Verifying and Synthesising Multi-Agent Systems against One-Goal Strategy Logic Specifications

Petr Čermák and **Alessio Lomuscio**
Imperial College London, UK

Aniello Murano
Università degli Studi di Napoli Federico II, Italy

Abstract

Strategy Logic (SL) has recently come to the fore as a useful specification language to reason about multi-agent systems. Its one-goal fragment, or SL[1G], is of particular interest as it strictly subsumes widely used logics such as ATL*, while maintaining attractive complexity features. In this paper we put forward an automata-based methodology for verifying and synthesising multi-agent systems against specifications given in SL[1G]. We show that the algorithm is sound and optimal from a computational point of view. A key feature of the approach is that all data structures and operations on them can be performed on BDDs. We report on a BDD-based model checker implementing the algorithm and evaluate its performance against a number of scalable scenarios.

1 Introduction

A concern in the deployment of autonomous multi-agent systems (MAS) is the limited availability of efficient techniques and toolkits for their verification. The problem is compounded by the fact that MAS require ad-hoc techniques and tools. This is because, while reactive systems are typically specified purely by temporal properties, MAS are instead described by statements expressing a number of typical AI concepts including knowledge, beliefs, intentions, and abilities.

Some progress in this direction has been achieved in the past decade. For example, several efficient techniques are now available for the verification of MAS against temporal-epistemic languages (Meyden and Shilov 1999; Raimondi and Lomuscio 2005; Kouvaros and Lomuscio 2013; Meski et al. 2014). Some of these have been implemented into fully-fledged model checkers (Gammie and van der Meyden 2004; Kacprzak et al. 2008; Lomuscio, Qu, and Raimondi 2009).

Less attention has so far been given to the verification of properties expressing cooperation and enforcement (Lomuscio and Raimondi 2006; Kacprzak and Penczek 2005). While the underlying logics have been thoroughly investigated at theoretical level (Bulling and Jamroga 2014), tool support is more sketchy and typically limited to alternating-time temporal logic (ATL) (Alur, Henzinger, and Kupferman 2002). A number of recent papers (Chatterjee, Henzinger, and Piterman 2010; Mogavero, Murano, and Vardi 2010) have however pointed out significant limitations of ATL when used in a MAS setting. One of these is the syntactic impossibility of referring explicitly to what particular strategies a group of agents ought to use when evaluating the realisability of temporal properties in a MAS. Being able to do so would enable us to express typical MAS properties, including strategic game-theoretic considerations for a group of agents in a cooperative or adversarial setting.

In response to this shortcoming, *Strategy logic* (SL), a strict extension of any logic in the ATL hierarchy, has recently been put forward (Mogavero, Murano, and Vardi 2010). In SL, strategies are explicitly referred to by using first-order quantifiers and bindings to agents. Sophisticated concepts such as Nash equilibria, which cannot be expressed in ATL, can naturally be encoded in SL.

Given this, a natural and compelling question that arises is whether *automatic and efficient* verification methodologies for MAS against SL specifications can be devised. The answer to this is negative in general: model checking systems against SL specifications is NONELEMENTARYSPACE-HARD (Mogavero et al. 2014), thereby hindering any concrete application on large systems. It is therefore of interest to investigate whether computationally attractive methodologies can be put forward for fragments of SL. The only contributions we are aware of in this regard are (Čermák et al. 2014; Huang and Meyden 2014), where model checking MAS against a *memoryless* fragment of SL combined with epistemic modalities was studied. Although a tool was released, memoryless strategies severely constrain the expressivity of the formalism.

To overcome this difficulty, we here put forward a technique for the verification and synthesis of MAS against specifications given in *One-Goal Strategy Logic*, or SL[1G], an expressive variant of SL. We claim there are several advantages of choosing this setting. Firstly, and differently from full SL, strategies in SL[1G] are *behavioural* (Mogavero et al. 2014). A consequence of this is that they can be synthesised automatically, as we show later. Secondly, SL[1G], in the perfect recall setting we here consider, retains considerable expressiveness and is strictly more expressive than any ATL variant, including ATL*. Thirdly, the complexity of the model checking problem is the same as that for ATL*, thereby making its verification attractive.

The rest of the paper is organised as follows. In Section 2 we recall the logic SL[1G], introduce the model checking and synthesis problems and a few related concepts. In Section 3 we put forward practical algorithms for the model checking and synthesis of MAS against SL[1G] specifications. We also show that these are provably optimal when considered against the theoretical complexity known for the problem. In Section 4 we show that the algorithms are amenable to symbolic implementation with binary-decision diagrams and present an experimental model checker implementing the algorithms discussed. We evaluate its performance on the fair process scheduler synthesis. We conclude in Section 5 where we also point to further work.

2 One-Goal Strategy Logic

In this section we introduce some basic concepts and recall SL[1G], a syntactic fragment of SL, introduced in (Mogavero et al. 2014).

Underlying Framework. Differently from other treatments of SL, which were originally defined on concurrent game structures, we here use *interpreted systems*, which are commonly used to reason about knowledge and strategies in multi-agent systems (Fagin et al. 1995; Lomuscio and Raimondi 2006). An interpreted system is a tuple $\mathcal{I} = \langle (L_i, Act_i, P_i, t_i)_{i \in Agt}, I, h \rangle$, where each agent $i \in Agt$ is modelled in terms of its set of *local states* L_i, set of *actions* Act_i, *protocol* $P_i : L_i \to 2^{Act_i}$ specifying what actions can be performed at a given local state, and *evolution function* $t_i : L_i \times Act \to L_i$ returning the next local state given the current local state and a joint action for all agents.

