Satisfiability Checking of Strategy Logic with Simple Goals

Magdalena Kacprzak[1], **Artur Niewiadomski**[2], **Wojciech Penczek**[3]
[1]Bialystok University of Technology, Bialystok, Poland
[2]Siedlce University, Faculty of Exact and Natural Sciences, Siedlce, Poland
[3]Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
m.kacprzak@pb.edu.pl, artur.niewiadomski@uph.edu.pl, penczek@ipipan.waw.pl

Abstract

In this paper, we introduce a new method of the satisfiability (SAT) checking for *Simple-Goal Strategy Logic* (SL[SG]), using symbolic Boolean model encoding and the SAT Modulo Monotonic Theories techniques, which was implemented into the tool SGSAT. To the best of our knowledge, this is the only tool solving the SAT problem for SL[SG]. Its applications include process synthesis, developing controllers as well as automatic planners in multi-agent scenarios.

1 Introduction

An important area of research in artificial intelligence systems is the development of techniques for reasoning on strategies, especially in open-systems, where the interaction between the system and the environment is viewed as a strategy game. Recently, a lot of work in the field of verification of multi-agent system (MAS) has been devoted to establishing what strategic properties in the system agents have (Jamroga and van der Hoek 2004; Ågotnes, Goranko, and Jamroga 2007; Walther, van der Hoek, and Wooldridge 2007; Pauly 2002; Kacprzak and Penczek 2005). The two main lines of research involve specifications in the formalisms of *Alternating-time Temporal Logic* (ATL and ATL*) (Alur, Henzinger, and Kupferman 2002) supported by the tools like MOCHA (Alur et al. 2001) or MCMAS (Lomuscio and Raimondi 2006) and its generalization, *Strategy Logic* (SL) (Mogavero, Murano, and Vardi 2010) supported by the tools like MCMAS-SLK (Cermák et al. 2014).

SL was originally defined over perfect recall and complete information semantics. It extends Linear-time Temporal Logic (LTL) (Pnueli 1977) by strategy quantifiers and agent bindings. The syntactic separation of these concepts makes it possible to construct complex formulae in which different temporal goals referring to the same strategies can be combined and nested, as well as different players can share the same strategy. Strategies are treated as a first order concept over strategy and agent variables, which allows for expressing interesting properties like strategic plans over temporal goals, Nash equilibria or Stackelberg equilibria.

High expressivity of SL has its price, namely the high computational complexity. The model checking problem (MC) for SL is non-elementarily decidable - k-EXPSPACE-HARD in the alternation number k of quantifications in the specification (Mogavero et al. 2014). The satisfiability problem (SAT) for SL is highly undecidable - Σ_1^1-HARD. Moreover, the logic does not have the bounded-tree model property (Mogavero et al. 2017). Since this makes SL quite limited in its practical application, restrictions of SL are of interest. These include *One-Goal Strategy Logic* (SL[1G]) having the SAT problem in 2EXPTIME (Mogavero et al. 2014) and its sublogic *Simple-Goal Strategy Logic* (SL[SG]) for which MC is P-complete (Belardinelli et al. 2019).

This paper offers a new method of SAT checking for SL[SG] extending the one for ATL (Kacprzak, Niewiadomski, and Penczek 2020) and uses symbolic Boolean model encoding and the SAT Modulo Monotonic Theories (SMMT) techniques (Bayless et al. 2015). The method was implemented into the tool SGSAT, which, to our best knowledge, is the only one solving the SAT problem for SL[SG]. Applications of SL[SG] include specification of Stackelberg equilibria, coercion-resistance, and module checking for strategic abilities (Belardinelli et al. 2019; Tabatabaei, Jamroga, and Ryan 2016; Jamroga and Murano 2014).

Typically, given a SAT method for a logic, one can solve the MC problem using the automata approach. In this paper we show that it is possible to solve the SAT problem, given a MC method. We develop an efficient method for generating candidate models for an SL[SG] formula, and then use an existing MC method for checking whether the formula holds in some of these models. The best analogy to our approach is how SAT-solvers check satisfiability of a propositional formula. Clearly, only some candidate valuations are generated thanks to sophisticated SAT-algorithms like CDCL (Silva and Sakallah 2003). Similarly, in our case, only some candidate models are generated thanks to the algorithms for SMMT. This is where monotonicity of our theory is exploited, which allows for a substantial reduction in the number of generated model candidates. Therefore, our SAT method for SL[SG] at the stage of finding candidate models can be as efficient as a SAT-solver is in finding candidate valuations for a propositional formula.

Our paper is not the first one adopting that approach. Here we build on the results for CTL (Klenze, Bayless, and Hu 2016) and ATL (Niewiadomski et al. 2020). The main contributions for SL[SG] consist in: a new encoding method for SL[SG] models, new proofs of monotonicity of the relation $\models$ for SL[SG] in some cases, new definitions of over-

and under-approximations for SL[SG] partial models, new proofs of the correctness of the method for the formulae prefixed with strategy quantifiers and bindings, and a new implementation and the tool.
It is worth mentioning that with some technical difficulties our approach for SL[SG] can be extended to SL[1G] and SL.

Outline. The related work is discussed in Sec. 2. Next, the necessary definitions are given in Sec. 3. In Sec. 4 the Boolean encoding of MAS and its model is introduced. The proof of the monotonicity property for SL[SG] is in Sec. 5, followed by the construction of the monotonic approximation for a given class of models, in Sec. 6. Sec. 7 and 8 describe the satisfiability procedure, its implementation, experimental results, and provide conclusions.

2 Related Work

To overcome the problem of high computational complexity, several restrictions of SL are considered. The following logics have been defined over the original semantics: *Nested-Goal Strategy Logic* (SL[NG]), *Boolean Goal Strategy Logic* (SL[BG]), *Conjunctive-Goal Strategy Logic* (SL[CG]), *Disjunctive-Goal Strategy Logic* (SL[DG]), and the already mentioned *One-Goal Strategy Logic* (SL[1G]) (Mogavero et al. 2014; Mogavero, Murano, and Sauro 2013). The SAT problem for SL[NG] and SL[BG] is undecidable and strategies are non-behavioural. SAT for SL[CG] and SL[DG] is an open problem while the strategies are behavioural. SL[1G], unlike the previous logics, enjoys both desired properties: the decidable SAT problem and behavioural strategies which is fundamental for proving decidability (Mogavero et al. 2014). The SAT procedure for SL[1G] is given in (Mogavero et al. 2017).

In (Acar, Benerecetti, and Mogavero 2019) a fragment of SL is considered, which prevents strategic quantification within the scope of temporal operators, called *Flat Conjunctive-Goal Strategy Logic* (FSL[CG]). This logic still allows for expressing the existence of Nash and immune equilibria, but its SAT problem is PSPACE-complete while the MC problem is 2EXPTIME-HARD.

A model checker for Strategy Logic with Knowledge for systems with memoryless strategies and incomplete information is introduced in (Cermák et al. 2018). This tool is an extension of the model checker MCMAS, which has also been enhanced with verification of SL[1G] defined for systems with perfect recall and complete information (Cermák, Lomuscio, and Murano 2015). An epistemic extension of SL with incomplete information and uniform and coherent strategies is also defined in (Belardinelli et al. 2017).

3 Strategy Logic with Simple Goals

We deal with a fragment of SL, restricted to "simple" goals (Belardinelli et al. 2019). However, the semantics is defined over interpreted systems (IS) (Fagin et al. 1995; Lomuscio and Raimondi 2006) instead of concurrent game structures.

3.1 Multi-agent System and Interpreted System

We start with a definition of multi-agent systems.

Definition 1 (MAS). *A* multi-agent system (MAS) *consists of n agents $\mathcal{A} = \{1, \ldots, n\}$*[1]*, where each agent $i \in \mathcal{A}$ is associated with a 7-tuple $AG_i = (L_i, \iota_i, Act_i, P_i, T_i, \mathcal{PV}_i, V_i)$ including:*
(1) *a finite non-empty set of* local states $L_i = \{l_i^1, l_i^2, \ldots, l_i^{n_i}\}$*;* (2) *an* initial local state $\iota_i \in L_i$*;* (3) *a finite non-empty set of* local actions $Act_i = \{a_i^1, a_i^2, \ldots, a_i^{m_i}\}$*;* (4) *a* local protocol $P_i : L_i \to 2^{Act_i} \setminus \{\emptyset\}$ *selecting the actions available at each local state;* (5) *a (partial)* local transition function $T_i : L_i \times Act \to L_i$ *such that $T_i(l_i, \alpha)$ is defined iff $\alpha^i \in P_i(l_i)$, where $Act \triangleq \prod_{i \in \mathcal{A}} Act_i$ is the set of* joint (global) actions *of all agents; α^i is the action of agent $i \in \mathcal{A}$ in the joint action $\alpha \in Act$;* (6) *a finite non-empty set of* local propositions $\mathcal{PV}_i = \{p_i^1, p_i^2, \ldots, p_i^{r_i}\}$*;* (7) *a local valuation function $V_i : L_i \to 2^{\mathcal{PV}_i}$.*

For every non-empty set $\Gamma \subseteq \mathcal{A}$, the set of *shared actions* of agents Γ is determined $Act_\Gamma \triangleq \cap_{i \in \Gamma} Act_i$. Such a set must be non-empty if we want to ensure the existence of a non-empty set of *shared strategies* for Γ (i.e. a set of strategies that can be used by every $i \in \Gamma$).

Notice that we consider *synchronous* multi-agent systems, where each *global action* is a n-tuple $\langle a_i \rangle_{i \in \mathcal{A}}$ with $a_i \in Act_i$, i.e., each agent performs one local action. To describe the behavior of a MAS, its model is defined.

