Scalable Verification of Strategy Logic through Three-valued Abstraction

F. Belardinelli[1] , **A. Ferrando**[2] , **W. Jamroga**[3] , **V. Malvone**[4] , **A. Murano**[5]

[1]Imperial College London, United Kingdom [2]University of Genoa, Italy
[3]SnT, University of Luxembourg & Institute of Computer Science, Polish Academy of Sciences
[4]Telecom Paris, France [5]University of Naples Federico II, Italy
francesco.belardinelli@imperial.ac.uk, angelo.ferrando@unige.it, w.jamroga@ipipan.waw.pl, vadim.malvone@telecom-paris.fr, aniello.murano@unina.it

Abstract

The model checking problem for multi-agent systems against Strategy Logic specifications is known to be non-elementary. On this logic several fragments have been defined to tackle this issue but at the expense of expressiveness. In this paper, we propose a three-valued semantics for Strategy Logic upon which we define an abstraction method. We show that the latter semantics is an approximation of the classic two-valued one for Strategy Logic. Furthermore, we extend MCMAS, an open-source model checker for multi-agent specifications, to incorporate our abstraction method and present some promising experimental results.

1 Introduction

In multi-agent systems, logics for strategic reasoning play a key role. In this domain, one of the success stories is Alternating-time Temporal Logic (ATL*)[Alur *et al.*, 2002], which can express cooperation and competition among teams of agents in order to achieve temporal goals, such as fairness, liveness, safety requirements. In fact, ATL* extends the well known branching-time temporal logic CTL* [Halpern and Shoham, 1986] by generalizing the existential E and universal A path quantifiers of CTL* with the strategic modalities $\langle\langle C\rangle\rangle$ and $[\![C]\!]$, where C is a coalition of agents. However, it has been observed that ATL* suffers from a number of limitations that, on the one hand, make the model-checking and satisfiability problems decidable (both are 2ExpTime-complete); but, on the other hand, make the logic too weak to express key game-theoretic concepts, such as Nash equilibria [Mogavero *et al.*, 2012]. To overcome these limitations, *Strategy Logic* (SL) [Mogavero *et al.*, 2014; Chatterjee *et al.*, 2010] has been put forward. A key aspect of SL is to consider strategies as first-order objects that can be existentially or universally quantified over by means of the strategy quantifiers $\exists x$ and $\forall x$, respectively. Then, by means of a binding operator (a, x), a strategy x can be associated to a specific agent a. This allows to reuse strategies as well as to share them among different agents. Since its introduction, SL has proved to be a powerful formalism: it can express complex solution concepts, including Nash equilibria, and subsumes all previously introduced logics for strategic reasoning, including ATL*. The high expressivity of SL has spurred its analysis in a number of directions and extensions, such as prompt [Aminof *et al.*, 2016], graded [Aminof *et al.*, 2018], fuzzy [Bouyer *et al.*, 2019], probabilistic [Aminof *et al.*, 2019], and imperfect [Berthon *et al.*, 2021; Belardinelli *et al.*, 2020] strategic reasoning.

As one may expect, the high expressivity of SL comes at a price. Indeed, its model-checking problem turns out to be non-elementary [Mogavero *et al.*, 2014]. Moreover, the model checking procedure is not immune to the well-known state-space explosion, as faithful models of real-world systems are intrinsically complex and often infeasible even to generate, let alone verify. These issues call for techniques to make model checking SL amenable at least in practice. A technique that has been increasingly used in industrial settings to verify hardware and software systems is state abstraction, which allows to reduce the state space to manageable size by clustering "similar" concrete states into abstract states. Abstraction has been first introduced for stand-alone systems [Clarke *et al.*, 1994], then extended to two-agent system verification [Grumberg *et al.*, 2007; Shoham and Grumberg, 2004; Bruns and Godefroid, 2000; Aminof *et al.*, 2012]. Recently, abstraction approaches have been investigated for multi-agent systems w.r.t. ATL* specifications [Kouvaros and Lomuscio, 2017; Belardinelli *et al.*, 2019; Belardinelli and Lomuscio, 2017; Jamroga *et al.*, 2016; Jamroga *et al.*, 2020; Belardinelli *et al.*, 2023; Ferrando and Malvone, 2023; Belardinelli *et al.*, 2022; Belardinelli *et al.*, 2018; Belardinelli and Malvone, 2020; Ferrando and Malvone, 2021; Ferrando and Malvone, 2022]. A natural direction is then to investigate a form of abstraction suitable for Strategy Logic as well.