The set of *global states* G of the whole system consists of tuples of local states for all agents. As a special subset G contains the set I of *initial global* states. The labelling function h maps each *atomic proposition* $p \in AP$ to the set of global states $h(p) \subseteq G$ in which it is true. *Joint actions* Act are tuples of local actions for all the agents in the system; *shared actions* in the set $Act_A \triangleq \bigcap_{i \in A} Act_i$ are actions for the agents $A \subseteq Agt$; The *global protocol* $P : G \to 2^{Act}$ and *global evolution function* $t : G \times Act \to G$, which are composed of their local counterparts P_i and t_i, complete the description of the evolution of the entire system.

Syntax of SL[1G]. SL has been introduced as a powerful formalism to reason about sophisticated cooperation concepts in multi-agent systems (Mogavero, Murano, and Vardi 2010). Formally, it is defined as a syntactic extension of the logic LTL by means of an *existential strategy quantifier* $\langle\!\langle x \rangle\!\rangle \varphi$, a *universal strategy quantifier* $[\![x]\!] \varphi$, and an *agent binding* operator $(a, x)\varphi$. Intuitively, $\langle\!\langle x \rangle\!\rangle \varphi$ is read as *"there exists a strategy x such that φ holds"*, $[\![x]\!] \varphi$ is its dual, and $(a, x)\varphi$ stands for *"bind agent a to the strategy associated with the variable x in φ"*. In SL[1G], these three new constructs are merged into one rule $\wp \flat \varphi$, where $\wp$ is a quantification prefix over strategies (e.g. $[\![x]\!] \langle\!\langle y \rangle\!\rangle [\![z]\!]$) and $\flat$ is a binding prefix (e.g. $(a, x)(b, y)(c, x)$). As this limits the use of strategy quantifiers and bindings in SL, SL[1G] is less expressive than SL (Mogavero et al. 2014). Nevertheless, it still strictly subsumes commonly considered logics for strategic reasoning such as ATL*. Additionally, several attractive features of ATL* hold in SL[1G], including the fact that satisfiability and model checking are 2EXPTIME-COMPLETE (Mogavero et al. 2014). Crucially, SL[1G] can be said to be *behavioural*, that is the choice of a strategy for a group of agents at a given state depends only on the history of the game and the actions performed by other agents. This is in contrast with the *non-behavioural* aspects of SL in which strategy choices depend on other agents' actions in the future or in counterfactual games. In summary, SL[1G] is strictly more expressive than ATL*, whereas it retains ATL*'s elementary complexity of key decision problems, including the strategy synthesis problem.

To define formally the syntax of SL[1G], we first introduce the concepts of a quantification and binding prefix (Mogavero et al. 2014).

A *quantification prefix* over a set of variables $V \subseteq Var$ is a finite word $\wp \in \{\langle\!\langle x \rangle\!\rangle, [\![x]\!] \mid x \in V\}^{|V|}$ of length $|V|$ such that each variable $x \in V$ occurs in $\wp$ exactly once. $QPre_V$ denotes the set of all quantification prefixes over V. $QPre = \bigcup_{V \subseteq Var} QPre_V$ is the set of all quantification prefixes. A *binding prefix* over a set of variables $V \subseteq Var$ is a finite word $\flat \in \{(i, x) \mid i \in Agt \wedge x \in V\}^{|Agt|}$ of length $|Agt|$ such that each agent $i \in Agt$ occurs in $\flat$ exactly once. $BPre_V$ denotes the set of all binding prefixes over V. $BPre = \bigcup_{V \subseteq Var} BPre_V$ is the set of all binding prefixes. Similarly to first-order languages, we also use $\mathsf{free}(\varphi)$ to represent the *free agents and variables* in a formula φ. Formally, $\mathsf{free}(\varphi) \subseteq Agt \cup Var$ contains *(i)* all agents having no binding after the occurrence of a temporal operator and *(ii)* all variables having a binding but no quantification.

Definition 1 (SL[1G] Syntax)**.** SL[1G] formulas *are built inductively from the set of atomic propositions AP, strategy variables Var, agents Agt, quantification prefixes $QPre$, and binding prefixes $BPre$, by using the following grammar, with $p \in AP$, $x \in Var$, $a \in Agt$, $\flat \in BPre$, and $\wp \in QPre$:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \wp\flat\varphi$$

where $\wp \in QPre_{\mathsf{free}(\flat\varphi)}$.

The conditions on $\wp$ and $\flat$ ensure that $\wp\flat\varphi$ is an SL[1G] sentence, i.e. , it does not have any free agents or variables.

Semantics of SL[1G]. We assume perfect recall and complete information. So agents have full memory of the past and complete information of the global state they are in. Note that allowing incomplete information would make the logic undecidable (Dima and Tiplea 2011), whereas SL[1G] with incomplete information and imperfect recall is equivalent to a fragment of the logic SLK already studied in (Čermák et al. 2014).

To establish the truth of a formula, the set of strategies over which a variable can range needs to be determined. For this purpose we introduce the set $\mathsf{sharing}(\varphi, x)$ representing the agents sharing the variable x within the formula φ. Also, we make use of the general concepts of *path*, *track*, *play*, *strategy*, and *assignment* for agents and variables. We refer to (Mogavero et al. 2014) for a detailed presentation. Intuitively, a strategy identifies paths in the model on which a formula needs to be checked. So, for each track (i.e. a finite prefix of a path), a strategy determines which action has

to be performed by a variable, possibly shared by a set of agents. More formally, given an SL[1G] formula φ, for each variable x in φ, the strategy $f : Trk \rightharpoonup Act_{\mathsf{sharing}(\varphi,x)}$ determines the action to be taken by agents in $\mathsf{sharing}(\varphi, x)$.