Definition 2 (Model). *Let $\mathcal{PV} = \bigcup_{i=1}^n \mathcal{PV}_i$ be the union of the local propositions. The* model *for MAS is a 4-tuple $M = (\mathcal{S}t, \iota, T, V)$, where*
1) $\mathcal{S}t = L_1 \times \cdots \times L_n$ is a set of the global states,
2) $\iota = (\iota_1, \ldots, \iota_n) \in \mathcal{S}t$ is the initial global state,
3) $T : \mathcal{S}t \times Act \to \mathcal{S}t$ is the partial global transition function, such that $T(g, \alpha) = g'$ iff $T_i(g_i, \alpha) = g'_i$ for all $i \in \mathcal{A}$, where for a global state $g = (l_1, \ldots, l_n)$, by $g_i = l_i$ we denote the local component of agent i,
4) $V : \mathcal{S}t \to 2^{\mathcal{PV}}$ is the valuation function such that $V((l_1, \ldots, l_n)) = \bigcup_{i=1}^n V_i(l_i)$.

We say that action $\alpha \in Act$ is *enabled* at $g \in \mathcal{S}t$ if $T(g, \alpha) = g'$ for some $g' \in \mathcal{S}t$. We assume that at each $g \in \mathcal{S}t$ there exists at least one enabled action, i.e., for each $g \in \mathcal{S}t$ there are $\alpha \in Act$ and $g' \in \mathcal{S}t$ s.t. $T(g, \alpha) = g'$. A *path* is a (finite or infinite) sequence of global states $\pi \in \mathcal{S}t^* \cup \mathcal{S}t^\omega$ such that for each $j \geq 1, \pi_{j+1} = T(\pi_j, \alpha_j)$ for some joint action $\alpha_j \in Act$, where π_j denotes the j-th state of π. We distinguish between finite paths, or *histories* of $\mathcal{S}t^*$, and infinite paths, or *computations* of $\mathcal{S}t^\omega$. For a path π and $j \geq 1$, $\pi_{\leq j}$ denotes the initial history of length j, and $last(h)$ is last element in history h.

Example 1. *Consider the following MAS:*

- $\mathcal{A} = \{1, 2\}$*;* $L_1 = \{1, 2, 3\}, \iota_1 = 1, L_2 = \{1, 2\}, \iota_2 = 1,$
- $Act_1 = \{1, 2\}$, $P_1 = \{(1, \{1\}), (2, \{1\}), (3, \{1, 2\}\},$
- $Act_2 = \{1, 2\}$, $P_2 = \{(1, \{1, 2\}), (2, \{2\})\},$
- $T_1 = \{(1, (1, 1), 2), (1, (1, 2), 3), (2, (1, 1), 3), (2, (1, 2), 3),$ $(3, (1, 1), 2), (3, (1, 2), 2), (3, (2, 2), 2), (3, (2, 1), 1)\},$
  $T_2 = \{(1, (1, 1), 1), (1, (2, 1), 1), (1, (1, 2), 2), (1, (2, 2), 2),$

[1]The environment component may be added here with no technical difficulty.

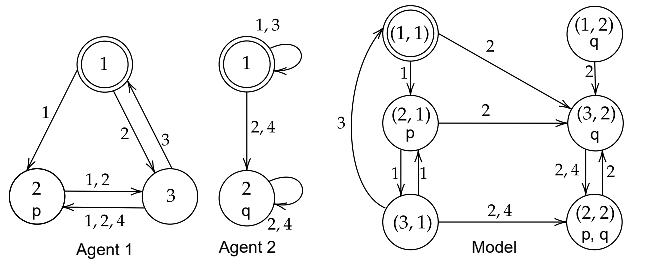

Figure 1: Example MAS and induced model.

$(2,(1,2),2),(2,(2,2),2)\}$; $\mathcal{PV}_1 = \{p\}, \mathcal{PV}_2 = \{q\}$, $V_1 = \{(1,\emptyset),(2,\{p\}),(3,\emptyset)\}, V_2 = \{(1,\emptyset),(2,\{q\})\}$,

depicted in Fig. 1 (left), while at the right hand side of the figure its induced model is presented. For a better readability each joint action is represented by the integer corresponding to the action's position sorted lexicographically. That is, 1 states for $(1,1)$, 2 represents $(1,2)$, and so on.

3.2 Syntax and Semantics of SL[SG]

The following definitions, needed for defining the syntax of SL[SG], are adopted from (Belardinelli et al. 2019). Let *Var* be an infinite set of variables $x_0, x_1, \ldots$ for referring to the strategies. A *binding prefix* over a set $\Gamma \subseteq \mathcal{A}$ of agents and a set $\mathcal{V} \subseteq \mathit{Var}$ of variables is a finite sequence $\flat \in \{(x,i) \mid i \in \Gamma \text{ and } x \in \mathcal{V}\}^{|\Gamma|}$ of length $|\flat| = |\Gamma|$, such that every agent $i \in \Gamma$ occurs exactly once in $\flat$. However, the same variable $x \in \mathcal{V}$ can occur several times in $\flat$, i.e., intuitively, the same strategy denoted by x can be used by several agents in Γ. The *binding operator* (x,i) means that strategy x is used by agent i. A *quantification prefix* over a set $\mathcal{V} \subseteq \mathit{Var}$ is a finite sequence $\wp \in \{\exists x, \forall x \mid x \in \mathcal{V}\}^{|\mathcal{V}|}$ of length $|\wp| = |\mathcal{V}|$ such that each variable $x \in \mathcal{V}$ occurs in $\wp$ exactly once. The strategy quantifier $\exists x$ (resp. $\forall x$) reads as "for some (resp. every) strategy $x, \ldots$". Let $\mathit{Qnt}(\mathcal{V}) \subset \{\exists x, \forall x \mid x \in \mathcal{V}\}^{|\mathcal{V}|}$ and $\mathit{Bnd}(\Gamma) \subset \{(x,i) \mid i \in \Gamma \text{ and } x \in \mathit{Var}\}^{|\Gamma|}$ denote the sets of all quantification and binding prefixes over variables in $\mathcal{V}$ and agents in Γ. For a given $\wp\flat$, let $E(\wp\flat)$ (resp. $A(\wp\flat)$) be the set of all agents i such that (x,i) and $\exists x$ (resp. $\forall x$) appear in $\wp\flat$ (i.e. the set of agents assigned to strategies under existential (resp. universal) quantification). For example, $\wp = \forall x \exists y \forall z \in \mathit{Qnt}(\{x,y,z\})$, $\flat = (x,1)(x,2)(y,3)(z,4) \in \mathit{Bnd}(\{1,2,3,4\})$ with $\{x,y,z\} \subseteq \mathit{Var}$, and for $\wp\flat = \forall x \exists y \forall z(x,1)(x,2)(y,3)(z,4)$, $E(\wp\flat) = \{3\}$ and $A(\wp\flat) = \{1,2,4\}$.

For a given formula φ, the set $free(\varphi)$ of *free agents/variables* was defined in (Mogavero et al. 2014) as the subset of $\mathcal{A} \cup \mathit{Var}$ containing (i) all agents for which there is no binding after the occurrence of a temporal operator and (ii) all variables for which there is a binding but no quantifications. Formally, $free$: SL[SG] $\rightarrow 2^{\mathcal{A} \cup \mathit{Var}}$ is a function defined as follows: (i) $free(p) \triangleq \emptyset$, where $p \in \mathcal{PV}$; (ii) $free(\neg\varphi) \triangleq free(\varphi)$; (iii) $free(\varphi_1 \wedge \varphi_2) \triangleq free(\varphi_1) \cup free(\varphi_2)$; (iv) $free(\mathsf{X}\,\varphi) \triangleq \mathcal{A} \cup free(\varphi)$; (v) $free(\varphi_1 U \varphi_2) \triangleq \mathcal{A} \cup free(\varphi_1) \cup free(\varphi_2)$; (vi) $free(Qn\varphi) \triangleq free(\varphi) \setminus \{x\}$, where $Qn \in \{\exists x, \forall x : x \in \mathit{Var}\}$; (vii) $free((x,i)\varphi) \triangleq free(\varphi)$, if $i \notin free(\varphi)$, where $i \in \mathcal{A}$ and $x \in \mathit{Var}$; (viii) $free((x,i)\varphi) \triangleq (free(\varphi)\setminus\{i\}) \cup \{x\}$, if $i \in free(\varphi)$, where $i \in \mathcal{A}$ and $x \in \mathit{Var}$.

