bot\}| \rceil + \cdots + \lceil \log_2 |\{\bot\}| \rceil + \\
& \lceil \log_2 |\{\bot\}| \rceil + \cdots + \lceil \log_2 |\{\bot\}| \rceil + \\
& \lceil \log_2 |\{\bot\}| \rceil + \lceil \log_2 |\{1\}| \rceil + \lceil \log_2 |\{1,2\}| \rceil + \lceil \log_2 |\{1,2\}| \rceil + \lceil \log_2 |\{1,2\}| \rceil + \\
& \lceil \log_2 |\{\bot\}| \rceil + \cdots + \lceil \log_2 |\{\bot\}| \rceil + \\
& \lceil \log_2 |\{\bot\}| \rceil + \cdots + \lceil \log_2 |\{\bot\}| \rceil + \\
& \lceil \log_2 |\{\bot\}| \rceil + \lceil \log_2 |\{1\}| \rceil + \lceil \log_2 |\{1,2\}| \rceil + \lceil \log_2 |\{1,2\}| \rceil + \lceil \log_2 |\{1,2\}| \rceil \\
= & \, 6
\end{aligned}$$

Intuitively, this is because there are only 6 states in which some strategy has two actions to choose from: strategy $f_x = v(x)$ in states $s_{2\mathrm{A}}$, $s_{3\mathrm{A}}$, $s_{4\mathrm{A}}$ and strategy $f_y = v(y)$ in states $s_{2\mathrm{B}}$, $s_{3\mathrm{B}}$, $s_{4\mathrm{B}}$. Each Boolean variable corresponds to one of these situations ($u_0 \rightsquigarrow (x, s_{2\mathrm{A}})$, $u_1 \rightsquigarrow (x, s_{3\mathrm{A}})$, $u_2 \rightsquigarrow (x, s_{4\mathrm{A}})$, $u_3 \rightsquigarrow (y, s_{2\mathrm{B}})$, $u_4 \rightsquigarrow (y, s_{3\mathrm{B}})$, $u_5 \rightsquigarrow (y, s_{4\mathrm{B}})$). If a variable is true, then the relevant strategy prescribes removing two items from the heap in the relevant state. For example, if u_0 is false, then $f_x(s_{2\mathrm{A}}) = 1$. Conversely, if u_4 is true, then $f_y(s_{3\mathrm{B}}) = 2$. Furthermore, a sample extended state $e = (s_{0\mathrm{A}}, \{x \mapsto f_x, y \mapsto f_y, e \mapsto f_e\})$ with strategies:

$$\begin{array}{lll}
f_x(s_{0*}) = \bot & f_y(s_{0*}) = \bot & f_e(s_{**}) = \bot \\
f_x(s_{1\mathrm{A}}) = 1 & f_y(s_{1\mathrm{B}}) = 1 & \\
f_x(s_{2\mathrm{A}}) = 2 \quad [u_0] & f_y(s_{2\mathrm{B}}) = 1 \quad [\neg u_3] & \\
f_x(s_{3\mathrm{A}}) = 1 \quad [\neg u_1] & f_y(s_{3\mathrm{B}}) = 2 \quad [u_4] & \\
f_x(s_{4\mathrm{A}}) = 1 \quad [\neg u_2] & f_y(s_{4\mathrm{B}}) = 2 \quad [u_6] & \\
f_x(s_{*\mathrm{B}}) = \bot & f_y(s_{*\mathrm{A}}) = \bot &
\end{array}$$

is symbolically encoded as $e[\overline{v}, \overline{u}] = \neg v_0 \wedge \neg v_1 \wedge \neg v_2 \wedge \neg v_3 \wedge u_0 \wedge \neg u_1 \wedge \neg u_2 \wedge \neg u_3 \wedge u_4 \wedge u_6$.

3.3.3. Encoding of the Algorithm

Given a binding $b \in \mathrm{Bnd}$ such that $\mathsf{dom}(b) = \mathrm{Ag}$, we define a formula $S^b(e, a)$, where $e \in \mathrm{Ext}$ and $\delta \in \mathrm{Dc}$, representing the *strategy restrictions* of the implied transition relation (see Definition 11). The formula asserts that all agents act according to their strategies:

$$S^b(\langle s, v \rangle, \delta) \triangleq \forall a \in \mathrm{Ag}.\, v(b(a))(s) = \mathsf{c}_a(\delta)$$

Let $\overline{v}$, $\overline{w}$ and $\overline{u}$ be the Boolean vectors for representing current global states, decisions and variable assignments, respectively. We can encode the strategy restrictions as a Boolean formula $S^b[\overline{v}, \overline{w}, \overline{u}]$:

$$S^b[\overline{v}, \overline{w}, \overline{u}] := \bigwedge_{a \in \mathrm{Ag}} \bigvee_{s_a \in \mathrm{St}_a} \left(s_a[\overline{v}_a] \wedge \bigvee_{c \in \mathsf{P}_a(s_a)} c[\overline{w}_a] \wedge c[\overline{u}_{s_a, b(a)}] \right)$$

where:

- $s_a[\overline{v}_a]$ is the Boolean formula representing that the local state[4] of agent $a \in \mathrm{Ag}$ is $s_a \in \mathrm{St}_a$;
- $c[\overline{w}_a]$ is the Boolean formula representing that the action of agent $a \in \mathrm{Ag}$ is $c \in \mathrm{Ac}_a$;
- $c[\overline{u}_{s_a, b(a)}]$ is the Boolean formula representing that the action in local state[5] s_a assigned by the strategy mapped to variable $b(a)$ is $c \in \mathrm{Ac}_a$.

Given the binding $b = (\mathrm{A} \mapsto x, \mathrm{B} \mapsto y, \mathrm{Env} \mapsto e)$ in $\varphi_{\mathrm{win}_\mathrm{A}}$, the strategy re-

[4] Note that for a proper agent $a \in \Sigma$, the local state is a combination of a private state and an image of the environment state, *i.e.*, $s_a = \big(s_a^{\mathrm{P}}, \mathsf{vis}_a(s_{\mathrm{Env}})\big)$ for some $s_a^{\mathrm{P}} \in \mathrm{St}_a^{\mathrm{P}}$, $s_{\mathrm{Env}} \in \mathrm{St}_{\mathrm{Env}}$. This is reflected in the encoding: $s_a[\overline{v}_a] = \big(s_a^{\mathrm{P}}, \mathsf{vis}_a(s_{\mathrm{Env}})\big)[\overline{v}_a] := s_a^{\mathrm{P}}[\overline{v}_a^{\mathrm{P}}] \wedge \mathsf{vis}_a(s_{\mathrm{Env}})[\overline{v}_{\mathrm{Env}}]$.

Typically, vis_a is encoded as a filter over $\overline{v}_{\mathrm{Env}}$, *e.g.*, `Obsvars` and `Lobsvars` sections of the ISPL file format used by MCMAS [20]. Intuitively, only some Boolean variables encoding the environment state are visible to agent a.