Our Contribution. In this paper we introduce the first notion of three-valued abstraction for SL. The contribution of this paper is threefold. First, in Sec. 3 we define a three-valued semantics for SL where, besides the standard truth values true $\top$ and false $\perp$, we have a third value undefined $\mathbf{u}$ that models situations where the verification procedure is not able to return a conclusive answer. Second, in Sec. 4 we introduce an abstraction procedure for SL, which can reduce significantly the size of the state space of SL models, although at the cost of making some formulas undefined. The main theoretical result is the Preservation Theorem 4.2, which allows us to model check SL formulas in the three-valued abstraction and then lift any defined answer to the original two-valued model. Third, in Sec. 6 we evaluate empirically the trade-off between state-space reduction and definiteness, by applying our abstraction procedure to a scheduling scenario. What we observe empirically is a significant reduction of the model size, which allows us to verify instances that are not amenable to current model checking tools.

Related Work. The present contribution is inspired by a long tradition of works on the abstraction of MAS models, including through three-valued semantics. An abstraction-refinement framework for the temporal logic CTL over a three-valued semantics was first studied in [Shoham and Grumberg, 2004; Shoham and Grumberg, 2007], and then extended to the full μ-calculus [Grumberg *et al.*, 2007] and hierarchical systems [Aminof *et al.*, 2012]. Three-valued abstractions for the verification of Alternating-time Temporal Logic have been put forward in [Ball and Kupferman, 2006; Lomuscio and Michaliszyn, 2014; Lomuscio and Michaliszyn, 2015; Lomuscio and Michaliszyn, 2016]. In [Ball and Kupferman, 2006; Shoham and Grumberg, 2004] ATL* is interpreted under perfect information; while [Lomuscio and Michaliszyn, 2014; Lomuscio and Michaliszyn, 2015; Lomuscio and Michaliszyn, 2016] consider *non-uniform* strategies [Raimondi and Lomuscio, 2005]. Finally, [Jamroga *et al.*, 2016; Jamroga *et al.*, 2020] introduce a multi-valued semantics for ATL* that is a conservative extension of the classical two-valued variant. Related to this line, three-valued logics have been extensively applied to system verification, including [Bruns and Godefroid, 1999; Huth *et al.*, 2001; Godefroid and Jagadeesan, 2003]

Clearly, we build in this long line of works, but the expressiveness of SL raises specific challenges that the authors of the contributions above need not to tackle. We briefly mention them here and refer to specific sections for further details. First, we have to introduce individual $must$ and may actions and strategies as under- and over- approximations of the behaviours of our agents. Second, the loosely-coupled nature of agents requires to consider non-deterministic transitions in the abstraction (Sec. 4). Third, the arbitrary alternation of existential and universal strategy quantifiers makes proving the Preservation Theorem 4.2 significantly more challenging, and complicates our experiments in verifying three-valued SL in the two-valued model-checking tool MCMAS (Sec. 6).

2 Reasoning about Strategies

In this section we recall the definitions of basic notions for Strategy Logic [Mogavero *et al.*, 2014].

2.1 Syntax

Strategy Logic (*SL*) syntactically extends LTL with two *strategy quantifiers*, the existential $\exists x$ and universal $\forall x$, and an *agent binding* (a, x), where a is an agent and x a variable. Intuitively, these additional elements can be respectively read as *"there exists a strategy x"*, *"for all strategies x"*, and *"bind agent a to the strategy associated with the variable x"*. Since negated quantifiers often prove problematic in many valued settings, we restrict the syntax of SL to formulas in Negation Normal Form (NNF), without loss of expressiveness. In that case, the universal strategic quantifier $\forall x$ and the temporal operator "Release" R are added as primitives, and negation is allowed only at the level of literals. Note that every formula of SL can be equivalently transformed to one in NNF, with at most a linear blowup [Mogavero *et al.*, 2014].