Definition 3 (SL[SG])**.** *The formulae of* Strategy Logic with Simple Goals *are defined inductively from the set of atomic propositions $\mathcal{PV}$, strategy variables Var, agents $\mathcal{A}$, $\flat \in \mathit{Bnd}(\mathcal{A})$, $\wp \in \mathit{Qnt}(\mathit{free}(\flat\varphi))$:*

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \wp\flat\,\mathsf{X}\,\varphi \mid \wp\flat(\varphi\,\mathsf{U}\,\varphi),$$

According to the syntax rule, the SL[SG] formulae $\wp\flat\,\mathsf{X}\,\varphi$ and $\wp\flat(\varphi\,\mathsf{U}\,\varphi)$ consist of a quantification prefix $\wp$, a binding prefix $\flat$, the temporal operator *next* X or *until* U, and the well-formed (sub)formula φ. Standard abbreviations are used for other temporal operators like *globally* G and *eventually* F. For a given set of all agents $\mathcal{A}$ of MAS, each binding prefix $\flat$ contains every agent of $\mathcal{A}$, and every agent is bound to exactly one variable which is quantified in $\wp$. Therefore, the length of every $\flat$ is $|\mathcal{A}|$. The conditions on $\flat$ and $\wp$ ensure that every SL[SG] formula ψ is a sentence, i.e., $free(\psi) = \emptyset$. Unlike in ATL, the quantifiers over variables referring to strategies can be of different types and can appear in any order. Moreover, two or more agents can be assigned to the same strategy, which is also not expressible in ATL, e.g., $\exists_x(i,x)(j,x)\,\mathsf{X}\,p$. SL[SG] is less expressive than SL, but still allows to express interesting properties. In particular, one can specify and verify distributed systems in which knowledge explicitly represented is the basis for decision making, strategy selection, and implementation. An example of the above is cooperation of autonomous vehicles with the environment in order to avoid collisions. Thus, for each possible environment (e) strategy there must be a vehicle (v) strategy such that no collision occurs for any behavior of other road users (u): $\forall x_e \exists x_v \forall x_u (x_e, e)(x_v, v)(x_u, u)\mathrm{G}(\neg\mathtt{collision})$. Another example is the verification of correctness and fairness of voting protocols, in particular ensuring that coercion in voting, which has a Stackelberg structure, can be avoided. That is to ensure that for every strategy of the coercer (c), there is a strategy of the voter (v) such that no matter what the environment (e) is doing, the voter will vote according to its belief and knowledge without being penalized: $\forall x_c \exists x_v \forall x_e (x_c, c)(x_v, v)(x_e, e)\mathrm{F}(\mathtt{voted} \wedge \neg\mathtt{punished})$ (Belardinelli et al. 2019).

Strategies. Formally, a *memoryfull strategy* for an agent $i \in \mathcal{A}$, or i-strategy, is a function $\sigma : \mathcal{S}t^+ \rightarrow Act_i$[2]. The set of all strategies, for all agents, in model M, is denoted as $\Sigma(M)$. A *joint strategy* $\sigma_{\mathcal{A}}$ assigns a strategy to every agent $i \in \mathcal{A}$. Given a strategy σ for an agent i, if a different agent j is such that the range of strategy σ is a subset of Act_j (i.e. $ran(\sigma) \subseteq Act_j$), then intuitively also agent j can use the strategy σ. We say that such a strategy is *shared* by agents i and j and denote by $shr(x,\varphi)$ the set of agents bounded to the variable x within the formula φ. The set of all strategies shared by agents from $shr(x,\varphi)$ is denoted

[2]$\mathcal{S}t^+$ is the set of non-empty finite sequences of states

$\Sigma(M)_{shr(x,\varphi)}$. Finally, an *assignment* is a function $\chi : \mathit{Var} \cup \mathcal{A} \to \Sigma(M)$ such that for every agent $i \in \mathcal{A}$, $\chi(i)$ is a strategy for i. For $z \in \mathit{Var} \cup \mathcal{A}$ and $\sigma \in \Sigma(M)$, the *variant* χ^z_σ is the assignment that maps z to σ and coincides with χ on all other variables and agents. Given a history $h \in \mathcal{S}t^+$, an assignment χ defines a unique computation $\lambda(h, \chi) = h \xrightarrow{\langle \chi(i)(h) \rangle_{i \in \mathcal{A}}} g_1 \xrightarrow{\langle \chi(i)(h \cdot g_1) \rangle_{i \in \mathcal{A}}} g_2 \ldots$ starting with h and being consistent with χ.

Semantics. Given a model (an interpreted system) M, we inductively define the satisfaction relation $(M, h, \chi) \models \varphi$ where h is a history, φ is a formula, and χ is an assignment. To give interpretation of formulae in a model, we need to distinguish sub-formulae that are not SL[SG] formulae by themselves and give semantics also for them. Let $TF = \{\mathsf{X}\,\varphi, \varphi_1 \,\mathsf{U}\, \varphi_2 : \varphi, \varphi_1, \varphi_2 \in \text{SL[SG]}\ \}$ - be a set of the temporal subformulas of the formulas of SL[SG].

$$\begin{array}{lll}
(M,h,\chi) \models p & \text{iff} & p \in V(last(h)), \text{ for } p \in \mathcal{PV} \\
(M,h,\chi) \models \neg\varphi & \text{iff} & (M,h,\chi) \not\models \varphi \\
(M,h,\chi) \models \varphi_1 \wedge \varphi_2 & \text{iff} & (M,h,\chi) \models \varphi_1 \text{ and } (M,h,\chi) \models \varphi_2 \\
(M,h,\chi) \models \exists x\psi & \text{iff} & \text{there is a strategy } \sigma \in \Sigma(M)_{shr(x,\psi)}, \\
& \multicolumn{2}{l}{(M,h,\chi^x_\sigma) \models \psi, \text{ for } \psi \in \{\wp'\flat\phi, \flat\phi \mid \flat \in \mathit{Bnd}(\mathcal{A}),} \\
& \multicolumn{2}{l}{\phi \in TF, \wp' \in \bigcup_{\mathcal{V} \subseteq free(\flat\phi)} \mathit{Qnt}(\mathcal{V})\}} \\
(M,h,\chi) \models \forall x\psi, & \text{iff} & \text{for every strategy } \sigma \in \Sigma(M)_{shr(x,\psi)}, \\
& \multicolumn{2}{l}{(M,h,\chi^x_\sigma) \models \psi, \text{ for } \psi \in \{\wp'\flat\phi, \flat\phi \mid \flat \in \mathit{Bnd}(\mathcal{A}),} \\
& \multicolumn{2}{l}{\phi \in TF, \wp' \in \bigcup_{\mathcal{V} \subseteq free(\flat\phi)} \mathit{Qnt}(\mathcal{V})\}} \\
(M,h,\chi) \models (x,i)\psi & \text{iff} & (M,h,\chi^i_{\chi(x)}) \models \psi, \\
& \multicolumn{2}{l}{\text{for } \psi \in \{\flat'\phi, \phi \mid \phi \in TF, \flat' \in \bigcup_{\Gamma \subseteq \mathcal{A} \setminus \{i\}} \mathit{Bnd}(\Gamma)\}} \\
(M,h,\chi) \models \mathsf{X}\,\varphi & \text{iff} & (M, \lambda(h,\chi)_{\leq |h|+1}, \chi) \models \varphi, \\
(M,h,\chi) \models \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} & \exists k \geq |h| \text{ s.t. } (M, \lambda(h,\chi)_{\leq k}, \chi) \models \varphi_2 \\
& \multicolumn{2}{l}{\text{and } (M, \lambda(h,\chi)_{\leq l}, \chi) \models \varphi_1, \forall l\text{: } |h| \leq l < k.}
\end{array}$$

Definition 4 (Model, satisfiability)**.** *We write $(M, h) \models \varphi$ iff $(M, h, \chi) \models \varphi$ for every assignment χ and say that interpreted system M is a* model *of an* SL[SG] *formula φ, in symbols $M \models \varphi$, iff $M, \iota \models \varphi$, i.e., φ is true at the initial state (the history of length 1) of the model M. An* SL[SG] *formula φ is* satisfiable *iff there is a model for it.*

We consider the problem of deciding whether a SL[SG] formula is satisfiable under some fixed initial restrictions on MAS, concerning the number of agents, local actions, local states, and local propositions of every agent. Therefore, we deal with the bounded satisfiability problem. Since SL[1G] has the Bounded Model Property (Mogavero et al. 2017), so SL[SG] (a sublogic of SL[1G]), inherits this property. We call this decision problem SL[SG] SAT.

4 Boolean Representation of SL[SG] Models

In this section we show how to encode a MAS and its models using a set of fresh propositional variables. For a given model, the number of variables is fixed. A function is determined that assigns Boolean values to the variables and thus encodes all elements of the model. Then, to find this model, it is enough to determine the valuation function. The procedure looking for this function assigns values to variables step by step. Thus, until the model is found, we are dealing with an incomplete (partial) model. In what follows, we show how we define and encode partial models, and extend them to total models that are an input for a model checker.

4.1 Encoding a MAS

Consider a MAS as defined in Section 1. We assume that for each agent $i \in \mathcal{A}$ the i-local states are numbered from 1 to $|L_i|$, and the joint actions are numbered from 1 to $|Act|$. Below, we define sets of fresh propositional variables: PB_i, TB_i, VB_i for $i \in \mathcal{A}$, and let $VB = \bigcup_{i \in \mathcal{A}} VB_i$ and $\mathcal{PVB} = \bigcup_{i \in \mathcal{A}} (TB_i \cup PB_i \cup VB_i)$. The semantics of these variables is defined by the valuation function: $v_M : \mathcal{PVB} \to \{0, 1\}$, described after the definition of each set of the propositional variables of $\mathcal{PVB}$. Next, we allocate the above mentioned fresh variables:

- For the i-local protocols for $i \in \mathcal{A}$:
  $PB_i = \{pb_i(k,t) \mid 1 \leq k \leq |L_i|, 1 \leq t \leq |Act_i|\}$.
  C1) $v_M(pb_i(k,t)) = 1$ iff the action a^t_i of agent $i \in \mathcal{A}$ is enabled at location l^k_i, i.e., $a^t_i \in P_i(l^k_i)$.
- For the i-local transition functions for $i \in \mathcal{A}$:
  $TB_i = \{tb_i(k,j,k') \mid 1 \leq k, k' \leq |L_i|, 1 \leq j \leq |Act|\}$:
  C2) $v_M(tb_i(k,j,k')) = 1$ iff we have $T_i(l^k_i, \alpha_j) = l^{k'}_i$.
- For the i-local valuation functions for $i \in \mathcal{A}$:
  $VB_i = \{vb_i(k,j) \mid 1 \leq k \leq |L_i|,\ 1 \leq j \leq |\mathcal{PV}_i|\}$.
  C3) $v_M(vb_i(k,j)) = 1$ iff the i-local proposition $p^j_i \in \mathcal{PV}_i$ is true at the local state $l^k_i \in L_i$, i.e., $p^j_i \in V_i(l^k_i)$.