[5] Strictly speaking, the strategy maps *global* states to actions. However, as explained earlier, there exists a set $\mathrm{S} \subseteq \mathrm{St}/\sim^{\mathsf{C}}_{\mathsf{shr}(\varphi, b(a))}$ such that, for all global states $s \in \mathrm{St}$, if $\mathsf{s}_a(s) = s_a$ then $s \in \mathrm{S}$. Due to uniformity, the strategy mapped to $b(a)$ must assign the same action to all global states in S. Thus, we can also interpret the strategy as a mapping from *local* states to actions.

strictions for our running example are encoded as follows:

$$\begin{aligned}
S^b[\overline{v}, \overline{w}, \overline{u}] = {} & (v\texttt{0000} \wedge w\texttt{00--} \wedge u\texttt{------} \vee v\texttt{0001} \wedge w\texttt{00--} \wedge u\texttt{------} \vee \\
& \; v\texttt{0010} \wedge w\texttt{01--} \wedge u\texttt{------} \vee v\texttt{0011} \wedge w\texttt{00--} \wedge u\texttt{------} \vee \\
& \; v\texttt{0100} \wedge (w\texttt{01--} \wedge u\texttt{0-----} \vee w\texttt{10--} \wedge u\texttt{1-----}) \vee v\texttt{0101} \wedge w\texttt{00--} \wedge u\texttt{------} \vee \\
& \; v\texttt{0110} \wedge (w\texttt{01--} \wedge u\texttt{-0----} \vee w\texttt{10--} \wedge u\texttt{-1----}) \vee v\texttt{0111} \wedge w\texttt{00--} \wedge u\texttt{------} \vee \\
& \; v\texttt{1000} \wedge (w\texttt{01--} \wedge u\texttt{--0---} \vee w\texttt{10--} \wedge u\texttt{--1---}) \vee v\texttt{1001} \wedge w\texttt{00--} \wedge u\texttt{------}) \wedge \\
& (v\texttt{0000} \wedge w\texttt{--00} \wedge u\texttt{------} \vee v\texttt{0001} \wedge w\texttt{--00} \wedge u\texttt{------} \vee \\
& \; v\texttt{0010} \wedge w\texttt{--00} \wedge u\texttt{------} \vee v\texttt{0011} \wedge w\texttt{--01} \wedge u\texttt{------} \vee \\
& \; v\texttt{0100} \wedge w\texttt{--00} \wedge u\texttt{------} \vee v\texttt{0101} \wedge (w\texttt{--01} \wedge u\texttt{---0--} \vee w\texttt{--10} \wedge u\texttt{---1--}) \vee \\
& \; v\texttt{0110} \wedge w\texttt{--00} \wedge u\texttt{------} \vee v\texttt{0111} \wedge (w\texttt{--01} \wedge u\texttt{----0-} \vee w\texttt{--10} \wedge u\texttt{----1-}) \vee \\
& \; v\texttt{1000} \wedge w\texttt{--00} \wedge u\texttt{------} \vee v\texttt{1001} \wedge (w\texttt{--01} \wedge u\texttt{-----0} \vee w\texttt{--10} \wedge u\texttt{-----1})) \wedge \\
& (v\texttt{0000} \wedge w\texttt{----} \wedge u\texttt{------} \vee v\texttt{0001} \wedge w\texttt{----} \wedge u\texttt{------} \vee \\
& \; v\texttt{0010} \wedge w\texttt{----} \wedge u\texttt{------} \vee v\texttt{0011} \wedge w\texttt{----} \wedge u\texttt{------} \vee \\
& \; v\texttt{0100} \wedge w\texttt{----} \wedge u\texttt{------} \vee v\texttt{0101} \wedge w\texttt{----} \wedge u\texttt{------} \vee \\
& \; v\texttt{0110} \wedge w\texttt{----} \wedge u\texttt{------} \vee v\texttt{0111} \wedge w\texttt{----} \wedge u\texttt{------} \vee \\
& \; v\texttt{1000} \wedge w\texttt{----} \wedge u\texttt{------} \vee v\texttt{1001} \wedge w\texttt{----} \wedge u\texttt{------})
\end{aligned}$$

Above we replaced conjunctions of variables within a Boolean vector with a sequential representation of their sign or absence. For example $w\texttt{--01}$ is a shorthand for $\neg w_2 \wedge w_3$.

Furthermore, let $\overline{v'}$ be the Boolean vector representing successor (global) states. We can then represent the global protocol and the evolution function as Boolean formulas $\mathsf{P}[\overline{v}, \overline{w}]$ and $\mathsf{tr}[\overline{v}, \overline{w}, \overline{v'}]$ by taking the conjunctions of Boolean formulas representing the individual agents' protocols P_a and the evolution functions tr_a for $a \in \mathrm{Ag}$:

$$\mathsf{P}[\overline{v}, \overline{w}] := \bigwedge_{a \in \mathrm{Ag}} \bigvee_{s_a \in \mathrm{St}_a} \left(s_a[\overline{v}_a] \wedge \bigvee_{c \in \mathsf{P}_a(s_a)} c[\overline{w}_a] \right)$$

$$\mathsf{tr}[\overline{v}, \overline{w}, \overline{v'}] := \bigwedge_{a \in \mathrm{Ag}} \bigvee_{s_a \in \mathrm{St}_a} \left(s_a[\overline{v}_a] \wedge \bigvee_{\delta \in \mathrm{Dc}} \delta[\overline{w}] \wedge \mathsf{tr}_a(s_a, \delta)[\overline{v'}^{\mathrm{p}}_a] \right)$$

The evolution function for our running example is encoded as follows:

$$\begin{aligned}
\mathsf{tr}[\overline{v}, \overline{w}, \overline{v'}] = {} & (v\texttt{0000} \wedge w\texttt{0000} \wedge v'\texttt{0000} \vee v\texttt{0001} \wedge w\texttt{0000} \wedge v'\texttt{0001} \vee \\
& v\texttt{0010} \wedge w\texttt{0100} \wedge v'\texttt{0001} \vee v\texttt{0011} \wedge w\texttt{0100} \wedge v'\texttt{0000} \vee \\
& v\texttt{0100} \wedge (w\texttt{0100} \wedge v'\texttt{0011} \vee w\texttt{1000} \wedge v'\texttt{0001}) \vee \\
& v\texttt{0101} \wedge (w\texttt{0001} \wedge v'\texttt{0010} \vee w\texttt{0010} \wedge v'\texttt{0000}) \vee \\
& v\texttt{0110} \wedge (w\texttt{0100} \wedge v'\texttt{0101} \vee w\texttt{1000} \wedge v'\texttt{0011}) \vee \\
& v\texttt{0111} \wedge (w\texttt{0001} \wedge v'\texttt{0100} \vee w\texttt{0010} \wedge v'\texttt{0010}) \vee \\
& v\texttt{1000} \wedge (w\texttt{0100} \wedge v'\texttt{0111} \vee w\texttt{1000} \wedge v'\texttt{0101}) \vee \\
& v\texttt{1001} \wedge (w\texttt{0001} \wedge v'\texttt{0110} \vee w\texttt{0010} \wedge v'\texttt{0100}))
\end{aligned}$$

The Boolean formula $R^b_t[\overline{v}, \overline{w}, \overline{v'}]$ for the *implied transition relation* $\rightarrow^b_v \subseteq \mathrm{St} \times \mathrm{St}$ (see Definition 11) is then constructed from the conjunction of the Boolean formulas representing the global protocol, the global evolution function and the strategy restrictions[6]:

$$R_t^b[\overline{v}, \overline{v'}, \overline{u}] := \exists \overline{w}.\, \mathsf{P}[\overline{v}, \overline{w}] \wedge \mathsf{tr}[\overline{v}, \overline{w}, \overline{v'}] \wedge S^b[\overline{v}, \overline{w}, \overline{u}]$$

Note that we quantify over actions, encoded as $\overline{w}$, but we keep the variable assignment in the extra parameter $\overline{u}$.[7] Quantification over the variable assignment is performed when a strategy quantifier ($\langle\!\langle x \rangle\!\rangle$, $[\![x]\!]$) is encountered. Also note that the strategy restrictions S^b depend on the binding b and thus have to be recomputed when the binding is updated, *i.e.*, when the agent binding operator (a, x) is encountered. This is not the case for the other Boolean formulas ($\mathsf{P}[\overline{v}, \overline{w}]$ and $\mathsf{tr}[\overline{v}, \overline{w}, \overline{v'}]$), which are constant for a given interpreted system.