Given an interpreted system $\mathcal{I}$ having a set of global states G, a global state $g \in G$, and an assignment χ defined on $\mathsf{free}(\varphi)$, we write $\mathcal{I}, \chi, g \models \varphi$ to represent that the SL[1G] formula φ holds at g in $\mathcal{I}$ under χ. The satisfaction relation for SL[1G] formulas is inductively defined by using the usual LTL interpretation for the atomic propositions, the Boolean connectives $\neg$ and $\wedge$, as well as the temporal operators X, F, G, and U. The inductive cases for the strategy quantification $\langle\!\langle x \rangle\!\rangle$ and the agent binding (a, x) are given as follows. The cases for universal quantification $[\![x]\!]$ are omitted as they can be given as the dual of the existential ones.

- $\mathcal{I}, \chi, g \models \langle\!\langle x \rangle\!\rangle \varphi$ iff there is a strategy f for the agents in $\mathsf{sharing}(\varphi, x)$ such that $\mathcal{I}, \chi[x \mapsto f], g \models \varphi$ where $\chi[x \mapsto f]$ is the assignment equal to χ except for the variable x, for which it assumes the value f.
- $\mathcal{I}, \chi, g \models (x, a)\varphi$ iff $\mathcal{I}, \chi[a \mapsto \chi(x)], g \models \varphi$, where $\chi[a \mapsto \chi(x)]$ denotes the assignment χ in which agent a is bound to the strategy $\chi(x)$.

Model Checking and Strategy Synthesis. The model checking problem is about deciding whether an SL[1G] formula holds in a certain model. Precisely, given an interpreted system $\mathcal{I}$, an initial global state g_0, an SL[1G] formula φ and an assignment χ defined on $\mathsf{free}(\varphi)$, the *model checking problem* concerns determining whether $\mathcal{I}, \chi, g_0 \models \varphi$.

Synthesis can be further used as a witness for the model checking problem as it allows to construct the strategies the agents need to perform to make the formula true. This amounts to deciding which action has to be taken by each shared variable. More formally, let $\mathcal{I}$ be an interpreted system and φ an SL[1G] formula. W.l.o.g., assume φ to be a so called *principal sentence*[1] of the form $\wp \flat \psi$, with $\wp \in QPre_{\mathsf{free}(\flat\psi)}$, $\wp = \wp(0) \cdot \wp(1) \cdots \wp(|\wp| - 1)$, and $\flat \in BPre$. Additionally assume that there exists an integer $0 \leq k < |\wp|$ such that for each $0 \leq j < k$ there exists a strategy f_j for variable $\wp_v(j)$ shared by agents in $\mathsf{sharing}(\flat\psi, \wp_v(j))$. Then, *strategy synthesis* amounts to defining the strategy $f_k : Trk \rightharpoonup Act_{\mathsf{sharing}(\flat\psi, \wp_v(k))}$ for variable $\wp_v(k)$ such that if $\mathcal{I}, \chi, g \models \wp_{\geq k} \flat \psi$, then $\mathcal{I}, \chi[\wp_v(k) \mapsto f_k], g \models \wp_{>k} \flat \psi$, where χ is an assignment defined on $\{\wp_v(j) \mid 0 \leq j < k\}$ such that for all $0 \leq j < k$ we have $\chi(\wp_v(j)) \triangleq f_j$, $\wp_{\geq k} \triangleq \wp(k) \cdots \wp(|\wp| - 1)$, and $\wp_{>k} \triangleq \wp(k+1) \cdots \wp(|\wp| - 1)$.

3 Symbolic Model Checking SL[1G]

We now introduce a novel algorithm for model checking an interpreted system $\mathcal{I}$ against an arbitrary SL[1G] sentence φ. For simplicity we assume that φ is a *principal sentence* of the form $\wp \flat \psi$.

Our aim is to find the set of all global reachable states $\|\varphi\|_{\mathcal{I}} \subseteq G$ at which the SL[1G] sentence φ holds, i.e. $\|\varphi\|_{\mathcal{I}} \triangleq \{g \in G \mid \mathcal{I}, \emptyset, g \models \varphi\}$. We proceed in a recursive manner over the structure of φ: According to SL[1G] syntax, ψ is a formula which combines atoms AP and *direct principal subsentences* of the form $\varphi' = \wp' \flat' \psi'$ using only Boolean and temporal connectives. Since φ' is also an SL[1G] principal sentence, we can recursively calculate $\|\varphi'\|_{\mathcal{I}}$; then replace φ' in φ with a new atom $p_{\varphi'} \in AP$; and finally update the assignment with $h(p_{\varphi'}) \triangleq \|\varphi'\|_{\mathcal{I}}$. This allows us to consider the simpler problem of model checking an SL[1G] *basic principal sentence* $\varphi = \wp \flat \psi$ where ψ is an LTL formula. Our general procedure is as follows:

1. We construct a *deterministic parity automaton* $\mathfrak{P}^{\psi}_{\mathcal{I}}$ equivalent to the LTL formula ψ.
2. We construct a two-player *formula arena* $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$ representing the global state space G and the interdependency of strategies in the prefix $\wp\flat$.
3. We combine $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$ and $\mathfrak{P}^{\psi}_{\mathcal{I}}$ into an infinite two-player *parity game* $\mathfrak{G}^{\wp\flat\psi}_{\mathcal{I}}$. Solving the parity game yields its winning regions and strategies, which can in turn be used to calculate $\|\varphi\|_{\mathcal{I}}$ and the strategies in $\wp$.