Next, we encode the properties satisfied by the components of a MAS.

A1 For each $i \in \mathcal{A}$ and $l_i \in L_i$: $P_i(l_i) \neq \emptyset$. This is encoded by the following formula $\varphi_1 = \bigwedge_{i \leq n; k \leq n_i} \bigvee_{t \leq m_i} pb_i(k,t)$.

A2 Let $gl_i : \{1, \ldots, |Act_i|\} \mapsto 2^{\{1,\ldots,|Act|\}}$, for $i \in \mathcal{A}$, be a function that a given local action number t assigns a set of numbers of the joint actions s.t. t is the number of their i-th components. That is: $gl_i(t) = \{1 \leq j \leq |Act| \mid \alpha^i_j = a^t_i\}$. For each agent $i \in \mathcal{A}$, $T_i(l_i, \alpha)$ is defined iff $\alpha^i \in P_i(l_i)$. This is encoded by the following formula $\varphi_2 =$

$$\bigwedge_{i \leq n; t \leq m_i; k \leq n_i; j \in gl_i(t)} \Big(\big(\bigvee_{k' \leq n_i} tb_i(k,j,k') \big) \leftrightarrow pb_i(k,t) \Big)$$

A3 For each agent $i \in \mathcal{A}$, T_i is a function. This is encoded by the following formula $\varphi_3 =$

$$\bigwedge_{i \leq n; k,k' \leq n_i; j \leq |Act|} \Big(tb_i(k,j,k') \to \bigwedge_{k'' \leq n_i, k'' \neq k'} \neg tb_i(k,j,k'') \Big).$$

4.2 Encoding Models

A valuation function v_M which satisfies the conditions **C1-C3** and the formula $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$, determine the model M of MAS without an initial state specified. Therefore, v_M actually encodes a family of the models that differ only in the initial state. The set of global states is a product of the local state sets. Then, given some v_M, we can reconstruct the (partial) global transition function T using the following fact: $T(g,a) = g'$ iff $\forall_{i \leq n} v_M(tb_i(g_i, a, g'_i)) = 1$. Finally, the valuation function V can be easily reconstructed using the observation: $p^j_i \in V\big(g\big)$ iff $v_M(vb_i(g_i, j)) = 1$. Thus, searching for a model for any formula ψ can be then reduced to searching for a valuation function v_M.

4.3 Partial MAS and Partial Models

Notice that the i-local components of each agent $i \in \mathcal{A}$ can be rewritten as follows. Let $B = \{0, 1\}$.

- $P_i : L_i \times Act_i \to B$ s.t. $\forall_{l \in L_i} \exists_{a \in Act_i}\ P_i(l, a) = 1$;
- $T_i : L_i \times Act \times L_i \to B$ s.t. $\forall_{l \in L_i, \alpha \in Act}\ (P_i(l, \alpha^i) = 1$ iff $\exists_{l' \in L_i} T_i(l, \alpha, l') = 1)$ and if $T_i(l, \alpha, l') = 1$, then $\forall_{l'' \in L_i : l'' \neq l'} T_i(l, \alpha, l'') = 0$.
- $V_i : L_i \times \mathcal{PV}_i \to B$.

The above conditions are the same as of **A1-A3**. The partial versions PP_i, PT_i, PV_i of the i-local components of each agent $i \in \mathcal{A}$, are defined by extending the set of values B of the functions P_i, T_i, V_i to the set $B_u = \{0, 1, u\}$, where u stands for undefined. Formally:

- a *partial protocol function* $PP_i : L_i \times Act_i \to B_u$,
- a *partial transition* function $PT_i : L_i \times Act \times L_i \to B_u$,
- a *partial valuation function* $PV_i : L_i \times \mathcal{PV}_i \to B_u$.

By a *partial MAS*, denoted MAS$_P$, we mean a MAS in which each agent i is associated with the i-local partial components.

A partial MAS$_P$ induces the partial model M_P, which differs from a model M for a MAS in the global transition function and the global valuation function, called partial, and defined as follows:

- $PT : \mathcal{S}t \times Act \times \mathcal{S}t \to B_u$ such that
  $PT(g, \alpha, g') = 1$ iff $PT_i(g_i, \alpha, g'_i) = 1$ for all $i \in \mathcal{A}$,
  $PT(g, \alpha, g') = u$ iff $PT_i(g_i, \alpha, g'_i) = u$ for some $i \in \mathcal{A}$,
  $PT(g, \alpha, g') = 0$ otherwise,
- $PV : \mathcal{S}t \times \mathcal{PV} \to B_u$ such that $PV((l_1, \ldots, l_n), p) = PV_i(l_i, p)$, where $p \in \mathcal{PV}_i$ for $i \in \mathcal{A}$.

Each partial model M_P can be represented by the partial valuation function $v_{M_P} : \mathcal{PVB} \to B_u$. The goal of our satisfiability algorithm is to find values of B for the undefined variables and to extend v_{M_P} to a total function satisfying the conditions **C1-C3** and the formulae **A1-A3**.

4.4 Construction of $\mathbf{M^{\Gamma}_{over}}$ and $\mathbf{M^{\Gamma}_{under}}$

We construct under and over approximations of partial models. For each partial model M satisfying **A1-A3** and $\Gamma \subseteq \mathcal{A}$, (total) models M^{Γ}_{under} and M^{Γ}_{over} are constructed. First, for every agent $i \in \mathcal{A}$ we define *a necessary local protocol*: $\underline{P_i} : L_i \times Act_i \to B$, *a necessary local transition function* $\underline{T_i} : L_i \times Act \times L_i \to B$, and *a necessary local valuation function* $\underline{V_i} : L_i \times \mathcal{PV}_i \to B$, *a possible local protocol* $\overline{P_i} : L_i \times Act_i \to B$, *a possible local transition function* $\overline{T_i} : L_i \times Act \times L_i \to B$, *a possible local valuation function* $\overline{V_i} : L_i \times \mathcal{PV}_i \to B$, where:
• if $PP_i(l, a) \neq u$, then $\underline{P_i}(l, a) = \overline{P_i}(l, a) = PP_i(l, a)$,
• if $PT_i(l, \alpha, l') \neq u$, then $\underline{T_i}(l, \alpha, l') = \overline{T_i}(l, \alpha, l') = PT_i(l, \alpha, l')$,
• if $PV_i(l, p) \neq u$, then $\underline{V_i}(l, p) = \overline{V_i}(l, p) = PV_i(l, p)$,
• if $PP_i(l, a) = u$, then $\underline{P_i}(l, a) = 0$, $\overline{P_i}(l, a) = 1$,
• if $PT_i(l, \alpha, l') = u$, then $\underline{T_i}(l, \alpha, l') = 0$, $\overline{T_i}(l, \alpha, l') = 1$,
• if $PV_i(l, p) = u$, then $\underline{V_i}(l, p) = 0$, $\overline{V_i}(l, p) = 1$.

Notice that the possible local protocol is an extension of the necessary local protocol, i.e., $\forall_{l \in L_i; a \in Act_i}\ \underline{P_i}(l, a) \leq \overline{P_i}(l, a)$ and the possible local transition function is an extension of the necessary local transition function, i.e., $\forall_{l, l' \in L_i; \alpha \in Act}\ \underline{T_i}(l, \alpha, l') \leq \overline{T_i}(l, \alpha, l')$.

Similarly, total valuations of the propositional variables are defined: *a necessary valuation* $\underline{V} : \mathcal{S}t \times \mathcal{PV} \to B$ and *a possible valuation* $\overline{V} : \mathcal{S}t \times \mathcal{PV} \to B$ such that:
• if $PV(g, p) \neq u$, then $\underline{V}(g, p) = \overline{V}(g, p) = PV(g, p)$,
• if $PV(g, p) = u$, then $\underline{V}(g, p) = 0$, $\overline{V}(g, p) = 1$.
Thus, for every $g \in \mathcal{S}t$, $p \in \mathcal{PV}$ we have $\underline{V}(g, p) \leq \overline{V}(g, p)$.

The (total) model M^{Γ}_{under} is defined as in Def. 2 for the MAS, which consists of n agents, where each agent $i \in \Gamma$ is associated with $AG_i = (L_i, \iota_i, Act_i, \underline{P_i}, \underline{T_i}, \mathcal{PV}_i, \underline{V_i})$, and each agent $j \in \mathcal{A} \setminus \Gamma$ is associated with $AG_j = (L_j, \iota_j, Act_j, \overline{P_j}, \overline{T_j}, \mathcal{PV}_j, \overline{V_j})$. The (total) model M^{Γ}_{over} is defined as in Def. 2 for the MAS, which consists of n agents, where each agent $i \in \Gamma$ is associated with $AG_i = (L_i, \iota_i, Act_i, \overline{P_i}, \overline{T_i}, \mathcal{PV}_i, \overline{V_i})$, and each agent $j \in \mathcal{A} \setminus \Gamma$ is associated with $AG_j = (L_j, \iota_j, Act_j, \underline{P_j}, \underline{T_j}, \mathcal{PV}_j, \underline{V_j})$.