Definition 2.1 (SL Syntax)**.** *Given the set AP of atoms, variables Var, and agents Ag, the formal syntax of SL is defined as follows, where $p \in AP$, $x \in Var$, and $a \in Ag$:*

$$\begin{aligned}\varphi \quad ::= \quad & p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x\varphi \mid \forall x\varphi \mid (a, x)\varphi \mid \\ & X\varphi \mid \varphi\, U \varphi \mid \varphi\, R\, \varphi\end{aligned}$$

We introduce the derived temporal operators as usual: $F\varphi = \top U\varphi$ ("eventually") and $G\varphi = \bot R\varphi$ ("always").

Usually, predicate logics need the concepts of free and bound *placeholders* in order to formally define their semantics. In SL, since strategies can be associated to both agents and variables, we introduce the set of *free agents/variables* $free(\varphi)$ as the subset of $Ag \cup Var$ containing *(i)* all agents a for which there is no binding (a, x) before the occurrence of a temporal operator, and *(ii)* all variables x for which there is a binding (a, x) but no quantification $\exists x$ or $\forall x$. A formula φ without free agents (resp., variables), i.e., with $free(\varphi) \cap Ag = \emptyset$ (resp., $free(\varphi) \cap Var = \emptyset$), is called *agent-closed* (resp., *variable-closed*). If φ is both agent- and variable-closed, it is a *sentence*.

2.2 Two-valued Semantics

We now provide a formal semantics to Strategy Logic.

Models. To model the behaviour of multi-agent systems, we use a variant of concurrent game structures [Alur *et al.*, 2002].

Definition 2.2 (CGS)**.** *A* concurrent game structure (CGS) *is a tuple $G = \langle Ag, St, s_0, Act, \tau, AP, V\rangle$ such that (i) Ag is a finite, non-empty set of* agents*. (ii) St is a finite, non-empty set of* states*, with* initial state $s_0 \in St$*. (iii) Act is a finite, nonempty set of* actions*. We use $ACT = Act^{|Ag|}$ for the set of all* joint actions *(a.k.a.* action profiles*), i.e., tuples of individual actions, played synchronously by all agents. (iv) $\tau : St \times ACT \to 2^{St}$ is the* transition function *assigning successor states $\{s', s'', \dots\} = \tau(s, \vec{\alpha})$ to each state $s \in St$ and joint action $\vec{\alpha} \in ACT$. We assume that the transitions in a CGS are deterministic, i.e., $\tau(s, \vec{\alpha})$ is always a singleton.*[1] *(v) AP is a set of* atomic propositions*, and (vi) $V : St \times AP \to \{\top, \bot\}$ is a* two-valued labelling function*.*

By Def. 2.2 a CGS describes the interactions of a group Ag of agents, starting from the initial state $s_0 \in St$, according to the transition function τ. We use G as a subscript for Ag_G, St_G, etc., whenever the model is not clear from the context.

Note that the CGSs used in the semantics of Strategy Logic assume that all the actions are available to every agent at every state [Mogavero *et al.*, 2014]. This is because a strategy assigned to variable x can be later associated with any agent a by means of the binding operator (a, x). As a consequence,

[1] The deterministic transitions in a CGS are usually defined by a function of type $\tau : St \times ACT \to St$. We use a slightly different (but equivalent) formulation. This will make it easier for us to extend it to nondeterministic transitions in three-valued models (see Def. 3.1).

the available strategies (and hence also the available actions) are the same for every agent.

Tracks, Paths, Strategies. We denote the i-th element of a tuple v as v_i, the prefix of v of lenght i as $v_{\leq i}$, and with $last(v)$ as the last element of v. A *track* is a *finite nonempty* sequence of states $\rho \in St^+$ such that, for all $0 \leq i \leq |\rho| - 1$, there is an action profile $\vec{\alpha} \in ACT$ with $(\rho)_{i+1} \in \tau((\rho)_i, \vec{\alpha})$. Similarly, a *path* is an *infinite* sequence of states $\pi \in St^\omega$ such that, for all $i \in N$, there is $\vec{\alpha} \in ACT$ with $(\pi)_{i+1} \in \tau((\pi)_i, \vec{\alpha})$. The set $Trk \subseteq St^+$ contains all the tracks in the model, and $Trk(s)$ the tracks starting at state $s \in St$. The sets Pth and $Pth(s)$ are defined analogously. We denote the prefix of a path π up to position $i \in \mathbb{N}$ as $\pi_{\leq i}$.