We shall now expand on each of the steps above.

Formula automaton. The first step of our algorithm is the standard construction of a deterministic parity automaton $\mathfrak{P}^{\psi}_{\mathcal{I}}$ equivalent to the underlying LTL formula ψ. This is usually performed in three steps: *(i)* ψ is converted to a non-deterministic generalised Büchi automaton $\mathfrak{A}^{\psi}_{\mathcal{I}}$ via standard translation (Schneider 2004); *(ii)* $\mathfrak{A}^{\psi}_{\mathcal{I}}$ is translated to an equivalent non-deterministic Büchi automaton $\mathfrak{B}^{\psi}_{\mathcal{I}}$ by adding a counter for fairness constraints (Schneider 2004); *(iii)* $\mathfrak{B}^{\psi}_{\mathcal{I}}$ is transformed into a deterministic parity automaton $\mathfrak{P}^{\psi}_{\mathcal{I}} = (S, s_I, \delta, c)$ with a non-empty set of states S, an initial state $s_I \in S$, a transition function $\delta : S \times G \to S$, and a colouring function $c : S \to \mathbb{N}$. While the third step is typically done using Safra's construction (Safra 1988), we perform the determinisation using a recently put forward procedure (Morgenstern and Schneider 2008) instead, which is amenable to a symbolic implementation. It is worth pointing out that the recursive step (replacing direct principal subsentences φ' with atoms $p_{\varphi'}$) can be incorporated as an extra case of the standard translation in the first step.

As an example, consider the simple interpreted system $\mathcal{I}_{\mathrm{RPS}}$ in Figure 1a with agents $Agt \triangleq \{1, 2\}$ representing the *Rock-Paper-Scissors game*. The global states of the system are $G \triangleq \{g_{\mathrm{g}}, g_1, g_2\}$ meaning "game", "player 1 won", and "player 2 won", respectively. The actions available to both players are: $Act_1 \triangleq Act_2 \triangleq \{\mathrm{r}, \mathrm{p}, \mathrm{s}, \mathrm{i}\}$ meaning "rock", "paper", "scissors", and "idle". Finally, the atoms p_1 and p_2 encode that player 1 and player 2 won, respectively. The assignment is defined as $h(p_1) \triangleq \{g_1\}$ and $h(p_2) \triangleq \{g_2\}$. Furthermore, consider the SL[1G] basic principal sentence $\gamma \triangleq [\![x]\!]\langle\!\langle y \rangle\!\rangle (1, x)(2, y)\, \mathsf{G}\, [\neg p_1 \wedge \neg p_2]$ which expresses that *"Whichever action player 1 performs, there exists an action for player 2 such that neither player will ever win"*. The corresponding deterministic parity automaton $\mathfrak{P}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$ con-

[1] If this is not the case, one can simply add one quantifier and agent binding for each agent without changing the semantics as φ is a sentence.

structed using the three-step procedure described in the previous paragraph is shown in Figure 1b.

Formula arena. The second step of the algorithm involves building a two-player formula arena $\mathcal{A}^{\wp\flat}_{\mathcal{I}} = (V_0, V_1, E)$, which encodes the state space of the interpreted system $\mathcal{I}$ and the interdependency of strategies in the prefix $\wp\flat$. The vertices V of $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$ are pairs $(g, d) \in G \times Dec^{\wp\flat}_{\mathcal{I}}$ of global reachable states and lists of actions such that for all $0 \le k < |d|$ we have $d(k) \in \bigcap_{i \in \mathsf{sharing}(\flat\top, \wp_v(k))} P_i(l_i(g))$, where $Dec^{\wp\flat}_{\mathcal{I}} \triangleq \bigcup_{\ell=0}^{|\wp|} \prod_{k=0}^{\ell-1} Act_{\mathsf{sharing}(\flat\top, \wp_v(k))}$ and $l_i(g)$ is the local state of agent i in g. The existential player vertices $V_0 \subseteq V$ are vertices $(g, d) \in V$ such that $|d| < |\wp|$ and $\wp(|d|)$ is an existential strategy quantifier. Conversely, the universal player vertices are $V_1 = V \setminus V_0$. The edge relation $E \subseteq V \times V$ is defined as:

$$E \triangleq \{((g,d),(g,d\cdot a)) \in V \times V \mid |d| < |\wp|\} \cup \left\{\left((g,d),\left(t(g,d^{Act}),[]\right)\right) \in V \times V \;\middle|\; |d| = |\wp|\right\}$$

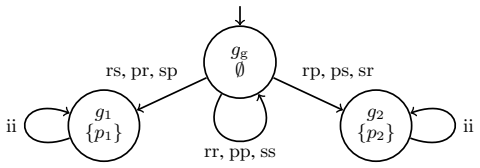
where $d^{Act} \in Act$ is a joint action such that for all $0 \le k < |\wp|$ and $i \in \mathsf{sharing}(\flat\top, \wp_v(k))$ we have $\mathsf{act}_i(d^{Act}) = d(k)$.