Definition 5. *Let $M = (\mathcal{S}t, \iota, PT, PV)$ and $M' = (\mathcal{S}t, \iota, PT', PV')$ be two (partial) models for the partial MAS and MAS', consisting of n agents, where each agent $i \in \mathcal{A}$ is associated with $AG_i = (L_i, \iota_i, Act_i, PP_i, PT_i, \mathcal{PV}_i, PV_i)$ and $AG'_i = (L_i, \iota_i, Act_i, PP'_i, PT'_i, \mathcal{PV}_i, PV'_i)$, resp. We say that M* extends *M' iff the following conditions hold:*

- *if $PP'_i(l, a) \neq u$, then $PP_i(l, a) = PP'_i(l, a)$,*
- *if $PT'_i(l, \alpha, l') \neq u$, then $PT_i(l, \alpha, l') = PT'_i(l, \alpha, l')$,*
- *if $PV'(g, p) \neq u$, then $PV(g, p) = PV'(g, p)$.*

Notice that for a given partial model M and $\Gamma \subseteq \mathcal{A}$, both total models M^{Γ}_{under} and M^{Γ}_{over} extend M.

Theorem 1. *Consider two partial models M and M' s.t. M extends M', and let $\Gamma \subseteq \mathcal{A}$. The following conditions hold:*
• $v_{(M')^{\Gamma}_{under}}(b_i) \leq v_{M^{\Gamma}_{under}}(b_i) \leq v_{M^{\Gamma}_{over}}(b_i) \leq v_{(M')^{\Gamma}_{over}}(b_i)$ *for $b_i \in TB_i \cup PB_i$ with $i \in \Gamma$,*
• $v_{(M')^{\Gamma}_{under}}(b_i) \geq v_{M^{\Gamma}_{under}}(b_i) \geq v_{M^{\Gamma}_{over}}(b_i) \geq v_{(M')^{\Gamma}_{over}}(b_i)$ *for $b_i \in TB_i \cup PB_i$ with $i \in \mathcal{A} \setminus \Gamma$,*
• $v_{(M')^{\Gamma}_{under}}(vb) \leq v_{M^{\Gamma}_{under}}(vb) \leq v_{M^{\Gamma}_{over}}(vb) \leq v_{(M')^{\Gamma}_{over}}(vb)$ *for $vb \in VB$.*

Proof. From the definitions of *over* and *under* models. □

5 Monotonic Theory for SL[SG]

Our satisfiability algorithm for SL[SG] applies the technique used earlier for monotonic theories for checking both CTL satisfiability (Bayless et al. 2015; Klenze, Bayless, and Hu 2016) and ATL satisfiablity (Kacprzak, Niewiadomski, and Penczek 2020). Although SL[SG] is not a monotone theory, we show how the same technique can be applied in this case. In what follows, specific cases of positive or negative monotonicity are described. These properties are used to determine the class of models to which the model of a given formula belongs. This makes it possible to substantially limit the set of all potential models, only to models belonging to this class and significantly reduce the search space for optimizing the satisfiability procedure.

5.1 Boolean Monotonic Theory

Consider a predicate $P : B^n \mapsto B$. We say that P is Boolean *positive monotonic* iff $P(s_1, \ldots, s_{i-1}, 0, s_{i+1}, \ldots, s_n) \leq P(s_1, \ldots, s_{i-1}, 1, s_{i+1}, \ldots, s_n)$, for $1 \leq i \leq n$.
P is Boolean *negative monotonic* iff $P(s_1, \ldots, s_{i-1}, 1, s_{i+1}, \ldots, s_n) \leq P(s_1, \ldots, s_{i-1}, 0, s_{i+1}, \ldots, s_n)$, for $1 \leq i \leq n$.
The definition of (positive and negative Boolean) monotonicity for a function $F : B^n \mapsto 2^S$ (for some set S) is analogous. F is Boolean *positive monotonic* iff $F(s_1, \ldots, s_{i-1}, 0, s_{i+1}, \ldots, s_n) \subseteq F(s_1, \ldots, s_{i-1}, 1, s_{i+1}, \ldots, s_n)$, for $1 \leq i \leq n$. A function F is Boolean *negative monotonic* iff $F(s_1, \ldots, s_{i-1}, 1, s_{i+1}, \ldots, s_n) \subseteq$
$F(s_1, \ldots, s_{i-1}, 0, s_{i+1}, \ldots, s_n)$, for $1 \leq i \leq n$.

Definition 6 (Boolean Monotonic Theory). *A theory T with a signature $\Omega = (S, S_f, S_r, Ar)$, where S is a non-empty set of elements called sorts, S_f is a set of function symbols, S_r is a set of relation symbols, and Ar is the arity of the relation and function symbols, is (Boolean) monotonic iff all sorts in Ω are Boolean, and all predicates and functions in Ω are monotonic.*

5.2 Monotonicity Property

The satisfaction relation $\models$ for SL[SG] is not monotonic, but satisfies the condition of monotonicity in some special cases.

Theorem 2. *Let M and M' be two models defined over the same set of agents $\mathcal{A}$, for two MASes that share the same sets $L_i, Act_i, \mathcal{PV}_i$ for each $i \in \mathcal{A}$. Then, we have $(M, h, \chi) \models \phi$ implies $(M', h, \chi) \models \phi$, for any h and χ, if the following conditions hold:*

(H1) *$\phi \in \{p, p_1 \wedge p_2\}$ for $p, p_1, p_2 \in \mathcal{PV}$ and $v_M(vb) \leq v_{M'}(vb)$ for each $vb \in VB$,*

(H2) *$\phi = \neg p$ for $p \in \mathcal{PV}$ and $v_M(vb) \geq v_{M'}(vb)$ for each $vb \in VB$,*

(H3) *$\phi \in \{\wp\flat \mathsf{X} p, \wp\flat(p_1 \mathsf{U} p_2)\}$ for $p, p_1, p_2 \in \mathcal{PV}$ and*
(H3.1) $v_M(vb) \leq v_{M'}(vb)$ for each $vb \in VB$, and
(H3.2) $v_M(b_i) \leq v_{M'}(b_i)$ for each $b_i \in TB_i \cup PB_i$ and $i \in E(\wp\flat)$, and
(H3.3) $v_M(b_i) \geq v_{M'}(b_i)$ for each $b_i \in TB_i \cup PB_i$ and $i \in A(\wp\flat)$.

Proof. Let M and M' be two models defined over the same set of agents $\mathcal{A}$, for two multi-agent systems that share the same sets $L_i, Act_i, \mathcal{PV}_i$ for each $i \in \mathcal{A}$.
1. Let $\phi = p_1 \wedge p_2$, h be a history, χ be an assignment, and $p_1 \in \mathcal{PV}_i$, $p_2 \in \mathcal{PV}_j$ for some $i, j \in \mathcal{A}$. Assume that $(M, h, \chi) \models p_1 \wedge p_2$, i.e. $p_1, p_2 \in V(last(h))$. Thus $v_M(vb_i(last(h)_i, p_1)) = v_M(vb_j(last(h)_j, p_2)) = 1$. On the other hand, assume that $(M', h, \chi) \not\models p_1 \wedge p_2$, i.e. $p_1 \notin V'(last(h))$ or $p_2 \notin V'(last(h))$. Thus $v_{M'}(vb_i(last(h)_i, p_1)) = 0$ or $v_{M'}(vb_j(last(h)_j, p_2)) = 0$. Therefore, $v_M(vb_i(last(h)_i, p_1)) > v_{M'}(vb_j(last(h)_j, p_1))$ or $v_M(vb_i(last(h)_i, p_2)) > v_{M'}(vb_j(last(h)_j, p_2))$, what contradicts (H1). Finally, $(M, h, \chi) \models \phi$ implies $(M', h, \chi) \models \phi$. The proof for $\phi = p$ is analogous.
2. Let $\phi = \neg p$, h be a history, χ be an assignment, and $p \in \mathcal{PV}_i$ for some $i \in \mathcal{A}$. Assume that $(M, h, \chi) \models \neg p$, i.e. $p \notin V(last(h))$. Thus $v_M(vb_i(last(h)_i, p)) = 0$. On the other hand, assume that $(M', h, \chi) \not\models \neg p$, i.e. $p \in V'(last(h))$. Thus $v_{M'}(vb_i(last(h)_i, p)) = 1$. Therefore, $v_M(vb_i(last(h)_i, p)) < v_{M'}(vb_i(last(h)_i, p))$, what contradicts (H2). Finally, $(M, h, \chi) \models \phi$ implies $(M', h, \chi) \models \phi$.
3. Let $\phi = \wp\flat \mathsf{X} p$, h be a history, χ be an assignment, and $p \in \mathcal{PV}_i$ for some $i \in \mathcal{A}$. Denote by EV (AV) the set of all variables that appear under existential (universal, resp.) quantifier in ϕ. Assume that $(M, h, \chi) \models \phi$. Thus for every $x \in EV$ there exists a strategy σ, and for every $y \in AV$ for every strategy σ', $(M, h, \chi') \models \mathsf{X} p$ for χ' being an assignment that maps x to σ, i to σ for every agent $i \in shr(x, \phi)$, y to σ', j to σ' for every agent $j \in shr(y, \phi)$, and coincides with χ on all other variables. Thus $(M, \lambda(h, \chi')_{\leq |h|+1}, \chi') \models p$, i.e. $p \in V(last(\lambda(h, \chi')_{\leq |h|+1}))$.
On the other hand, assume that $(M', h, \chi) \not\models \phi$. Thus
(*) in both models M and M' there exist the same strategies and $last(\lambda(h, \chi')_{\leq |h|+1}) = last(\lambda'(h, \chi')_{\leq |h|+1}) = g$, but $p \in V(g)$ and $p \notin V'(g)$, or
(**) in both models M and M' there exist the same strategies, but $last(\lambda(h, \chi')_{\leq |h|+1}) \neq last(\lambda'(h, \chi')_{\leq |h|+1})$, or
(***) the set of strategies that can be assigned to agents $i \in E(\wp\flat)$ in M (denoted $\Sigma(E(\wp\flat))(M)$) is a subset of the set strategies that can be assigned to these agents in M' (denoted $\Sigma(E(\wp\flat))(M')$), and the set of strategies that can be assigned to agents $j \in A(\wp\flat)$ in M (denoted $\Sigma(A(\wp\flat))(M)$) is a subset of strategies that can be assigned to these agents in M' (denoted $\Sigma(A(\wp\flat))(M')$) (notice that if $\Sigma(A(\wp\flat))(M') \subseteq \Sigma(A(\wp\flat))(M)$ the condition $(M', h, \chi) \models \phi$ holds), or
(****) $\Sigma(E(\wp\flat))(M') \subset \Sigma(E(\wp\flat))(M)$, i.e., the set of strategies that can be assigned to agents $i \in E(\wp\flat)$ in M' is a proper subset of the set of strategies that can be assigned to these agents in M. Let us consider all the above cases.
If (*) then $v_M(vb_i((last(\lambda(h, \chi')_{\leq |h|+1}))_i, p)) > v_{M'}(vb_i((last(\lambda'(h, \chi')_{\leq |h|+1}))_i, p))$, what contr. (H3.1).
If (**) then
(a) for some $i \in E(\wp\flat)$, let $l_i = last(\lambda(h, \chi')_{\leq |h|+1})_i$ and $l'_i = last(\lambda'(h, \chi')_{\leq |h|+1})_i$. Then, $v_M(tb_i(last(h)_i, \alpha, l_i)) = 1$ and $v_{M'}(tb_i(last(h)_i, \alpha, l'_i)) = 1$ for a global action α s.t. α^k for $k \in \mathcal{A}$ is determined by the strategy assigned to k by χ'. Thus from **A1-A3** (action determinism) $v_{M'}(tb_i(last(h)_i, \alpha, l_i)) = 0$. Therefore, $v_M(tb_i(last(h)_i, \alpha, l_i)) > v_{M'}(tb_i(last(h)_i, \alpha, l_i))$ what contradicts (H3.2).
(b) for some $j \in A(\wp\flat)$, let $l_j = last(\lambda(h, \chi')_{\leq |h|+1})_j$ and $l'_j = last(\lambda'(h, \chi')_{\leq |h|+1})_j$. Then, $v_M(tb_j(last(h)_j, \alpha, l_j)) = 1$ and $v_{M'}(tb_j(last(h)_j, \alpha, l'_j)) = 1$ for a global action α