The implied transition relation for our running example is encoded as follows:

$$\begin{aligned}
R_t^b[\overline{v}, \overline{v'}, \overline{u}] = (&v\mathsf{0000} \wedge v'\mathsf{0000} \vee v\mathsf{0001} \wedge v'\mathsf{0001} \vee \\
&v\mathsf{0010} \wedge v'\mathsf{0001} \vee v\mathsf{0011} \wedge v'\mathsf{0000} \vee \\
&v\mathsf{0100} \wedge (v'\mathsf{0011} \wedge u\mathsf{0{-}{-}{-}{-}{-}} \vee v'\mathsf{0001} \wedge u\mathsf{1{-}{-}{-}{-}{-}}) \vee \\
&v\mathsf{0101} \wedge (v'\mathsf{0010} \wedge u\mathsf{{-}{-}{-}0{-}{-}} \vee v'\mathsf{0000} \wedge u\mathsf{{-}{-}{-}1{-}{-}}) \vee \\
&v\mathsf{0110} \wedge (v'\mathsf{0101} \wedge u\mathsf{{-}0{-}{-}{-}{-}} \vee v'\mathsf{0011} \wedge u\mathsf{{-}1{-}{-}{-}{-}}) \vee \\
&v\mathsf{0111} \wedge (v'\mathsf{0100} \wedge u\mathsf{{-}{-}{-}{-}0{-}} \vee v'\mathsf{0010} \wedge u\mathsf{{-}{-}{-}{-}1{-}}) \vee \\
&v\mathsf{1000} \wedge (v'\mathsf{0111} \wedge u\mathsf{{-}{-}0{-}{-}{-}} \vee v'\mathsf{0101} \wedge u\mathsf{{-}{-}1{-}{-}{-}}) \vee \\
&v\mathsf{1001} \wedge (v'\mathsf{0110} \wedge u\mathsf{{-}{-}{-}{-}{-}0} \vee v'\mathsf{0100} \wedge u\mathsf{{-}{-}{-}{-}{-}1}))
\end{aligned}$$

The *epistemic accessibility relations* R_a^{K}, $R_{\mathrm{A}}^{\mathsf{E}}$, $R_{\mathrm{A}}^{\mathsf{D}}$, $R_{\mathrm{A}}^{\mathsf{C}}$ for an agent $a \in \mathrm{Ag}$ and a set of agents $\mathrm{A} \subseteq \mathrm{Ag}$ are also constant for a given interpreted system and are encoded in a similar fashion [39]:

$$\begin{aligned}
R_a^{\mathsf{K}}[\overline{v}, \overline{v'}] &:= \bigvee_{(s,s') \in \sim_a} s[\overline{v}] \wedge s'[\overline{v'}] \\
R_{\mathrm{A}}^{\mathsf{E}}[\overline{v}, \overline{v'}] &:= \bigvee_{a \in \mathrm{A}} R_a^{\mathsf{K}}[\overline{v}, \overline{v'}] \\
R_{\mathrm{A}}^{\mathsf{D}}[\overline{v}, \overline{v'}] &:= \bigwedge_{a \in \mathrm{A}} R_a^{\mathsf{K}}[\overline{v}, \overline{v'}] \\
R_{\mathrm{A}}^{\mathsf{C}}[\overline{v}, \overline{v'}] &:= \mathrm{lfp}_\rho\big(R_{\mathrm{A}}^{\mathsf{E}}[\overline{v}, \overline{v'}] \vee \big(\exists \overline{v^\circ}.\, \rho[\overline{v}, \overline{v^\circ}] \wedge \rho[\overline{v^\circ}, \overline{v'}]\big)\big)
\end{aligned}$$

Finally, the algorithm $\mathtt{Check}_{\mathcal{I}} \colon \textsc{Slk} \times \mathrm{Bnd} \to 2^{\mathrm{Ext}}$ can be translated into operations on BDDs representing sets of extended states.

To check the $\textsc{Slk}$ sentence $\varphi_{\mathrm{win}_{\mathrm{A}}}$ in $\mathcal{I}_{\mathrm{Nim}}$, we perform the following steps (see Definition 13):

1. We calculate the set of extended states which guarantee $\mathrm{win}_{\mathrm{A}}$:

$$\mathtt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\mathrm{win}_{\mathrm{A}}, b)[\overline{v}, \overline{u}] = \mathsf{h}(\mathrm{win}_{\mathrm{A}})[v] = v\mathsf{0001}$$

[6]Observe that, for all bindings b and Boolean vectors $\overline{u}$, $\overline{v}$, $\overline{w}$, if $S^b[\overline{v}, \overline{w}, \overline{u}]$ is true, then so is $\mathsf{P}[\overline{v}, \overline{w}]$. In other words, the protocol is already included in the strategy restrictions. Hence, the encoding of $R_t^b[\overline{v}, \overline{v'}, \overline{u}]$ could be simplified to $\exists \overline{w}.\, \mathsf{tr}[\overline{v}, \overline{w}, \overline{v'}] \wedge S^b[\overline{v}, \overline{w}, \overline{u}]$.

[7]Observe that we do not need to allocate BDD variables $\overline{u'}$ for any successor states as these are not affected by a temporal transition (see Definition 12).

2. We evaluate the least fixed point of $F[\overline{v},\overline{u}] = \texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\mathrm{win}_{\mathrm{A}}, b)[\overline{v},\overline{u}] \vee \exists\overline{v'}.\, R_t^b[\overline{v},\overline{u},\overline{v'}] \wedge F[\overline{v'},\overline{u}]$ to find the set of extended states from which $\mathrm{win}_{\mathrm{A}}$ can be eventually guaranteed:

$$\begin{aligned}
F^0[\overline{v},\overline{u}] &= v\texttt{0001}\\
F^1[\overline{v},\overline{u}] &= F^0[\overline{v},\overline{u}] \vee v\texttt{0010} \vee v\texttt{0100} \wedge u\texttt{1-----}\\
F^2[\overline{v},\overline{u}] &= F^1[\overline{v},\overline{u}] \vee v\texttt{0101} \wedge u\texttt{---0--} \vee v\texttt{0111} \wedge u\texttt{----1-} \vee\\
& v\texttt{0111} \wedge u\texttt{1---0-} \vee v\texttt{1001} \wedge u\texttt{1----1}\\
F^3[\overline{v},\overline{u}] &= F^2[\overline{v},\overline{u}] \vee v\texttt{0110} \wedge u\texttt{-0-0--} \vee v\texttt{1000} \wedge u\texttt{--10--} \vee\\
& v\texttt{1000} \wedge u\texttt{--0-1-} \vee v\texttt{1000} \wedge u\texttt{1-0-0-}\\
F^4[\overline{v},\overline{u}] &= F^3[\overline{v},\overline{u}] \vee v\texttt{1001} \wedge u\texttt{-0-0-0} = F^5[\overline{v},\overline{u}]\\
\texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\mathsf{F}\,\mathrm{win}_{\mathrm{A}}, b)[\overline{v},\overline{u}] &= v\texttt{0001} \vee v\texttt{0010} \vee v\texttt{0100} \wedge u\texttt{1-----} \vee v\texttt{0101} \wedge u\texttt{---0--} \vee\\
& v\texttt{0111} \wedge (u\texttt{----1-} \vee u\texttt{1---0-}) \vee v\texttt{0110} \wedge u\texttt{-0-0--} \vee\\
& v\texttt{1000} \wedge (u\texttt{--10--} \vee u\texttt{--0-1-} \vee u\texttt{1-0-0-}) \vee\\
& v\texttt{1001} \wedge (u\texttt{1----1} \vee u\texttt{-0-0-0})
\end{aligned}$$