Intuitively, the existential (universal) player represents all existential (universal) quantifiers in the quantification prefix $\wp$. Equivalently, the two players correspond to the existential-universal partition of Agt. The game starts in some vertex $(g, [])$. The players take turns to select actions $d(0), \ldots, d(|\wp|-1)$ for the quantifiers $\wp(0), \ldots, \wp(|\wp|-1)$. The decision d then determines the joint action of all agents d^{Act} and a temporal transition to $\left(t(g, d^{Act}), []\right)$ is performed. This pattern is repeated forever. The formula arena $\mathcal{A}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$ of the Rock-Paper-Scissors game interpreted system $\mathcal{I}_{\mathrm{RPS}}$ for the SL[1G] formula γ introduced earlier is shown in Figure 1d. Observe that the three grey blobs in $\mathcal{A}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$ correspond to the three global reachable states in Figure 1a.

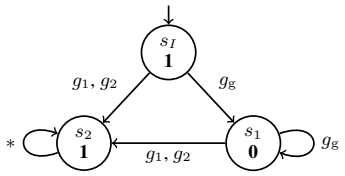
We now consider a pseudo-LTL game $\mathfrak{L}^{\wp\flat\psi}_{\mathcal{I}}$ based on the arena $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$. We define an infinite path $\pi \in V^{\omega}$ in $\mathfrak{L}^{\wp\flat\psi}_{\mathcal{I}}$ to be winning for the existential player iff the LTL formula ψ holds along the underlying infinite path $\pi_{\mathcal{I}} \in G^{\omega}$ in $\mathcal{I}$.

Lemma 1. *An* SL[1G] *principal sentence $\wp\flat\psi$ holds at a global state $g \in G$ in an interpreted system $\mathcal{I}$ iff the vertex $(g, []) \in V$ is winning for the existential player in the pseudo-*LTL *game $\mathfrak{L}^{\wp\flat\psi}_{\mathcal{I}}$ defined above.*

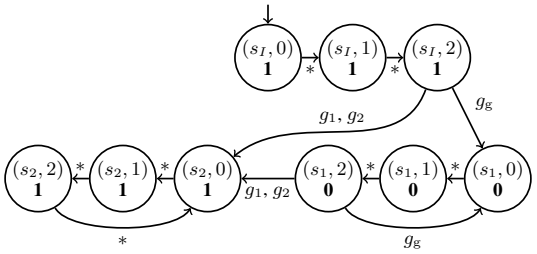
Proof. This follows from the fact that SL[1G] model checking can be reduced to solving a so-called *dependence-vs-valuation game* (Mogavero et al. 2014) in which the existential player chooses a dependence map $\theta : \left(\llbracket\wp\rrbracket \to \bigcup_{i\in Agt} Act_i\right) \to \left(\wp \to \bigcup_{i\in Agt} Act_i\right)$ for $\wp$ over actions in the current global state $g \in G$ and then the universal player chooses a valuation $v : \llbracket\wp\rrbracket \to \bigcup_{i\in Agt} Act_i$. The combination $\theta(v) : \wp \to \bigcup_{i\in Agt} Act_i$ assigns actions to all variables and determines the next state $g' \in G$. Instead of choosing the whole dependence map and valuation at once, the players in $\mathfrak{L}^{\wp\flat\psi}_{\mathcal{I}}$ assign actions to strategies one by one for each quantifier. Furthermore, the order of the players' moves in $\mathfrak{L}^{\wp\flat\psi}_{\mathcal{I}}$ game ensures that the independence constraints of θ are satisfied. Hence, our claim follows. □



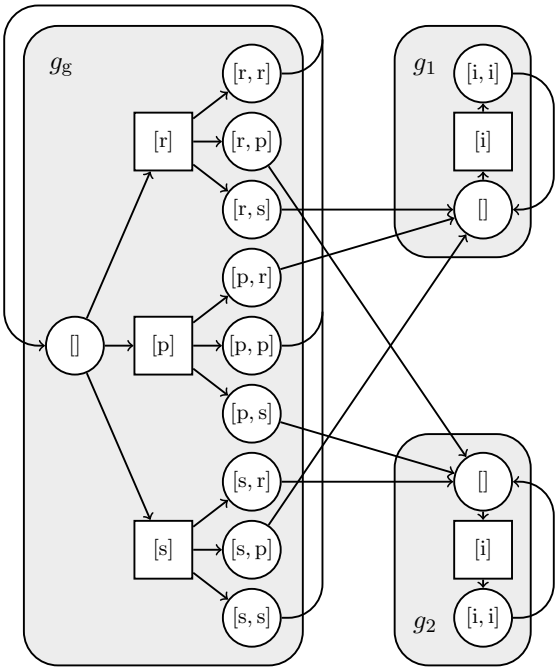
(a) Interpreted system $\mathcal{I}_{\mathrm{RPS}}$.



(b) Deterministic parity automaton $\mathfrak{P}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$.



(c) Delayed automaton $\mathfrak{D}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$.



(d) Formula arena $\mathcal{A}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$. The existential and universal player states are represented by squares and circles respectively.

Figure 1: Interpreted system $\mathcal{I}_{\mathrm{RPS}}$, parity automaton $\mathfrak{P}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$, delayed automaton $\mathfrak{D}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$, and formula arena $\mathcal{A}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$ of the Rock-Paper-Scissors game and the SL[1G] basic principal sentence $\gamma \triangleq \llbracket x\rrbracket\langle\langle y\rangle\rangle(1,x)(2,y)\,\mathsf{G}\,[\neg p_1 \wedge \neg p_2]$.

We shall next explain how this pseudo-LTL game can be converted to a standard parity game.