s.t. α^k for $k \in \mathcal{A}$ is determined by the strategy assigned to k by χ'. Thus from **A1-A3** (action determinism) $v_M(tb_j(last(h)_j, \alpha, l'_j)) = 0$. Therefore, $v_M(tb_j(last(h)_j, \alpha, l'_j)) < v_{M'}(tb_j(last(h)_j, \alpha, l'_j))$ what contradicts (H3.3).
If (***) then for some $j \in A(\wp\flat)$ there exists a strategy σ_j in M' which does not exist in M. It means that there exists an action α_j^k which is possible to execute in $last(h)_j$ in M' but not in M. Thus, $v_M(pb_j(last(h)_j, k)) = 0$ and $v_{M'}(pb_j(last(h)_j, k)) = 1$. Therefore, $v_M(pb_j(last(h)_j, k)) < v_{M'}(pb_j(last(h)_j, k))$ what contradicts (H3.3).
If (****) then for some $i \in E(\wp\flat)$ there exists a strategy σ_i in M which does not exist in M'. It means that there exists an action α_i^k which is possible to execute in $last(h)_i$ in M but not in M'. Thus, $v_M(pb_i(last(h)_i, k)) = 1$ and $v_{M'}(pb_i(last(h)_i, k)) = 0$. Therefore, $v_M(pb_i(last(h)_i, k)) > v_{M'}(pb_i(last(h)_i, k))$ what contradicts (H3.2).
Finally, $(M, h, \chi) \models \phi$ implies $(M', h, \chi) \models \phi$.
The proof for $\wp\flat(p_1 \mathsf{U} p_2)$ is analogous. □

It follows from Thm 2 that the relation $\models$ is positive monotonic wrt. VB for $\phi \in \{p, p_1 \wedge p_2, \wp\flat \mathsf{X} p, \wp\flat(p_1 \mathsf{U} p_2)\}$, negative monotonic wrt. VB for $\phi \in \{\neg p\}$, positive monotonic wrt. TB_i and PB_i for $\phi \in \{\wp\flat \mathsf{X} p, \wp\flat(p_1 \mathsf{U} p_2)\}$ and $i \in E(\wp\flat)$, negative monotonic wrt. TB_i and PB_i for $\phi \in \{\wp\flat \mathsf{X} p, \wp\flat(p_1 \mathsf{U} p_2)\}$ and $i \in A(\wp\flat)$.

5.3 Function MC: Model Checking

Let $\|\phi\|_{\mathrm{M}}$ denote the set of all states of M (i.e., histories of length 1), where ϕ holds. In order to compute $\|\phi\|_{\mathrm{M}}$, the evaluation function $MC(op, Z_1, V_m)$ is defined for an unary operator *op*, and $MC(op, Z_1, Z_2, V_m)$ for a binary operator *op*, where $Z_1, Z_2 \subseteq \mathcal{S}t$, and V_m is a valuation encoding a model. The function MC evaluates the operator *op* on the sets of states Z_1, Z_2 rather than the formulae holding in these states. If $\phi = p \in \mathcal{PV}$, then $\|p\|_{\mathrm{M}}$ returns the set of states of M in which p holds, i.e., $\{g \in \mathcal{S}t \mid p \in V(g)\}$. Otherwise, $\|\phi\|_{\mathrm{M}}$ takes the top-most operator *op* of ϕ and solves its argument(s) recursively using the function MC and applying $MC(op, Z_1, v_M)$ or $MC(op, Z_1, Z_2, v_M)$ to the returned set(s) of states. Notice that $MC(\neg, Z, v_M)$ returns $\mathcal{S}t \setminus Z$. Similarly, $MC(\wedge, Z_1, Z_2, v_M)$ returns $Z_1 \cap Z_2$. If $op \in \{\wp\flat \mathsf{X}, \wp\flat \mathsf{U}\}$, then the model checking algorithm of (Belardinelli et al. 2019) is used. The algorithm manipulates sets of states in $\mathcal{S}t$ that is inspired by the standard model checking algorithm for ATL. It works bottom-up on the structure of a formula. Its soundness and completeness is guaranteed by the fixed-point characterisation of SL[SG]. Since this algorithm works on sets of states, our approximation procedure also works on sets of states. One can show that for SL[SG] with complete information both semantics (defined on states and histories) are equivalent.

Now, Thm 2 can be rewritten by replacing the propositional variables by the sets of states these variables hold at.

Theorem 3. *The functions $MC(op, Z_1, V_m)$ and $MC(op, Z_1, Z_2, V_m)$ are:*

1. *negative monotonic w.r.t. VB and positive and negative monotonic w.r.t. PB_i and TB_i for $i \in \mathcal{A}$, for $op = \neg$,*
2. *positive monotonic w.r.t. VB and positive and negative monotonic w.r.t. PB_i and TB_i for $i \in \mathcal{A}$, for $op = \wedge$,*
3. *positive monotonic w.r.t. VB, positive monotonic w.r.t. PB_i and TB_i for $i \in E(\wp\flat)$, negative monotonic w.r.t. PB_i and TB_i for $i \in A(\wp\flat)$, for $op \in \{\wp\flat \mathsf{X}, \wp\flat \mathsf{U}\}$.*

6 Approximations

Next we show a technique for constructing an approximation for a given incomplete model derived from the initial requirements. This technique uses partial model extension methods and monotonicity properties, described earlier. It allows to determine the class of possible extensions of a partial model including all models satisfying the given formula.

For a given partial model M' and a formula ϕ two approximations are computed:

- approximation of models - it is a set of models, denoted $CLASS_{M'}(\phi)$, such that if M is a total extension of M' and $M, s \models \phi$ for some $s \in \mathcal{S}t$, then $M \in CLASS_{M'}(\phi)$;
- states - it is a set of states, denoted by $\|\phi\|_{M'}$, containing all the states satisfying ϕ for some model of $CLASS_{M'}(\phi)$ i.e., $\{s \in \mathcal{S}t \mid \exists M \in CLASS_{M'}(\phi)$ s.t. $M, s \models \phi\} \subseteq \|\phi\|_{M'}$.
  Notice that the above notion $\|\phi\|_{M'}$ for partial models M' extends the same notion defined before for models.

Since $CLASS_{M'}(\phi)$ and $\|\phi\|_{M'}$ are only approximations, models and states that do not satisfy the formula can also be included. Furthermore, for a fixed partial model M', a formula ϕ, and an initial state ι, it follows directly from the definitions that if M' can be extended to a total model M s.t. $M, \iota, \chi \models \phi$, then $M \in CLASS_{M'}(\phi)$ and $\iota \in \|\phi\|_{M'}$. The approximations are used in the satisfiability procedure we provide, for the early detection of conflicts and thus for finding the model faster. If the fixed initial state does not belong to the determined state approximation, then the model class determined by this approximation can be rejected, which greatly improves the efficiency of the procedure.

6.1 Algorithm Apx

The approximation of states is computed by the algorithm *Apx* - see Algorithm 1. Its input contains: an SL[SG] formula ϕ, the partial model M for a partial MAS, and a parameter λ, where $\lambda \in \{over, under\}$. The algorithm returns the λ-approximation of a set of states satisfying ϕ. In what follows, we use the following notation: $\overline{\lambda} = over$ for $\lambda = under$, and $\overline{\lambda} = under$ for $\lambda = over$.