3. We universally quantify over the Boolean variables associated with strategy variables y and e to find the set of extended states which guarantee $\psi_{\mathrm{win}_{\mathrm{A}}} = [\![y]\!]\,[\![e]\!]\,(\mathrm{A}, x)(\mathrm{B}, y)(\mathrm{Env}, e)\mathsf{F}\,\mathrm{win}_{\mathrm{A}}$:

$$\begin{aligned}
\texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\psi_{\mathrm{win}_{\mathrm{A}}}, \emptyset)[\overline{v},\overline{u}] &= \forall u_3u_4u_5.\, \texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\mathsf{F}\,\mathrm{win}_{\mathrm{A}}, b)[\overline{v},\overline{u}]\\
&= v\texttt{0001} \vee v\texttt{0010} \vee v\texttt{0100} \wedge u\texttt{1-----} \vee\\
& v\texttt{0111} \wedge u\texttt{1-----} \vee v\texttt{1000} \wedge u\texttt{1-0---}
\end{aligned}$$

4. We existentially quantify over the Boolean variables associated with the strategy variable x to find the set of extended states which guarantee $\varphi_{\mathrm{win}_{\mathrm{A}}} = \langle\!\langle x\rangle\!\rangle\psi_{\mathrm{win}_{\mathrm{A}}}$:

$$\begin{aligned}
\texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\varphi_{\mathrm{win}_{\mathrm{A}}}, \emptyset)[\overline{v},\overline{u}] &= \exists u_0u_1u_2.\, \texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\psi_{\mathrm{win}_{\mathrm{A}}}, \emptyset)[\overline{v},\overline{u}]\\
&= v\texttt{0001} \vee v\texttt{0010} \vee v\texttt{0100} \vee v\texttt{0111} \vee v\texttt{1000}
\end{aligned}$$

5. Finally, to determine whether $\varphi_{\mathrm{win}_{\mathrm{A}}}$ holds in all initial states of $\mathcal{I}_{\mathrm{Nim}}$, we check if I is a subset of $\texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\varphi_{\mathrm{win}_{\mathrm{A}}}, \emptyset)$ symbolically:

$$\begin{aligned}
(\mathrm{I} \cap \texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\varphi_{\mathrm{win}_{\mathrm{A}}}, \emptyset))[\overline{v},\overline{u}] &= \mathrm{I}[\overline{v}] \wedge \texttt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\varphi_{\mathrm{win}_{\mathrm{A}}}, \emptyset)[\overline{v},\overline{u}]\\
&= v\texttt{0001} \wedge (v\texttt{0001} \vee v\texttt{0010} \vee v\texttt{0100} \vee v\texttt{0111} \vee v\texttt{1000})\\
&= v\texttt{0001}\\
&= \mathrm{I}[\overline{v}]
\end{aligned}$$

   We conclude that $\varphi_{\mathrm{win}_{\mathrm{A}}}$ holds in $\mathcal{I}_{\mathrm{Nim}}$, *i.e.*, $\mathcal{I}_{\mathrm{Nim}} \models_{\mathrm{SLK}} \varphi_{\mathrm{win}_{\mathrm{A}}}$.

3.3.4. Strategy Synthesis

Synthesising *witness* and *counterexample strategies* (see Definition 14) using the symbolic implementation is straightforward because our representation of variable assignments ensures that all strategies are uniform (see Subsection 2.3).

To synthesise the witness strategies for an SLK sentence $\varphi = \langle\!\langle x_0\rangle\!\rangle \ldots \langle\!\langle x_{n-1}\rangle\!\rangle\, \psi$ at a global state $s \in \mathrm{St}$, we proceed as follows. Since $\mathcal{I}, (\emptyset, s) \models_{\mathrm{SLK}} \varphi$, there exist witness strategies for φ at s (see Lemma 4). We calculate $\mathrm{E} = \texttt{Check}_{\mathcal{I}}(\psi, \emptyset)$ and pick an arbitrary extended state $\langle s, v\rangle \in \mathrm{E}$. Note that all strategies in v are uniform due to our representation. By Lemma 5, there exists at least one such extended state $\langle s, v \rangle$ (because there exist witness strategies for φ at s) and for all such extended states, $v(x_0), \ldots, v(x_{n-1})$ are witness strategies for φ at s. These steps can be implemented symbolically as follows:

1. Calculate $\mathrm{E} = \mathtt{Check}_{\mathcal{I}}(\psi, \emptyset)$. This is performed using the symbolic implementation presented earlier in this subsection.
2. Remove from the set E all the extended states with a different underlying global state $\mathrm{E}' = \{(s', v') \in \mathrm{E} \mid s' = s\}$. The symbolic representation of this operation is $\mathrm{E}'[\overline{v}, \overline{u}] := \mathrm{E}[\overline{v}, \overline{u}] \wedge s[\overline{v}]$.
3. Pick an arbitrary extended state $\langle s, v \rangle \in \mathrm{E}'$. This is equivalent to selecting one conjunct (also referred to as *minterm*) $C[\overline{v}, \overline{u}]$ from $\mathrm{E}'[\overline{v}, \overline{u}]$. BDD packages usually provide a built-in function for this operation[8].
4. The conjunct $C[\overline{v}, \overline{u}]$ encodes the extended state $\langle s, v \rangle$, where v contains the witness strategies. To get the next action $f_i(s')$ of a strategy $f_i = v(x_i)$ for $0 \leq i < n$ at a global state $s' \in \mathrm{St}$, we find a possible action $c \in \bigcap_{a \in \mathsf{shr}(\psi, x_i)} \mathsf{P}_a(\mathsf{s}_a(s'))$ such that $C[\overline{v}, \overline{u}] \wedge c[\overline{u}_{s', x_i}]$ is *not* equivalent to false, where $c[\overline{u}_{s', x_i}]$ is the Boolean formula representing the fact that the next action of the strategy mapped to variable x_i at the global state s' is c.

To finish our running example, we show how to synthesise a witness strategy $f_x \colon \mathrm{St} \to \mathrm{Ac}_{\mathrm{A}}$ for x at the initial state $s_{4\mathrm{A}}$ in $\varphi_{\mathrm{win}_{\mathrm{A}}} = \langle\!\langle x \rangle\!\rangle \psi_{\mathrm{win}_{\mathrm{A}}}$:

1. $\mathrm{E}[\overline{v}, \overline{u}] = \mathtt{Check}_{\mathcal{I}_{\mathrm{Nim}}}(\psi_{\mathrm{win}_{\mathrm{A}}}, \emptyset)[\overline{v}, \overline{u}] = v\mathtt{0001} \vee v\mathtt{0010} \vee v\mathtt{0100} \wedge u\mathtt{1-----} \vee v\mathtt{0111} \wedge u\mathtt{1-----} \vee v\mathtt{1000} \wedge u\mathtt{1-0---}$ (we have already done this).
2. $\mathrm{E}'[\overline{v}, \overline{u}] = \mathrm{E}[\overline{v}, \overline{u}] \wedge s_{4\mathrm{A}}[\overline{v}] = v\mathtt{1000} \wedge u\mathtt{1-0---}$.
3. $C[\overline{v}, \overline{u}] = v\mathtt{1000} \wedge u\mathtt{1-0---}$ (there is only one conjunct in $\mathrm{E}'[\overline{v}, \overline{u}]$).
4. The variable assignment in $C[\overline{v}, \overline{u}] = v\mathtt{1000} \wedge u_0 \wedge \neg u_2$ encodes the witness strategy f_x:

$f_x(s_{*\mathrm{B}}) = \bot$	$f_x(s_{2\mathrm{A}}) = 2 \quad [u_0]$
$f_x(s_{0*}) = \bot$	$f_x(s_{3\mathrm{A}}) = 1 \text{ or } 2 \quad [u_1 \vee \neg u_1]$
$f_x(s_{1\mathrm{A}}) = 1$	$f_x(s_{4\mathrm{A}}) = 1 \quad [\neg u_2]$

3.3.5. Complexity

As we have just shown, the symbolic encoding of the labelling algorithm has polynomial size and requires polynomially many Boolean variables. Therefore, the algorithm runs in *exponential time* with respect to both the size of the model $|\mathcal{I}|$ and the number of quantified variables $|\mathsf{vars}(\varphi)|$.