Combined game. In order to construct the combined parity game, the solving of which is equivalent to model checking the basic principal sentence $\wp\flat\psi$, we need to combine the formula automaton $\mathfrak{P}^{\psi}_{\mathcal{I}}$ and the formula arena $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$ because $\mathfrak{P}^{\psi}_{\mathcal{I}}$ represents the winning condition of the pseudo-LTL game $\mathfrak{L}^{\wp\flat\psi}_{\mathcal{I}}$. However, we cannot simply take their product because, informally, they work at different, albeit constant, "speeds". While $\mathfrak{P}^{\psi}_{\mathcal{I}}$ performs a temporal transition at every step, it takes exactly $|\wp| + 1$ turns before a different underlying global state (grey blob in Figure 1d) is reached by $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$. To cater for this asynchrony, we can make the parity automaton "wait" for $|\wp|+1$ steps before each actual transition. We do this by extending the state of $\mathfrak{P}^{\psi}_{\mathcal{I}}$ with a simple counter from 0 to $|\wp|$. The resulting delayed (deterministic parity) automaton $\mathfrak{D}^{\gamma}_{\mathcal{I}_{\mathrm{RPS}}}$ for the basic principal sentence γ introduced earlier is shown in Figure 1c.

The delayed automaton $\mathfrak{D}^{\wp\psi}_{\mathcal{I}}$ accepts precisely those paths in the formula arena $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$ which are winning for the existential player. Hence, by combining the two structures, we obtain the combined parity game $\mathfrak{G}^{\wp\flat\psi}_{\mathcal{I}} \triangleq ((V_0 \times S, V_1 \times S, E_{\mathfrak{G}}), c_{\mathfrak{G}})$ with edge relation and colouring function defined as $E_{\mathfrak{G}} \triangleq \{((g,d,s),(g',d',s')) \in (V \times S) \times (V \times S) \mid E((g,d),(g,d)) \wedge \delta_{\mathfrak{D}}((s,|d|),g)\}$ and $c_{\mathfrak{G}}((g,d,s)) \triangleq c(s)$ respectively, where $\delta_{\mathfrak{D}}$ is the transition function of the delayed automaton.

Model Checking. Model checking of an SL[1G] principal sentence can finally be performed by solving the corresponding combined parity game (e.g. using Zielonka's algorithm (Zielonka 1998)) as formalised by the following lemma:

Lemma 2. *Let $\wp\flat\psi$ be an* SL[1G] *principal sentence, $g \in G$ a global state in an interpreted system $\mathcal{I}$, and (W_0, W_1) the winning regions of the combined parity game $\mathfrak{G}^{\wp\flat\psi}_{\mathcal{I}}$. $\wp\flat\psi$ holds at g (i.e. $\mathcal{I}, \emptyset, g \models \wp\flat\psi$) iff the vertex $(g, [], s_I)$ is in the winning region of the existential player (i.e. $(g, [], s_I) \in W_0$).*

Proof. Our claim follows directly from Lemma 1 and the correctness of the determinisation procedure. □

Strategy Synthesis. The formula arena encodes the effects and interdependency of agents' actions. Therefore, the solution, i.e., the winning strategies, of the combined parity game can be used for strategy synthesis.

Lemma 3. *Let $\wp\flat\psi$ be an* SL[1G] *principal sentence, $\mathcal{I}$ an interpreted system, (w_0, w_1) the winning strategies of the combined parity game $\mathfrak{G}^{\wp\psi\flat}_{\mathcal{I}}$, $0 \leq k < |\wp|$ an integer, and $f_0, \ldots, f_{k-1}$ strategies for variables $\wp_v(0), \ldots, \wp_v(k-1)$. Then the strategy $f_k : Trk \to Act_{\mathsf{sharing}(\flat\psi, \wp_v(k))}$ is defined for all tracks $\pi \in Trk$ implicitly as $\hat{w}((\mathsf{last}(\pi), [f_0(\pi), \ldots, f_{k-1}(\pi)], \delta(s_I, \pi_{\leq|\pi|-2}))) = (\mathsf{last}(\pi), [f_0(\pi), \ldots, f_{k-1}(\pi), \underline{f_k(\pi)}], \delta(s_I, \pi_{\leq|\pi|-2}))$ where $\delta(s_I, \pi_{\leq|\pi|-2}) \triangleq \delta(\ldots\delta(s_I, \pi(0))\ldots, \pi(|\pi|-2))$ and $\hat{w} : V \to \bar{V}$ is a total function such that $w_0 \cup w_1 \subseteq \hat{w}$.*

Proof. The correctness follows from the structure of the formula arena $\mathcal{A}^{\wp\flat}_{\mathcal{I}}$. See (Čermák 2014) for more details. □

Optimality. The theoretical complexity of SL[1G] model checking is 2EXPTIME-COMPLETE with respect to the size of the formula and P-COMPLETE with respect to the size of the model (Mogavero et al. 2014). Given this, we show that our algorithm has *optimal time complexity*:

Lemma 4. *Let φ be an arbitrary* SL[1G] *sentence and $\mathcal{I}$ an interpreted system. Our algorithm calculates the set of all global states $\|\varphi\|_{\mathcal{I}} \subseteq G$ satisfying φ in time $|\mathcal{I}|^{2^{O(|\varphi|)}}$.*

Proof. Let us first consider an arbitrary SL[1G] basic principal sentence $\wp\flat\psi$. The automata $\mathfrak{A}^{\psi}_{\mathcal{I}}$, $\mathfrak{B}^{\psi}_{\mathcal{I}}$, $\mathfrak{P}^{\psi}_{\mathcal{I}}$, and $\mathfrak{D}^{\wp\psi}_{\mathcal{I}}$ have $O(2^{|\psi|})$, $2^{O(|\psi|)}$, $2^{2^{O(|\psi|)}}$, and $|\wp| \times 2^{2^{O(|\psi|)}}$ states. Moreover, both parity automata have $2^{O(|\psi|)}$ colours. The arena $\mathcal{A}^{\flat\wp}_{\mathcal{I}}$ and game $\mathfrak{G}^{\wp\flat\psi}_{\mathcal{I}}$ have $O(|I|^{|\wp|})$ and $|I|^{|\wp|} \times 2^{2^{O(|\psi|)}}$ states. Given the number of states and colours, the game can be solved in time $|I|^{2^{O(|\wp\flat\psi|)}}$ (Jurdziński 2000).