If $\phi = p$ is a propositional variable, then $Apx(\phi, M, \lambda)$ returns a set of states satisfying p in the model $M_\lambda^{\mathcal{A}}$. If $\phi = \neg\varphi$, then $Apx(\phi, M, \lambda)$ computes the compliment of the approximation $Apx(\varphi, M, \overline{\lambda})$ of φ. If $\phi = \wp\flat \mathsf{X} \varphi$, then $Apx(\phi, M, \lambda)$ computes the approximation for φ performing $Z = Apx(\varphi, M, \lambda)$ and the function MC determines the set of states satisfying $\wp\flat \mathsf{X}(\kappa_Z)$ in $M_\lambda^{E(\wp\flat)}$, where κ_Z is a fresh propositional variable that holds in the states of Z. In fact, MC has a set of states as its input, not a formula. This

Algorithm 1 Algorithm *Apx*

```
Input: $\phi, \lambda, M$ for a partial MAS
Output: a set of states of $M$
 1: if $\phi \in \mathcal{PV}$ then
 2:     return $\{g \in \mathcal{S}t \mid M_\lambda^{\mathcal{A}}, g \models \phi\}$
 3: else if $\phi = op(\psi)$ then
 4:     if $op = \neg$ then    // negative monotonic
 5:         $Z := Apx(\psi, M, \overline{\lambda})$
 6:         return $\mathcal{S}t \setminus Z$
 7:     else    // op = $\wp\flat$ X
 8:         $Z := Apx(\psi, M, \lambda)$
 9:         return $MC(op, Z, v_{M_\lambda^{E(\wp\flat)}})$
10: else if $\phi = op(\psi_1, \psi_2)$ for $op \in \{\wp\flat\, \mathsf{U}, \wedge\}$ then
11:     $Z_1 := Apx(\psi_1, M, \lambda)$
12:     $Z_2 := Apx(\psi_2, M, \lambda)$
13:     if $op = \wp\flat\, \mathsf{U}$ then
14:         return $MC(op, Z_1, Z_2, v_{M_\lambda^{E(\wp\flat)}})$
15:     else    // op = $\wedge$
16:         return $Z_1 \cap Z_2$
```

technique allows for computing approximations for nested formulae, where the subformula φ is any SL[SG] formula, not just a propositional formula. For the remaining operators, the algorithm works similarly. The *Apx* algorithm has the following property.

Theorem 4. *Let M, M' be two partial models such that M extends M'. Then, $Apx(\phi, M, over) \subseteq Apx(\phi, M', over)$ and $Apx(\phi, M', under) \subseteq Apx(\phi, M, under)$.*

Proof. By a structural induction on a formula ϕ.
The base case. For $\phi = p \in \mathcal{PV}$, from the algorithm definition, $Apx(p, M, over)$ returns the set of the states satisfying p in $M_{over}^{\mathcal{A}}$ and $Apx(p, M', over)$ returns the set of the states satisfying p in $(M')_{over}^{\mathcal{A}}$. So, $Apx(\phi, M, over) \subseteq Apx(\phi, M', over)$ since $v_{M_{over}^{\mathcal{A}}}(vb) \leq v_{(M')_{over}^{\mathcal{A}}}(vb)$, for any $vb \in VB$, from Thm 1. Similarly, $Apx(\phi, M', under) \subseteq Apx(\phi, M, under)$ since $v_{(M')_{under}^{\mathcal{A}}}(vb) \leq v_{M_{under}^{\mathcal{A}}}(vb)$, for any $vb \in VB$, from Thm 1.
The induction step. We show the proof for the unary operators $\neg$, $\wp\flat\, \mathsf{X}$. The proofs for $\wp\flat\, \mathsf{U}$ and $\wedge$ are similar.
Induction assumption (IA): the thesis holds for a formula ψ.
Induction hypothesis (IH): the thesis holds for $\phi = op(\psi)$.
• Let $\phi = \neg\psi$. If $Z = Apx(\psi, M, under)$ and $Z' = Apx(\psi, M', under)$, then $Z' \subseteq Z$ from IA. Thus, $\overline{Z} \subseteq \overline{Z'}$ and consequently $Apx(\phi, M, over) \subseteq Apx(\phi, M', over)$ from def. of *Apx*.
If $Z = Apx(\psi, M, over)$ and $Z' = Apx(\psi, M', over)$, then $Z \subseteq Z'$ from IA. Thus, $\overline{Z}' \subseteq \overline{Z}$ and consequently $Apx(\phi, M', under) \subseteq Apx(\phi, M, under)$, from def. of *Apx*.
• If $\phi = op(\psi)$ with $op = \wp\flat X$. Let $\Gamma = E(\wp\flat)$.
If $Z = Apx(\psi, M, over)$ and $Z' = Apx(\psi, M', over)$, then $Z \subseteq Z'$ from IA. Observe that $Apx(\phi, M, over) = MC(op, Z, v_{M_{over}^{\Gamma}})$ from def. of *Apx*. Similarly, $Apx(\phi, M', over) = MC(op, Z', v_{(M')_{over}^{\Gamma}})$. Since $Z \subseteq Z'$ we have $MC(op, Z, v_{M_{over}^{\Gamma}}) \subseteq MC(op, Z', v_{M_{over}^{\Gamma}})$. Next, $MC(op, Z', v_{M_{over}^{\Gamma}}) \subseteq MC(op, Z', v_{(M')_{over}^{\Gamma}})$ from Thm 1 and Thm 3.3. Finally $MC(op, Z, v_{M_{over}^{\Gamma}}) \subseteq MC(op, Z', v_{(M')_{over}^{\Gamma}})$ and $Apx(\phi, M, over) \subseteq Apx(\phi, M', over)$, from def. of *Apx*.
If $Z = Apx(\psi, M, under)$ and $Z' = Apx(\psi, M', under)$, then $Z' \subseteq Z$ from IA. Observe that $Apx(\phi, M, under) = MC(op, Z, v_{M_{under}^{\Gamma}})$. Similarly, $Apx(\phi, M', over) = MC(op, Z', v_{(M')_{under}^{\Gamma}})$. Since $Z' \subseteq Z$ we have $MC(op, Z', v_{(M')_{under}^{\Gamma}}) \subseteq MC(op, Z, v_{(M')_{under}^{\Gamma}})$. Next $MC(op, Z, v_{(M')_{under}^{\Gamma}}) \subseteq MC(op, Z, v_{M_{under}^{\Gamma}})$ from Thm 1 and Thm 3.3. Finally $MC(op, Z', v_{(M')_{under}^{\Gamma}}) \subseteq MC(op, Z, v_{M_{under}^{\Gamma}})$ and $Apx(\phi, M', under) \subseteq Apx(\phi, M, under)$ from def. of *Apx*. □

6.2 Approximation of States

For a total model M extending a partial model M', the algorithm *Apx* computes over and under-approximation of $\|\phi\|_M$. More precisely, $Apx(\phi, M', over)$ returns a set of states, which is an over-approximation of $\|\phi\|_M$. This means that if $\iota \in \|\phi\|_M$, then $\iota \in Apx(\phi, M', over)$. So, if $\iota \notin Apx(\phi, M', over)$, then there is no model M extending M' s.t. $M, \iota \models \phi$. Similarly, $Apx(\phi, M', under)$ computes an under-approximation of $\|\phi\|_M$, i.e., if $\iota \in Apx(\phi, M', under)$, then $\iota \in \|\phi\|_M$.

Theorem 5. *Let M' be a partial model and M be a total model extending M'. Then, for a formula ϕ we have: $Apx(\phi, M', under) \subseteq \|\phi\|_M \subseteq Apx(\phi, M', over)$.*

Proof. Follows from Thm 4 and that for a total model M: $Apx(\phi, M, under) = Apx(\phi, M, over) = \|\phi\|_M$. □

7 Satisfiability Procedure

The *Axp* algorithm and its property of Theorem 5 are used below to provide a new method for testing the SAT of SL[SG] formulae and constructing their models.

7.1 Satisfiability Algorithm

The goal of the algorithm is to build a model for a given formula. Initially, the model requirements are established: the number of agents, the number of local actions of every agent, the number of local states of every agent, the number of local propositions of every agent, other requirements such as conditions **A1-A3**.

These requirements are given at the entry to the algorithm along with the SL[SG] formula ϕ and the initial state ι.
Input: ϕ, ι, and model requirements determining partial model M.
Output: total model M s.t. $M, \iota \models \phi$, meeting the model requirements or the answer that such a model does not exist.
Steps of the algorithm: Let *depth* be an integer variable tracking the decision depth of the solver.
1. *depth* := 0; set v_M and check whether v_M is consistent with the conditions **A1-A3**, if yes, then go to step 2, if no, then return UNSAT
2. compute $Apx(\phi, M, over)$

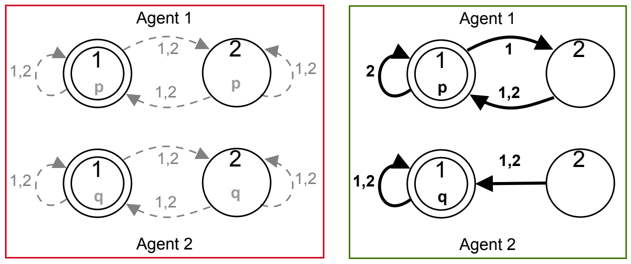

Figure 2: Possible MASes at the start of the algorithm (left), and the final one at the end (right).