Theorem 3. *Let $\mathcal{I}$ be an arbitrary interpreted and $\varphi \in$ SLK be an SLK sentence. The worst case time complexity of the symbolic implementation of the model checking algorithm* $\mathtt{Check}_{\mathcal{I}}(\varphi, \emptyset)$ *is:*

$$O(|\varphi| \times |\mathrm{Ag}|) \times 2^{O(|\mathrm{St}| \times |\mathsf{vars}(\varphi)| \times \log_2 |\mathrm{Ac}|)}$$

[8]If no such function is available, we can skip step 3 ($C[\overline{v}, \overline{u}] := \mathrm{E}'[\overline{v}, \overline{u}]$) and refine the conjunct upon each lookup in step 4 ($C[\overline{v}, \overline{u}] := C[\overline{v}, \overline{u}] \wedge c[\overline{u}_{s', x_i}]$).

where Ag, St *and* Ac *are the sets of agents, reachable global states and total actions of* $\mathcal{I}$, *respectively.*

Proof (Sketch). The symbolic implementation uses:

- $O(\log_2 |\mathrm{St}|)$ Boolean variables to represent the *current global state*;
- $O(\log_2 |\mathrm{St}|)$ Boolean variables to represent the *next global state*;
- $O(\log_2 |\mathrm{Ac}|)$ Boolean variables to represent the *decisions*;
- $O(|\mathrm{St}| \times |\mathsf{vars}(\varphi)| \times \log_2 |\mathrm{Ac}|)$ Boolean variables to represent the *variable assignment.*

The total number of Boolean variables is thus $O(|\mathrm{St}| \times |\mathsf{vars}(\varphi)| \times \log_2 |\mathrm{Ac}|)$. Consequently, any BDDs built on top of these variables will have at most $2^{O(|\mathrm{St}| \times |\mathsf{vars}(\varphi)| \times \log_2 |\mathrm{Ac}|)}$ nodes. Since individual BDD operations take polynomial time with respect to the size of the relevant BDDs [41], the worst case time complexity of each BDD operation is $2^{O(|\mathrm{St}| \times |\mathsf{vars}(\varphi)| \times \log_2 |\mathrm{Ac}|)}$. As the recursive algorithm presented in Definition 13 will perform at most

$$\underbrace{O(|\varphi| \times |\mathrm{St}|)}_{\mathtt{Check}_{\mathcal{I}} \text{ calls}} \times \underbrace{O(|\mathrm{St}| \times |\mathrm{Ag}| \times |\mathrm{Ac}|)}_{\text{size of } R^b_t \text{ and } R^{\mathsf{K}}_a \text{ defs}} \times \underbrace{O(|\mathrm{St}| \times |\mathsf{vars}(\varphi)| \times \log_2 |\mathrm{Ac}|)}_{\text{Boolean variables}}$$

BDD operations, our claim follows. □

The proof shows that the complexity of the algorithm is dominated by the encoding of the variable assignments.

Observe that the symbolic implementation of the algorithm $\mathtt{Check}_{\mathcal{I}}$ can use more than a polynomial amount of space because it performs operations on sets of extended states. While the procedure described in the proof of Theorem 1 uses only a polynomial amount of space (as it operates on individual states and assignments), we believe it is unlikely that such an explicit approach would outperform the symbolic algorithm in practice [42].

4. Implementation and Experimental Results

The model checker $\mathrm{MCMAS}_{\mathrm{SLK}}$ [43], first introduced in [24], implements the state labelling algorithm presented in Section 3. The tool is based on the existing open-source model checker MCMAS [20], which supports the verification of CTL and ATL formulas with fairness constraints and epistemic and deontic modalities. Both checkers are written in C++ and based on the CUDD BDD package [44].

$\mathrm{MCMAS}_{\mathrm{SLK}}$ takes as input an ISPL file [45] containing a description of an interpreted system (see Definition 4) and a list of SLK specifications. For each specification, the tool calculates the associated set of reachable extended states, encoded as BDDs, and verifies the specification using the symbolic implementation of the state labelling algorithm presented in Section 3. If requested, the checker also generates witnesses, counterexamples and strategies for each specification (see Definition 14). The latter is an enhancement of MCMAS as it provides the means to automatically synthesise agents' behaviour satisfying an SLK specification.

The usage of the SLK extension is the same as that of the original tool. Given an ISPL file `system.ispl`, the following command checks the SLK specifications against the interpreted system:

```
$ ./mcmas system.ispl
```

A witness/counterexample execution with strategies can be requested (where possible):

```
$ ./mcmas -c 1 system.ispl
```

We present here the experimental results obtained using $\text{MCMAS}_{\text{SLK}}$ on several scalable real-life scenarios. The experiments were run on an Intel Core i7-3770 CPU 3.40GHz machine with 16GB RAM running Linux kernel version `3.8.0-35-generic`. We measure the amount of time and memory used by the tool and compare it with the performance of the original tool, MCMAS, on CTL and ATL (where possible). The examples in this section also demonstrate the expressiveness of SLK.

4.1. Dining Cryptographers

The dining cryptographers protocol [46, 45] is a commonly studied anonymity protocol. It is suited as a case study as anonymity can naturally be expressed as lack of knowledge [47]. We model the protocol with $n \geq 3$ cryptographers as an interpreted system (see Definition 4) with agents $\text{Ag} = \{\text{Env}, c_1, \dots, c_n\}$. Table 1 reports the results obtained when verifying the dining cryptographers protocol against the following CTLK, ATLK and SLK specifications:

$$\varphi_{\text{CTLK}} = \mathsf{AG}\,\psi \tag{1}$$

$$\varphi_{\text{ATLK}} = \langle\!\langle \emptyset \rangle\!\rangle \mathsf{G}\,\psi \tag{2}$$

$$\varphi_{\text{SLK}} = [\![x_e]\!][\![x_1]\!]\cdots[\![x_n]\!](\text{Env}, x_e)(c_1, x_1)\cdots(c_n, x_n)\mathsf{G}\,\psi$$

where:

$$\psi = (\mathit{odd} \wedge \neg \mathit{paid}_1) \rightarrow \underbrace{\left[\mathsf{K}_{c_1} \bigvee_{i=2}^{n} \mathit{paid}_i\right]}_{\substack{\text{cryptographer } c_1 \\ \textit{knows} \text{ that another} \\ \text{cryptographer paid}}} \wedge \underbrace{\left[\bigwedge_{i=2}^{n} \neg \mathsf{K}_{c_1}\, \mathit{paid}_i\right]}_{\substack{\text{cryptographer } c_1 \\ \textit{does not know} \text{ which} \\ \text{cryptographer paid}}}$$

φ_{CTLK} and φ_{ATLK} are the usual epistemic specifications for the protocol [48, 45] and φ_{SLK} is its natural extension where strategies are quantified. The results indicate that the checker can verify reasonably large state spaces. The performance depends on the number of Boolean variables required to represent the extended states. In the case of Slk specifications, the number of Boolean variables is proportional to the number of states. The last six columns of Table 1 show that the tool's performance drops considerably faster when verifying Slk formulas compared to Ctlk and Atlk ones. This is because neither Ctlk nor Atlk verification requires assignments; hence extended states collapse to plain states. In contrast, both Ctlk and Atlk performance is dominated by the reachable state space computation. We found that splitting the extended state space generation across multiple threads has almost negligible impact on performance; this is due to the fact that the CUDD BDD package used by MCMAS does not support concurrency.