We model check φ in a recursive bottom-up manner as explained earlier. Hence, at most $|\varphi|$ SL[1G] basic principal sentences of size at most $|\varphi|$ need to be checked. If φ is *not* a principal sentence, it must be a Boolean combination of principal sentences, the results of which we can combine using set operations. Thus, the model checking time is $|\varphi| \times |I|^{2^{O(|\varphi|)}} + |\varphi| \times |\mathcal{I}| = |I|^{2^{O(|\varphi|)}}$ and our claim follows. □

Note that SL[1G] subsumes ATL*, which has the same model checking complexity (Laroussinie, Markey, and Oreiby 2007). Hence, our algorithm is also optimal for ATL* model checking. Moreover, the same complexity result applies to SL[1G] and, consequently, ATL* strategy synthesis.

4 Implementation and Experimental Results

We implemented the algorithm presented in the previous section as part of the new experimental model checker MCMAS-SL[1G]. The tool, available from (MCMAS-SL[1G]), takes as input the system in the form of an ISPL file (Lomuscio, Qu, and Raimondi 2009) describing the agents in the system, their local states, actions, protocols, and evolution functions as well as the SL[1G] specifications to be verified. Upon invocation MCMAS-SL[1G] calculates the set of reachable states, encoded as BDDs, and then checks whether each specification holds in the system. If requested, all quantified strategies in all formulas are synthesised (together with their interdependencies). While MCMAS-SL[1G] is built from the existing open-source model checker MCMAS (Lomuscio, Qu, and Raimondi 2009) and shares some of its algorithms, it necessarily differs from MCMAS in several key components, including a more complex encoding of the model, as described in the previous section, as well as the novel procedure for computing the sets of states satisfying SL[1G] formulas.

Evaluation. To evaluate the proposed approach, we present the experimental results obtained for the problem of fair process scheduler synthesis. The experiments were run on an Intel® Core™ i7-3770 CPU 3.40GHz machine with

algo.	processes n	possible states	reachable states	reachability time	model checking time (s)					memory (MB)
					total	automaton	arena	game	solve	
unoptimised	2	72	9	0.00	0.09	0.00	0.00	0.02	0.06	4.44
	3	432	21	0.00	10.11	0.01	0.02	1.06	9.02	15.14
	4	2592	49	0.01	631.11	0.33	0.26	86.47	544.04	41.80
	5	15552	113	0.02	29593.56	5.86	2.61	4323.81	25261.15	2792.22
	6	93312	257	0.02	*out of memory*					
optimised	2	72	9	0.00	0.12	0.00	0.00	0.02	0.10	4.52
	3	432	21	0.00	6.39	0.00	0.01	0.65	5.72	14.54
	4	2592	49	0.01	338.16	0.00	0.24	23.33	314.57	40.70
	5	15552	113	0.02	6131.43	0.00	2.65	444.06	5684.69	306.41
	6	93312	257	0.02	85976.57	0.00	38.27	8012.11	77925.96	2688.93

Table 1: Verification results for the fair process scheduler synthesis.

16GB RAM running Linux kernel version 3.8.0-35-generic. Table 1 reports the performance observed when *synthesising* a process scheduler satisfying the following SL[1G] specification which asserts absence of starvation (Mogavero, Murano, and Sauro 2013):

$$\phi \triangleq \xi \bigwedge_{i=1}^{n} \mathsf{G}\left(\langle \mathrm{wt}, i\rangle \rightarrow \mathsf{F} \neg \langle \mathrm{wt}, i\rangle\right)$$

where $\xi \triangleq \langle\!\langle x \rangle\!\rangle [\![y_1]\!] \cdots [\![y_n]\!] (\mathrm{Sched}, x)(1, y_1) \cdots (n, y_n)$ is a prefix and $\langle \mathrm{wt}, i\rangle$ denotes that process $1 \leq i \leq n$ is waiting for the resource.

We ran the experiments with two different versions of the SL[1G] model checking algorithm: an unoptimised one (described in the previous section) and an optimised one. Given an SL[1G] principal sentence of the form $\wp\flat(\psi_0 \wedge \psi_1 \wedge \cdots \wedge \psi_{n-1})$, the optimised algorithm determinises each conjunct ψ_i with $0 \leq i < n$ separately, i.e. it constructs the delayed automata $\mathfrak{D}_{\mathcal{I}}^{\wp\psi_0}, \mathfrak{D}_{\mathcal{I}}^{\wp\psi_1}, \ldots, \mathfrak{D}_{\mathcal{I}}^{\wp\psi_{n-1}}$. The resulting combined game $\mathfrak{G}_{\mathcal{I}}^{\wp\flat\psi} \triangleq \mathcal{A}_{\mathcal{I}}^{\wp\flat} \times \prod_{i=0}^{n-1} \mathfrak{D}_{\mathcal{I}}^{\wp\psi_i}$ is a *generalised parity game* (Chatterjee, Henzinger, and Piterman 2006). The reasoning behind this optimisation is that the size of the deterministic automata is doubly exponential in the size of the LTL formulas. Hence, separate determinisation may lead to much smaller combined games.