A *strategy* is a partial function $f : Trk \to Act$ that maps each track in its domain to an action. Intuitively, a strategy is a conditional plan that, for some tracks of G, prescribes an action to be executed. A strategy is *memoryless* (or positional), if $last(\rho) = last(\rho')$ implies $f(\rho) = f(\rho')$, that is, the strategy only depends on the last state. The set $Str = Trk \to Act$ (resp., $Str(s) = Trk(s) \to Act$) contains all strategies (resp., strategies starting from s).

Assignments. Let Var be the set of variables. An *assignment* is a partial function $\chi : Var \cup Ag \to Str$ mapping variables and agents in its domain to strategies. An assignment χ is *complete* if it is defined on all agents, i.e., $Ag \subseteq dom(\chi)$. The set $Asg = Var \cup Ag \to Str$ contains all assignments. Moreover, $Asg(X) = X \to Str$ indicates the subset of X*-defined* assignments, i.e., assignments defined on $X \subseteq Var \cup Ag$.

As in first-order logic, in order to quantify over strategies or bind a strategy to an agent, we update an assignment χ by associating an agent or a variable l with a new strategy f. Let $\chi \in Asg$ be an assignment, $f \in Str$ a strategy and $l \in Var \cup Ag$ either an agent or a variable. Then, $\chi[l \mapsto f] \in Asg$ denotes the new assignment that returns f on l and the same value that χ would return on the rest of its domain.

Outcome Plays of a Strategy. A *play* is the unique outcome of the game settled by all agent strategies engaged in it. Formally, given a state $s \in St$ and a complete assignment $\chi \in Asg(s)$, the function $play(\chi, s)$ returns the path $\pi \in Pth(s)$ such that, for all $i \in N$, it holds that $\{\pi_{i+1}\} = \tau(\pi_i, \vec{\alpha})$, where $\vec{\alpha}(a) = \chi(a)(\pi_{\leq i})$ for each $a \in Ag$.

We now define the translation of an assignment together with a related path (resp. state). It is used to keep track, at a certain stage of the play, of the current state and its updated assignment. For a path π and an assignment $\chi \in Asg$, the *i-th global translation* of (χ, π) with $i \in N$ is the pair $(\chi, \pi)^i = (\chi_{\pi_{\leq i}}, \pi_i)$ of an assignment and a state. Moreover, for a state $s \in St$, we define $(\chi, s)^i = (\chi, play(\chi, s))^i$.

As in the case of components of a model, in order to avoid any ambiguity, we sometimes use the name of the model as a subscript of the sets and functions introduced above.

Satisfaction. The (two-valued) satisfaction relation for SL is defined as follows.

Definition 2.3 (Two-valued Satisfaction)**.** *Given a model G, for all SL formulas φ, states $s \in St$, and assignments $\chi \in Asg$ with $free(\varphi) \subseteq dom(\chi)$, the satisfaction relation $(G, \chi, s) \models^2 \varphi$ is inductively defined as follows:*

- $(G, \chi, s) \models^2 p$ *iff* $V(s, p) = \top$, *for* $p \in AP$.
- *Boolean operators are interpreted as usual.*
- $(G, \chi, s) \models^2 \exists x\varphi$ *iff for some strategy* $f \in Str(s)$, $(G, \chi[x \mapsto f], s) \models^2 \varphi$.
- $(G, \chi, s) \models^2 \forall x\varphi$ *iff for all strategies* $f \in Str(s)$, $(G, \chi[x \mapsto f], s) \models^2 \varphi$.
- $(G, \chi, s) \models^2 (a, x)\varphi$ *iff* $(G, \chi[a \mapsto \chi(x)], s) \models^2 \varphi$.
- *Finally, if the assignment χ is also complete, it holds that:*
  - $(G, \chi, s) \models^2 X\varphi$ *iff* $(G, (\chi, s)^1) \models^2 \varphi$*;*
  - $(G, \chi, s) \models^2 \varphi_1 U \varphi_2$ *iff for some index* $i \in N$, $(G, (\chi, s)^i) \models^2 \varphi_2$ *and, for all* $j < i$, *it holds that* $(G, (\chi, s)^j) \models^2 \varphi_1$*;*
  - $(G, \chi, s) \models^2 \varphi_1 R \varphi_2$ *iff, for all* $i \in N$, $(G, (\chi, s)^i) \models^2 \varphi_2$ *or there is* $j \leq i$ *such that* $(G, (\chi, s)^i) \models^2 \varphi_1$.