4.2. Cake Cutting

The cake-cutting problem, a well-known mathematical puzzle in which n agents take turns to slice a cake of size d and the environment responds by trying to ensure the cake is divided fairly, was already described in Section 2 together with the Slk specification $\varphi = \langle\!\langle x \rangle\!\rangle\, (\varphi_{\mathrm{F}} \wedge \varphi_{\mathrm{S}})$ asserting the existence of a solution. This example demonstrates not only the ability of Slk to express *Nash equilibria*, but also the fact that the checker can synthesise protocols which achieve them.

We were able to verify the formula φ defined above on a system with $n = 2$ agents and a cake of size $d = 2$. Moreover, the checker automatically synthesised a witness strategy f_x for the environment as well as the Nash equilibrium (f_{y_1}, f_{y_2}) for the agents. For more details, see [49, Subsection 6.4.2].

We were unable to verify larger examples due to out of memory errors. For example, with $n = 2$ agents and $d = 3$ slices, there are 29 reachable states and the encoding requires 105 Boolean variables, 54 of which represent the strategy f_x we wish to synthesise. The intermediate BDDs represent 8.09×10^{20} possible *extended* states using in the order of 10^9 nodes. A single intermediate BDD thus requires approximately 32GB of memory and the whole algorithm would need to use over 10^{11} bytes to complete the calculation. This is to be expected given the theoretical difficulty of the problem.

4.3. Scheduler

Lastly, we consider a *preemptive scheduler system* [49, Subsection 6.4.3] composed of $n \in \mathbb{N}$ processes and an arbiter which ensures mutual exclusion of a shared resource while preventing starvation, to compare the performance of Ctl, Atl and Slk model checking on safety and fairness properties.

The desired properties of the system can be expressed using the following Slk specifications:

1. **Mutual exclusion.** The following specification asserts that at most one process owns the resource at any given point in time:

$$\varphi_{\mathrm{ME}} = [\![x]\!][\![y_1]\!] \cdots [\![y_n]\!](\mathrm{Env}, x)(1, y_1) \cdots (n, y_n)\, \mathsf{G} \neg \bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} \langle \mathrm{rs}, i \rangle \wedge \langle \mathrm{rs}, j \rangle$$

crypts n	**possible states**	**reachable states**	**reachability time** (s)	CTLK		ATLK		SLK					
								1 thread		2 threads		4 threads	
				time (s)	**mem** (MB)	**time** (s)	**mem** (MB)	**time** (s)	**mem** (MB)	**time** (s)	**mem** (MB)	**time** (s)	**mem** (MB)
3	82944	128	0.00	0.00	2.69	0.00	2.69	0.00	2.79	0.01	4.91	0.02	9.23
4	1.99×10^6	320	0.02	0.00	2.71	0.00	2.70	0.00	2.91	0.02	4.97	0.04	9.24
5	4.78×10^7	768	0.04	0.00	2.75	0.00	2.75	0.01	3.07	0.03	5.10	0.05	9.67
6	1.15×10^9	1792	0.08	0.00	2.79	0.00	2.77	0.02	3.41	0.04	5.37	0.07	9.94
7	2.75×10^{10}	4096	0.21	0.00	3.03	0.00	2.97	0.05	4.47	0.08	5.88	0.12	10.71
8	6.60×10^{11}	9216	0.40	0.00	3.19	0.00	3.27	0.18	6.37	0.21	7.42	0.24	12.56
9	1.58×10^{13}	20480	0.43	0.00	3.38	0.00	3.42	0.38	9.59	0.38	9.82	0.46	15.96
10	3.80×10^{14}	45056	4.19	0.28	13.02	0.16	12.75	2.29	18.62	2.23	22.33	2.43	28.21
11	9.13×10^{15}	98304	1.69	0.04	5.49	0.01	4.75	5.01	23.55	4.37	27.51	4.77	39.59
12	2.19×10^{17}	212992	2.32	0.01	4.40	0.01	4.20	11.61	39.25	9.61	40.36	10.32	60.82
13	5.26×10^{18}	458752	2.05	0.10	6.50	0.03	5.62	32.63	81.95	26.30	79.75	28.39	118.27
14	1.26×10^{20}	983040	1.96	0.08	6.87	0.03	5.00	89.46	169.95	67.99	169.68	73.57	243.80
15	3.03×10^{21}	2.10×10^6	21.35	0.35	12.73	0.08	9.29	163.77	363.66	139.42	354.18	156.57	500.96
16	7.27×10^{22}	4.46×10^6	6.66	0.09	6.65	0.04	4.83	422.03	758.43	345.83	736.02	377.50	977.79
17	1.74×10^{24}	9.44×10^6	9.10	0.13	6.67	0.08	6.04	734.44	1360.59	643.03	1270.93	705.44	1359.73
18	4.19×10^{25}	1.99×10^7	65.94	0.50	12.66	0.14	12.67	2654.46	3550.66	2129.54	3680.89	*process killed*	

Table 1: Experimental model checking results for the dining cryptographers protocol.

This property can be equivalently expressed in CTL and ATL using the AG and $\langle\!\langle\emptyset\rangle\!\rangle\mathsf{G}$ operators, respectively (see Equations 1 and 2). The results in Table 2 show that the equivalent CTL and ATL formulas are checked almost instantaneously. Similarly to the dining cryptographers scenario (see Subsection 4.1), the SLK model checking performance drops quickly for $n \geq 8$ processes.

2. **Absence of starvation.** The following specification asserts the existence of an arbiter strategy which ensures that every request is eventually satisfied:

$$\varphi_{\mathrm{AS}} = \langle\!\langle x\rangle\!\rangle[\![y_1]\!]\ldots[\![y_n]\!](\mathrm{Env}, x)(1, y_1)\cdots(n, y_n)\,\mathsf{G}\bigwedge_{i=1}^{n}(\langle \mathrm{wt}, i\rangle \to \mathsf{F}\,\neg\langle \mathrm{wt}, i\rangle)$$

This property cannot be expressed in CTL because it refers to the capability of an agent to enforce a property. Moreover, it cannot be expressed in ATL either, because it contains nested temporal operators (G and F) which depend on the same strategies. The model checking results for the formulas with $2 \leq n \leq 3$ processes are shown in Table 2. In this case MCMAS$_{\mathrm{SLK}}$ ran out of physical memory for $n > 3$ processes.

4.4. Analysis

In general, MCMAS$_{\mathrm{SLK}}$ has significantly lower performance than the original tool on CTL and ATL. We expected this outcome given the high model checking complexity of SLK. However, the checker can verify specifications which cannot be checked by the original tool. We have discussed two examples of such properties: *(i)* Nash equilibria (see Subsection 4.2) and *(ii)* fairness conditions (see Subsection 4.3). Moreover, the checker supports witness strategy synthesis, which can be used to automatically generate agents' behaviour. In other words, the aim of MCMAS$_{\mathrm{SLK}}$ is not to replace MCMAS, but to add support for completely new features. Therefore, the results presented in this section should be regarded as an illustration of the new functionality.