In the experiments, MCMAS-SL[1G] synthesised correct strategies using both versions of the algorithm. The results show that the main performance bottlenecks are the construction and solution of the combined parity game; this is in line with the theoretical complexity results reported in the proof of Lemma 4. We can observe that separate determinisation has indeed a significant impact on performance in terms of both time and memory footprint, thereby allowing us to reason about more processes. Note that the relative speedup increases with the number of processes with gains quickly reaching an order of magnitude and more.

We tested the tool on various other scalable scenarios (Čermák 2014). When verifying a fixed-size formula, the tool has efficiently handled systems with 10^5 reachable global states. This is approximately an order of magnitude worse than the MCMAS's performance on plain ATL specifications. This is because the expressiveness of SL[1G] requires a richer encoding for the models, as discussed earlier. We are not aware of any other tool capable of verifying specifications richer than plain ATL under the assumptions of perfect recall. Therefore, we cannot compare our results to any other in the literature.

5 Conclusions

Most approaches put forward over the past ten years for the verification of MAS are concerned with temporal-epistemic properties so to assess the evolution of the knowledge of the agents over time. Considerably less attention has been devoted so far to the problem of establishing what strategic properties agents in a system have. We are aware of two lines of research concerning this. The first concerns the verification of MAS against ATL specifications (Alur et al. 2001; Lomuscio and Raimondi 2006; Kacprzak and Penczek 2005); the second pertains to the verification of systems against an observational fragment of SL to which epistemic modalities are added (Čermák et al. 2014; Huang and Meyden 2014). As argued in the literature, the first line is limited by the fact that ATL specifications are not sufficiently rich to refer to strategies explicitly. The second direction suffers from the weakness of the observational fragments analysed as they cannot account for the perfect recall abilities normally assumed in a strategic setting.

In this paper we attempted to overcome both difficulties above and put forward a fully symbolic approach to the verification of MAS against specifications in SL[1G], a rich behaviourally fragment of SL. We showed the algorithm developed is provably optimal and built a BDD-based checker to support it. The experimental results obtained point to the feasibility of the practical verification problem for MAS against SL[1G] specifications. Since SL[1G] strictly subsumes ATL*, an important byproduct of the work presented is the fact that it also constitutes the first verification toolkit for ATL*. A further key innovative feature of our approach is that it does not only support verification, but also strategy synthesis. This enables us to use the synthesis engine for developing controllers or automatic planners in a MAS context. We leave this to further work.

Acknowledgments. The research described in this paper was partly supported by the EPSRC Research Project "Trusted Autonomous Systems" (grant No. EP/I00520X/1) and FP7 EU project 600958-SHERPA).

References

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

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Bulling, N., and Jamroga, W. 2014. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *Autonomous Agents and Multi-Agent Systems* 28(3):474–518.

Čermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2014. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Proceedings of the 26th International Conference on Computer Aided Verification (CAV'14)*, LNCS 8559, 524–531. Springer.

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

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2006. Generalized parity games. Technical Report UCB/EECS-2006-144, University of California, Berkeley.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Inf. Comput.* 208(6):677–693.

Dima, C., and Tiplea, F. L. 2011. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR* abs/1102.4225.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about Knowledge*. Cambridge: MIT Press.

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

Huang, X., and Meyden, R. v. 2014. Symbolic model checking epistemic strategy logic. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 1426–1432. AAAI Press.

Jurdziński, M. 2000. Small progress measures for solving parity games. In *Proceedings of the 17th Symposium on Theoretical Aspects of Computer Science (STACS'00)*, LNCS 1770. Springer. 290–301.

Kacprzak, M., and Penczek, W. 2005. Fully symbolic unbounded model checking for alternating-time temporal logic. *Autonomous Agents and Multi-Agent Systems* 11(1):69–89.

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

Kouvaros, P., and Lomuscio, A. 2013. A cutoff technique for the verification of parameterised interpreted systems with parameterised environments. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2013–2019. AAAI Press.

Laroussinie, F.; Markey, N.; and Oreiby, G. 2007. On the expressiveness and complexity of ATL. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'07)*, LNCS 4423. Springer. 243–257.

Lomuscio, A., and Raimondi, F. 2006. 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)*, 161–168. ACM Press.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. 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, 682–688. Springer.

MCMAS-SL[1G] – a model checker for the verification of one-goal strategy logic specifications. http://vas.doc.ic.ac.uk/software/tools/, 2014.

Meski, A.; Penczek, W.; Szreter, M.; Wozna-Szczesniak, B.; and Zbrzezny, A. 2014. BDD versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: algorithms and their performance. *Autonomous Agents and Multi-Agent Systems* 28(4):558–604.

Meyden, R. v., and Shilov, H. 1999. Model checking knowledge and time in systems with perfect recall. In *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, LNCS 1738, 432–445.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2014. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Logic* 15(4):1–47.

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

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010. Reasoning about strategies. In *Proceedings of the 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*, LIPIcs 8, 133–144.

Morgenstern, A., and Schneider, K. 2008. From LTL to symbolically represented deterministic automata. In *Proceedings of the 9th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'08)*, LNCS 4905. Springer. 279–293.

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

Safra, S. 1988. On the complexity of ω-automata. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, 319–327.

Schneider, K. 2004. *Verification of Reactive Systems*. Texts in Theoretical Computer Science. Springer.

Zielonka, W. 1998. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science* 200(1–2):135–183.