Due to the semantics of the Next X, Until U, and Release R operators, LTL semantics is clearly embedded into the SL one. Furthermore, since the satisfaction of a sentence φ does not depend on assignments, we omit them and write $(G, s) \models \varphi$, when s is a generic state in St, and $G \models \varphi$ when $s = s_0$.

Note that we can easily define the memoryless variant of strategy logic by restricting the clauses for operators $\exists x$ and (a, x) to memoryless strategies.

Finally, we define the (two-valued) model checking problem for SL as determining whether an SL formula ϕ holds in a CGS G, that is, whether $G \models^2 \phi$. We conclude this section by stating the related complexity result.

Theorem 2.4 ([Mogavero *et al.*, 2014])**.** *The model checking problem for Strategy Logic is non-elementary.*

3 Three-Valued Strategy Logic

In this section we introduce a novel three-valued semantics for Strategy Logic starting by extending CGSs.

3.1 Three-Valued CGSs

We extend (two-valued) CGSs with $must$ and may transitions as under- and over-approximations of the strategic abilities of agents.

Definition 3.1 (Three-valued CGS)**.** *A* three-valued CGS *is a tuple* $G = \langle Ag, St, s_0, Act^{may}, Act^{must}, \tau^{may}, \tau^{must}, AP, V \rangle$, *where:*

- Ag, St, s_0, AP *are defined as in Def. 2.2.*
- Act^{may} *and* Act^{must} *provide respectively the upper and lower approximation of the available actions. We assume that* $Act^{must} \subseteq Act^{may}$*. The sets of* may *and* must *action profiles are given by* $ACT^{may} = (Act^{may})^{|Ag|}$ *and* $ACT^{must} = (Act^{must})^{|Ag|}$*, respectively.*
- $\tau^{may} : St \times ACT^{may} \to 2^{St}$ *is the* may *transition function, and* $\tau^{must} : St \times ACT^{may} \to 2^{St}$ *the* must *transition function. Note that both functions are possibly nondeterministic and are* defined *on all the potential action profiles in the system, i.e.,* ACT^{may}*. However, we only require that they* return nonempty successor sets *on their respective action profiles. That is,* $\tau^{may}(s, \vec{\alpha}) \neq \emptyset$ *for every state* $s \in St$ *and action profile* $\vec{\alpha} \in ACT^{may}$*,*

and $\tau^{must}(s,\vec{\alpha}) \neq \emptyset$ for every state $s \in St$ and action profile $\vec{\alpha} \in ACT^{must}$.[2] *Moreover, it is required that $\tau^{must}(s,\vec{\alpha}) \subseteq \tau^{may}(s,\vec{\alpha})$ for every $s \in St$ and $\vec{\alpha} \in ACT^{may}$. In other words, every must transition is also a may transition, but not necessarily viceversa.*

- *The labelling function $V : St \times AP \to \{\bot, \top, \mathbf{u}\}$ maps now each pair of a state and an atom to a truth value of "true," "false," or "undefined."*

The notions of tracks, paths, and the definitions of sets Trk, $Trk(s)$, Pth, $Pth(s)$ carry over from Section 2.2.

May/Must Strategies and their Outcomes. A *may-strategy* (resp. *must-strategy*) is a function $f : Trk \to Act^{may}$ (resp. Act^{must}) that maps each track to a *may* (resp. *must*) action. Note that each *must*-strategy is a *may*-strategy, but not necessarily the other way around. Moreover, we can define memoryless *may*- and *must*-strategies in the standard way. The sets Str^{may} and Str^{must} are defined analogously to Section 2.2.