The experimental results presented in this section confirm that the main performance bottleneck of MCMAS$_{\mathrm{SLK}}$ is the BDD encoding of the extended states, which allocates separate BDD variables for each shared local state of a strategy (see Subsection 3.3). Nevertheless, the checker can handle reasonably large state spaces for certain specifications as we have seen in the dining cryptographers scenario (see Subsection 4.1). We were able to slightly improve the tool's performance by using multiple threads (see Table 1) despite the fact that the CUDD BDD package used by MCMAS has no built-in support for concurrency. This suggests that further speedup could be achieved by using a different BDD package with a higher degree of parallelism.

5. Conclusions and Related Work

As argued in the introduction, considerable progress has been made over the past 15 years in the development of methodologies for the verification of

procs n	**possible states**	**reach. states**	**reach. time** (s)	Mutual exclusion φ_{ME}						Absence of starvation φ_{AS}	
				Ctl		Atl		Slk		Slk	
				time (s)	**mem** (MB)	**time** (s)	**mem** (MB)	**time** (s)	**mem** (MB)	**time** (s)	**mem** (MB)
2	72	9	0.00	0.00	2.64	0.00	2.64	0.00	2.67	0.00	2.67
3	432	21	0.00	0.00	2.72	0.00	2.72	0.00	2.81	116.78	124.90
4	2592	49	0.01	0.00	2.81	0.00	2.81	0.00	3.11	*out of memory*	
5	15552	113	0.02	0.00	3.02	0.00	3.03	0.02	4.20	*out of memory*	
6	93312	257	0.02	0.00	3.40	0.00	3.40	0.11	8.98	*out of memory*	
7	559872	577	0.08	0.00	4.47	0.00	4.48	0.86	23.18	*out of memory*	
8	3.36×10^6	1281	0.17	0.00	6.73	0.00	6.72	4.88	91.26	*out of memory*	
9	2.02×10^7	2817	0.20	0.00	11.62	0.00	11.62	36.71	561.69	*out of memory*	
10	1.21×10^8	6145	0.10	0.00	22.23	0.00	22.23	222.36	2769.64	*out of memory*	

Table 2: Experimental model checking results for the preemptive scheduler system.

MAS against agent-based specifications, with particular emphasis on epistemic specifications. Statements capturing the strategic interplay between agents in a system have also been supported, but they have traditionally been limited to ATL specifications.

It has been argued that the intrinsic limitations of ATL with respect to direct naming and binding of strategies severely restrict the expressiveness of the specifications [50, 23, 51]. These limitations are particularly severe when reasoning about game theoretic concepts. In response to this criticism, *Strategy logic* (SL), among others, was put forward [23]. Sophisticated concepts such as Nash equilibria, which cannot be expressed by ATL, can naturally be encoded in SL.

Given this, the question that arises is whether *automatic and efficient* verification methodologies for MAS against SL specifications can be devised. To answer this question, we have introduced and studied SLK, a variant of epistemic logic incorporating some SL concepts defined on a memoryless variant of interpreted systems. We have shown that SLK admits an efficient labelling algorithm solving the associated model checking problem and introduced $\text{MCMAS}_{\text{SLK}}$, an extension of the model checker MCMAS [20], for the verification of MAS against SLK specifications.

A notable feature of $\text{MCMAS}_{\text{SLK}}$ is that it allows for automatic verification of game concepts such as various forms of equilibria, including Nash equilibria and subgame perfect Nash equilibria. Since SLK also supports a limited form of epistemic modalities, the proposed approach further enables us to express specifications concerning individual and group knowledge of cooperation properties. Furthermore, since SLK subsumes ATLK*, $\text{MCMAS}_{\text{SLK}}$ is also the first tool enabling the verification of multi-agent systems against specifications given in ATLK*.

Related Work. The present work builds on [19, 20] where a symbolic approach for the verification of a variant of ATL was put forward and implemented in MCMAS. While the present work reuses parts of the labelling algorithm and its implementation, it supports SL modalities which were not covered in [20].

A number of investigations have explored the complexity of the model checking problem for combinations of plain ATL and epistemic operators under a number of assumptions [15, 16, 17]. These identify lower and upper bounds for the verification problem; but no implementation is reported.

In [52, 53] a combination of epistemic logic and a logic supporting strategic concepts was introduced together with a BDD-based toolkit. Similarly to the present approach, [52] assumes incomplete information and memoryless strategies. This work, however, shares ATL's implicit notion of strategies. In contrast, here we follow SL closely and adopt an explicit notion of strategies which can be bound to and shared by different agents.

In [54] a verification methodology for a subset of SL, namely its one-goal fragment, was introduced. However the underlying assumptions are quite different as perfect recall and complete information is assumed. Furthermore, the logic does not support epistemic operators.

Future Work. The experiments that we reported show that the performance of MCMAS$_{\text{SLK}}$ on SLK is comparable to that of MCMAS on equivalent ATL and CTLK formulas. This is because we adopted an approach in which the colouring with strategies is specification-dependent and is only performed after the set of reachable states is computed. We found that the main impediment to better performance of the tool is the size of the BDDs required to encode sets of extended states. Future efforts will be devoted to mitigate this problem as well as to support logics stronger than SLK, including relaxing the assumption on epistemic sentences.

Acknowledgements. This research was partly funded by the EPSRC (grant EP/I00520X), the Regione Campania (Embedded System Cup project B25B09090 100007), the EU (FP7 project 600958-SHERPA), and the MIUR (ORCHESTRA project).

Bibliography

[1] M. Wooldridge, An introduction to MultiAgent systems, second edition Edition, Wiley, 2009.

[2] R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi, Reasoning about Knowledge, MIT Press, Cambridge, 1995.

[3] A. Lomuscio, M. Sergot, Deontic interpreted systems, Studia Logica 75 (1) (2003) 63–92.

[4] A. S. Rao, Decision procedures for propositional linear-time Belief-Desire-Intention logics, in: M. Wooldridge, J. P. Müller, M. Tambe (Eds.), Intelligent Agents II (LNAI 1037), Springer-Verlag: Heidelberg, Germany, 1996, pp. 33–48.

[5] A. Pnueli, Applications of temporal logic to the specification and verification of reactive systems: A survey of current trends., in: J. W. de Bakker, W. P. de Roever, G. Rozenberg (Eds.), Current Trends in Concurrency, Overviews and Tutorials, LNCS 224, Springer, 1986, pp. 510–584.

[6] W. Penczek, A. Lomuscio, Verifying epistemic properties of multi-agent systems via bounded model checking, Fundamenta Informaticae 55 (2) (2003) 167–185.

[7] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Wozna, A. Zbrzezny, Verics 2007 - a model checker for knowledge and real-time, Fundamenta Informaticae 85 (1-4) (2008) 313–328.

[8] R. Bryant, Graph-Based Algorithms for Boolean Function Manipulation., Transactions on Computers 35 (8) (1986) 677–691.

[9] P. Gammie, R. van der Meyden, MCK: Model checking the logic of knowledge, in: Proceedings of 16th International Conference on Computer Aided Verification (CAV'04), LNCS 3114, Springer, 2004, pp. 479–483.

[10] F. Raimondi, A. Lomuscio, Automatic verification of multi-agent systems by model checking via OBDDs, Journal of Applied Logic 5 (2) (2005) 235–251.

[11] M. Cohen, M. Dam, A. Lomuscio, H. Qu, A symmetry reduction technique for model checking temporal-epistemic logic, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09), Pasadena, USA, 2009, pp. 721–726.