3. if $\iota \in Apx(\phi, M, over)$, then
3.1 if M is total; SAT; return M
3.2 otherwise $depth := depth + 1$; assign value to an unsigned variable (the assignment must be consistent with the conditions **A1-A3**); a new partial model M is determined, restricting the class of the considered models; go to step 2
4. if $\iota \notin Apx(\phi, M, over)$, then
4.1 if $depth > 0$, analyse the conflict, undo decisions up to the conflict depth c and assign the opposite value to the conflicting variable; $depth := c$; a new partial model M is determined; go to step 2
4.2 if $depth = 0$ return UNSAT.

The main idea behind the algorithm is to build a model step by step. The models are encoded with Boolean variables, so finding a model consists in determining their values. Some of them are determined by the initial model requirements and allow for establishing a certain class of models (model approximation). Using the monotonicity property (see Thm 4), the algorithm verifies whether the model in question may belong to this class. If so, the procedure narrows down the class of models. If not, it designates a new model class. The algorithm stops when either the model satisfying ϕ is built or there is no model class to which it may belong. If a model satisfying ϕ exists, then after the algorithm stops, the model approximation contains only a model satisfying ϕ and the state approximation contains all states of this model that satisfy ϕ (see Thm 5).

Example 2. *Consider a MAS consisting of 2 agents, 2 local states and actions, and one local proposition per each agent. When the algorithm starts, none of the variables encoding local protocols, transitions, and propositions are assigned. Thus, all transitions and proposition valuations are possible at this step, as indicated by grey elements in Fig. 2 (left). Fig. 2 (right) presents the result of our algorithm - a MAS inducing a model for the formula φ_1 from Sec. 7.3. In this case all PB_1 and PB_2 variables has been assigned $true$ and thus, for every agent and local state, there is a transition labelled by every possible local action.*

7.2 SGSAT Tool

We have implemented our algorithm on the top of MonoSat (Bayless et al. 2015) platform obtaining a prototype tool SGSAT confirming our theoretical results. The core of the system is a modified MiniSat solver (Eén and Sörensson 2003) extended to enable an implementation of monotonic theories, which results in an application similar to lazy-SMT solvers (Sebastiani 2007). In this case, we implemented the SL[SG] theory solver which makes use of an external STV model-checker (Kurpiewski, Jamroga, and Knapik 2019). Our tool reads model requirements and allocates SAT-solver variables (encoding a set of MASes) that are tracked by the theory solver. The SAT core, according to the CDCL algorithm, proposes assignments to the successive variables. The obtained values are propagated by the theory solver, which (using the STV tool) checks if the proposed valuation is consistent. If so, we continue assigning values to the variables until we have a complete model. If not, we get a conflict (a partial model that cannot be extended to a satisfactory valuation) which is analysed, the incorrect decisions are backtracked, and the solver proposes another assignment.

					φ_1		φ_2	
n	**ls**	**la**	**lp**	**vars**	**satT**	**runT**	**satT**	**runT**
2	2	2	2	48	0.05	1.69	0.05	1.14
2	2	5	2	228	0.43	6.08	0.4	6.26
3	2	3	2	354	3.88	29.9	3.83	29.6
3	2	5	2	1542	66.9	607	79.8	708
4	2	2	2	288	13.6	96.4	13.9	101
4	2	3	2	1336	257	1700	270	2462
5	2	2	2	680	501	4121	423	3397

Table 1: Preliminary experimental results. The column meaning, from left, the number of: agents, local states, local actions, local propositions, and variables encoding MAS. Next, the time consumed by SAT-solver, and the total runtime (in seconds).

7.3 Experimental Results

Table 1 shows the preliminary experimental results. These are the very first results of testing satisfiability for SL[SG] formulae. The experiments have been performed on a PC with 16GB RAM and i7-1065G7 CPU running Linux OS, and involved testing satisfiability of two formulae within time limit of 70 minutes: $\varphi_1 = \exists x_1 ... \exists x_n (x_1, 1) ... (x_n, n) \mathrm{F}(p_1^1 \wedge ... \wedge p_n^1)$, and $\varphi_2 = \forall x_1 \exists x_2 ... \exists x_n (x_1, 1) ... (x_n, n) \mathrm{F}(p_1^1 \wedge ... \wedge p_n^1)$. The difference between runtime and SAT-time is the time consumed by the STV tool. Thus, the overall performance depends primarily on the model checker. However, there is still some space for optimizing the SGSAT.

8 Conclusions

A new method and the tool SGSAT for checking satisfiability for SL[SG] have been defined and implemented. Since this is the only tool for testing satisfiability of SL[SG] and SL is general, we could not compare the experimental results with others. Our future plan is to extend our approach to larger fragments of SL, in particular for SL with One Goal and SL with epistemic operators and incomplete information. As the technique we introduced involves an external model-checker, its efficiency and applicability depends on the capabilities of the model-checker applied.

Acknowledgments

The work of M. Kacprzak was supported by the Bialystok University of Technology, as part of the research grant W/WI-IIT/1/2019 of the Faculty of Computer Science and funded by Ministry of Science and Higher Education, Poland. W. Penczek acknowledges support from Luxembourg/Polish FNR/NCBiR project STV and CNRS/PAS project PARTIES.

References

Acar, E.; Benerecetti, M.; and Mogavero, F. 2019. Satisfiability in strategy logic can be easier than model checking. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2638–2645. AAAI Press.

Ågotnes, T.; Goranko, V.; and Jamroga, W. 2007. Alternating-time temporal logics with irrevocable strategies. In Samet, D., ed., *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007), Brussels, Belgium, June 25-27, 2007*, 15–24.

Alur, R.; de Alfaro, L.; Grosu, R.; Henzinger, T. A.; Kang, M.; Kirsch, C. M.; Majumdar, R.; Mang, F. Y. C.; and Wang, B. 2001. JMOCHA: A model checking tool that exploits design structure. In Müller, H. A.; Harrold, M. J.; and Schäfer, W., eds., *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, 12-19 May 2001, Toronto, Ontario, Canada*, 835–836. IEEE Computer Society.

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

Bayless, S.; Bayless, N.; Hoos, H.; and Hu, A. 2015. SAT modulo monotonic theories. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 3702–3709. AAAI Press.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2017. Verification of broadcasting multi-agent systems against an epistemic strategy logic. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 91–97. ijcai.org.

Belardinelli, F.; Jamroga, W.; Kurpiewski, D.; Malvone, V.; and Murano, A. 2019. Strategy logic with simple goals: Tractable reasoning about strategies. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 88–94. ijcai.org.

Cermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2014. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In Biere, A., and Bloem, R., eds., *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, 525–532. Springer.

Cermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2018. Practical verification of multi-agent systems against SLK specifications. *Inf. Comput.* 261(Part):588–614.

Cermák, P.; Lomuscio, A.; and Murano, A. 2015. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, 2038–2044. AAAI Press.

Eén, N., and Sörensson, N. 2003. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, 502–518. Springer.

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

Jamroga, W., and Murano, A. 2014. On module checking and strategies. In Bazzan, A. L. C.; Huhns, M. N.; Lomuscio, A.; and Scerri, P., eds., *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, 701–708. IFAAMAS/ACM.

Jamroga, W., and van der Hoek, W. 2004. Agents that know how to play. *Fundam. Informaticae* 63(2-3):185–219.

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

Kacprzak, M.; Niewiadomski, A.; and Penczek, W. 2020. Sat-based ATL satisfiability checking. In Calvanese, D.; Erdem, E.; and Thielscher, M., eds., *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, 539–549.

Klenze, T.; Bayless, S.; and Hu, A. 2016. Fast, flexible, and minimal CTL synthesis via SMT. In Chaudhuri, S., and Farzan, A., eds., *Computer Aided Verification*, 136–156. Springer International Publishing.

Kurpiewski, D.; Jamroga, W.; and Knapik, M. 2019. STV: model checking for strategies under imperfect information. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, 2372–2374.

Lomuscio, A., and Raimondi, F. 2006. Model checking knowledge, strategies, and games in multi-agent systems. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, 161–168.

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).

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2017. Reasoning about strategies: on the satisfiability problem. *Log. Methods Comput. Sci.* 13(1).

Mogavero, F.; Murano, A.; and Sauro, L. 2013. On the boundary of behavioral strategies. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, 263–272. IEEE Computer Society.

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010. Reasoning about strategies. In Lodaya, K., and Mahajan, M., eds., *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, 133–144. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Niewiadomski, A.; Kacprzak, M.; Kurpiewski, D.; Knapik, M.; Penczek, W.; and Jamroga, W. 2020. MsATL: A tool for SAT- based ATL satisfiability checking. In Seghrouchni, A. E. F.; Sukthankar, G.; An, B.; and Yorke-Smith, N., eds., *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, 2111–2113. International Foundation for Autonomous Agents and Multiagent Systems.

Pauly, M. 2002. A modal logic for coalitional power in games. *J. Log. Comput.* 12(1):149–166.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, 46–57. IEEE Computer Society.

Sebastiani, R. 2007. Lazy satisfiability modulo theories. *Journal on Satisfiability, Boolean Modeling and Computation* 3(3-4):141–224.

Silva, J. P. M., and Sakallah, K. A. 2003. Grasp—a new search algorithm for satisfiability. In *The Best of ICCAD*. Springer. 73–89.

Tabatabaei, M.; Jamroga, W.; and Ryan, P. Y. A. 2016. Expressing receipt-freeness and coercion-resistance in logics of strategic ability: Preliminary attempt. In *Proceedings of the 1st International Workshop on AI for Privacy and Security, PrAISe@ECAI 2016, The Hague, Netherlands, August 29-30, 2016*, 1:1–1:8. ACM.

Walther, D.; van der Hoek, W.; and Wooldridge, M. J. 2007. Alternating-time temporal logic with explicit strategies. In Samet, D., ed., *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007), Brussels, Belgium, June 25-27, 2007*, 269–278.