Given a state $s \in St$ and a profile of (*may* and/or *must*) strategies, represented by a complete assignment $\chi \in Asg$, we define two kinds of outcome sets, $plays^{may}(\chi, s)$ and $plays^{must}(\chi, s)$. The former over-approximates the set of paths that can really occur when executing χ from s, while the latter under-approximates it. Typically, we will use $plays^{may}$ to establish that the value of a temporal formula φ is $\top$ (if φ holds in all such paths), and $plays^{must}$ for $\bot$ (if φ is false in at least one path). Formally, the function $plays^{may}(\chi, s)$ returns the paths $\pi \in Pth(s)$ such that, for all $i \in N$, it holds that $\pi_{i+1} \in \tau^{may}(\pi_i, \vec{\alpha})$, where $\vec{\alpha}(a) = \chi(a)(\pi_{\leq i})$ for each $a \in Ag$. The definition of $plays^{must}(\chi, s)$ is analogous, only with τ^{must} being used instead of τ^{may}.

3.2 Three-valued Semantics

We now define the Three-valued satisfaction relation for Strategy Logic.

Definition 3.2 (Three-valued Satisfaction)**.** *Given a 3-valued model G, for all SL formulas φ, states $s \in St$, and assignments $\chi \in Asg(s)$ with $free(\varphi) \subseteq dom(\chi)$, the satisfaction relation $(G, \chi, s \models^3 \varphi) = tv$ is inductively defined as follows.*

- $(G, \chi, s \models^3 p) = V(s, p)$, *for $p \in AP$.*
- *Boolean operators are interpreted as in Łukasiewicz's three valued logic [Łukasiewicz, 1920].*
- *For $\phi = \exists x\varphi$,*
  - $(G, \chi, s \models^3 \phi) = \top$ *iff* $(G, \chi[x \mapsto f], s \models^3 \varphi) = \top$ *for some must-strategy $f \in Str^{must}(s)$;*
  - $(G, \chi, s \models^3 \phi) = \bot$ *iff* $(G, \chi[x \mapsto f], s \models^3 \varphi) = \bot$ *for all may-strategies $f \in Str^{may}(s)$;*
  - *otherwise,* $(G, \chi, s \models^3 \phi) = \mathbf{u}$.
- *For $\phi = \forall x\varphi$,*
  - $(G, \chi, s \models^3 \phi) = \top$ *iff for all may-strategies $f \in Str^{may}(s)$, $(G, \chi[x \mapsto f], s \models^3 \varphi) = \top$;*
  - $(G, \chi, s \models^3 \phi) = \bot$ *iff for some must-strategy $f \in Str^{must}(s)$, $(G, \chi[x \mapsto f], s \models^3 \varphi) = \bot$;*
  - *otherwise,* $(G, \chi, s \models^3 \phi) = \mathbf{u}$.
- $(G, \chi, s \models^3 (a, x)\varphi) = (G, \chi[a \mapsto \chi(x)], s \models^3 \varphi)$.
- *Finally, if the assignment χ is also complete, we define:*
  - $(G, \chi, s \models^3 X\varphi) = \top$ *iff for all $\pi \in plays^{may}(\chi, s)$, we have $(G, (\chi, \pi)^1 \models^3 \varphi) = \top$;*
  - $(G, \chi, s \models^3 X\varphi) = \bot$ *iff for some $\pi \in plays^{must}(\chi, s)$, we have $(G, (\chi, \pi)^1 \models^3 \varphi) = \bot$;*
  - *otherwise,* $(G, \chi, s \models^3 X\varphi) = \mathbf{u}$.
  - $(G, \chi, s \models^3 \varphi_1 U \varphi_2) = \top$ *iff for all $\pi \in plays^{may}(\chi, s)$, there is $i \in N$ such that $(G, (\chi, \pi)^i \models^3 \varphi_2) = \top$, and for all $j < i$ we have $(G, (\chi, \pi)^j \models^3 \varphi_1) = \top$;*
  - $(G, \chi, s \models^3 \varphi_1 U \varphi_2) = \bot$ *iff for some $\pi \in plays^{must}(\chi, s)$ and all $i \in N$, we have $(G, (\chi, \pi)^i \models^3 \varphi_2) = \bot$ or there exists $j < i$ such that $(G, (\chi, \pi)^j \models^3 \varphi_1) = \bot$;*
  - *otherwise,* $(G, \chi, s \models^3 \varphi_1 U \varphi_2) = \mathbf{u}$.
  - $(G, \chi, s \models^3 \varphi_1 R \varphi_2) = \top$ *iff for all $\pi \in plays^{may}(\chi, s)$ and $i \in N$, we have $(G, (\chi, \pi)^i \models^3 \varphi_2) = \top$ or there exists $j \leq i$ such that $(G, (\chi, \pi)^j \models^3 \varphi_1) = \top$;*
  - $(G, \chi, s \models^3 \varphi_1 R \varphi_2) = \bot$ *iff for some $\pi \in plays^{must}(\chi, s)$ and $i \in N$, we have $(G, (\chi, \pi)^i \models^3 \varphi_2) = \bot$ and for all $j \leq i$, we have $(G, (\chi, \pi)^j \models^3 \varphi_1) = \bot$;*
  - *otherwise,* $(G, \chi, s \models^3 \varphi_1 R \varphi_2) = \mathbf{u}$.