[12] M. Cohen, M. Dam, A. Lomuscio, F. Russo, Abstraction in model checking multi-agent systems, in: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09), IFAAMAS Press, Budapest, Hungary, 2009, pp. 945–952.

[13] A. Lomuscio, J. Michaliszyn, Verification of multi-agent systems via predicate abstraction against ATLK specifications, in: Proc. of the 15th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS'16), 2016, pp. 662–670.

[14] R. Alur, L. de Alfaro, R. Grosu, T. Henzinger, A. Thomas, M. Kang, C. Kirsch, R. Majumdar, F. Mang, B.-Y. Wang, jMocha: A model checking tool that exploits design structure, in: Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), IEEE, 2001, pp. 835–836.

[15] W. Hoek, M. Wooldridge, Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications, Studia Logica 75 (1) (2003) 125–157.

[16] W. Jamroga, Some remarks on alternating temporal epistemic logic, in: B. Dunin-Kęplicz, R. Verbrugge (Eds.), Proceedings of the International Workshop on Formal Approaches to Multi-Agent Systems (FAMAS'03), 2004, pp. 133–140.

[17] T. Ågotnes, V. Goranko, W. Jamroga, M. Wooldridge, Knowledge and ability, in: Handbook of Logics for Knowledge and Belief, College Publications, 2015.

[18] N. Bulling, W. Jamroga, Rational play and rational beliefs under uncertainty, in: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'16), IFAAMAS, 2009, pp. 257–264.

[19] A. Lomuscio, F. Raimondi, Model checking knowledge, strategies, and games in multi-agent systems, in: Proceedings of the 5th International Joint Conference on Autonomous agents and Multi-Agent Systems (AAMAS'06), ACM Press, 2006, pp. 161–168.

[20] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: A model checker for the verification of multi-agent systems, Software Tools for Technology Transfer.

[21] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, Journal of the ACM 49 (5) (2002) 672–713.

[22] G. Jonker, Feasible strategies in alternating-time temporal epistemic logic, Master's thesis, University of Utrech, The Netherlands (2003).

[23] F. Mogavero, A. Murano, M. Vardi, Reasoning About Strategies., in: Foundations of Software Technology and Theoretical Computer Science'10, LIPIcs 8, Leibniz-Zentrum fuer Informatik, 2010, pp. 133–144.

[24] P. Čermák, A. Lomuscio, F. Mogavero, A. Murano, MCMAS-SLK: A model checker for the verification of strategy logic specifications, in: Proceedings of the 26th International Conference n Computer Aided Verification (CAV'14), LNCS 8559, Springer, 2014, pp. 525–532.

[25] K. Chatterjee, T. Henzinger, N. Piterman, Strategy Logic., Information and Computation 208 (6) (2010) 677–693.

[26] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning About Strategies: On the Model-Checking Problem., Transactions On Computational Logic 15 (4) (2014) 34:1–42.

[27] R. Alur, T. Henzinger, O. Kupferman, Alternating-Time Temporal Logic., Journal of the ACM 49 (5) (2002) 672–713.

[28] R. Fagin, J. Halpern, Y. Moses, M. Vardi, Reasoning about Knowledge., MIT Press, 1995.

[29] A. Pnueli, The Temporal Logic of Programs., in: Foundation of Computer Science'77, IEEE Computer Society, 1977, pp. 46–57.

[30] A. Lomuscio, R. Meyden, M. Ryan, Knowledge in multi-agent systems: Initial configurations and broadcast, ACM Transactions of Computational Logic 1 (2) (2000) 246–282.

[31] A. Lomuscio, Knowledge sharing among ideal agents, Ph.D. thesis, School of Computer Science, University of Birmingham, Birmingham, UK (Jun. 1999).

[32] S. Even, A. Paz, A Note on Cake Cutting., Discrete Applied Mathematics 7 (1984) 285–296.

[33] O. Kupferman, M. Vardi, P. Wolper, An Automata Theoretic Approach to Branching-Time Model Checking., Journal of the ACM 47 (2) (2000) 312–360.

[34] M. Vardi, P. Wolper, An Automata-Theoretic Approach to Automatic Program Verification., in: Logic in Computer Science'86, IEEE Computer Society, 1986, pp. 332–344.

[35] N. Bulling, J. Dix, W. Jamroga, Model Checking Logics of Strategic Ability: Complexity., in: Specification and Verification of Multi-Agent Systems'10, Springer, 2010, pp. 125–159.

[36] L. Stockmeyer, A. Meyer, Word Problems Requiring Exponential Time (Preliminary Report)., in: Symposium on Theory of Computing'73, Association for Computing Machinery, 1973, pp. 1–9.

[37] M. Benerecetti, F. Mogavero, A. Murano, Reasoning About Substructures and Games., Transactions On Computational Logic 16 (3) (2015) 25:1–46.

[38] F. Mogavero, A. Murano, L. Sauro, On the boundary of behavioral strategies, in: Proceedings of the 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS'13), IEEE, 2013, pp. 263–272.

[39] A. Lomuscio, F. Raimondi, Model checking knowledge, strategies, and games in multi-agent systems, in: Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems (AAMAS'06), ACM Press, 2006, pp. 161–168.

[40] A. Morgenstern, Symbolic controller synthesis for LTL specifications, Ph.D. thesis (2010).

[41] M. Huth, M. Ryan, Logic in Computer Science: Modelling and Reasoning about Systems, Cambridge University Press, New York, NY, USA, 2004.

[42] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L. J. Hwang, Symbolic model checking: 10^{20} states and beyond, Information and Computation 98 (2) (1992) 142–170.

[43] MCMAS-SLK - A Model Checker for the Verification of Strategy Logic Specifications., http://vas.doc.ic.ac.uk/software/tools/.

[44] F. Somenzi, CUDD: CU decision diagram package, `http://vlsi.colorado.edu/~fabio/CUDD/`.

[45] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: A model checker for the verification of multi-agent systems, in: Proceedings of the 21th International Conference on Computer Aided Verification (CAV'09), LNCS 5643, Springer, 2009, pp. 682–688.

[46] D. Chaum, The dining cryptographers problem: Unconditional sender and recipient untraceability, Journal of Cryptology 1 (1) (1988) 65–75.

[47] W. van der Hoek, A. Lomuscio, A logic for ignorance, Electronic Notes in Theoretical Computer Science 85 (2) (2004) 117–133.

[48] R. van der Meyden, K. Su, Symbolic model checking the knowledge of the dining cryptographers, in: Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04), IEEE Computer Society, Washington, DC, USA, 2004, pp. 280–291.

[49] P. Čermák, A model checker for strategy logic, Master's thesis, Department of Computing, Imperial College London, UK (2014).

[50] K. Chatterjee, T. A. Henzinger, N. Piterman, Strategy logic, Inf. Comput. 208 (6) (2010) 677–693.

[51] A. D. C. Lopes, F. Laroussinie, N. Markey, ATL with strategy contexts: Expressiveness and model checking, in: Proceedings of the 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10), LIPIcs 8, 2010, pp. 120–132.

[52] X. Huang, R. v. Meyden, Symbolic model checking epistemic strategy logic, in: Proceedings of the 28th Conference on Artificial Intelligence (AAAI'14), AAAI, 2014, pp. 1426–1432.

[53] X. Huang, R. v. Meyden, A temporal logic of strategic knowledge, in: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14), AAAI, 2014, pp. 418–427.

[54] P. Čermák, A. Lomuscio, A. Murano, Verifying and synthesising multi-agent systems against one-goal strategy logic specifications, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15), AAAI Press, 2015, pp. 2038–2044.