Again, we can define the memoryless, three-valued satisfaction relation for SL by restricting the clauses for operators $\exists x$, $\forall x$, and (a, x) to memoryless strategies. Similarly to Section 2, if ϕ is a sentence, then $(G, s \models^3 \varphi) = (G, \chi, s \models^3 \varphi)$ for any assignment χ, and $(G \models^3 \varphi) = (G, s_0 \models^3 \varphi)$.

We now show that our three-valued semantics in Def. 2.3 is a conservative extension of the standard two-valued interpretation in Sec. 2.

Theorem 3.3 (Conservativeness)**.** *Let G be a standard CGS, that is, $Act^{may} = Act^{must}$, $\tau^{may} = \tau^{must}$ are functions, and the truth value of every atom is defined (i.e., it is equal to either $\top$ or $\bot$). Then, for every formula ϕ in SL,*

$$(G, \chi, s \models^3 \phi) = \top \quad \Leftrightarrow \quad (G, \chi, s) \models^2 \phi \tag{1}$$

$$(G, \chi, s \models^3 \phi) = \bot \quad \Leftrightarrow \quad (G, \chi, s) \not\models^2 \phi \tag{2}$$

Proof. The result follows from the fact that in standard CGS the clauses for the three-valued satisfaction relation collapse to those for two-valued satisfaction, whenever $Act^{may} = Act^{must}$, $\tau^{may} = \tau^{must}$ are functions, and the truth value of every atom is defined. □

Remark 3.4 (Model checking)**.** *For any syntactic fragment $\mathcal{L}$ of SL, model checking of $\mathcal{L}$ with 3-valued semantics can be reduced to 2-valued model checking of $\mathcal{L}$ by a construction similar to [Jamroga et al., 2016, Theorem 4]. Note also that 2-valued model checking for $\mathcal{L}$ is a special case of its 3-valued counterpart, due to Theorem 3.3. Thus, the decidability and complexity for 2-valued model checking in fragments of SL carry over to 3-valued verification.*

[2] Note that the function τ^{must} is total because we assume the empty set as an element of the co-domain.

4 Three-valued Abstraction for SL

Here, we define the 3-valued state abstraction for CGS. The idea is to cluster the states of a CGS (called the *concrete model*) according to a given equivalence relation $\approx$, e.g., one provided by a domain expert. Typically, two states are deemed equivalent if they agree on the evaluation of atoms, possibly just the atoms appearing on a given formula ϕ to be checked. In some cases, such an equivalence relation might be too coarse and therefore more domain-dependent information could be taken into account.

Then, the sets of may (resp. must) actions and the may (resp. must) transitions are computed in such a way that they always overapproximate (resp. underapproximate) the actions and transitions in the concrete model. Formally, the abstraction is defined as follows.

Definition 4.1 (Abstraction)**.** *Let $G = \langle Ag, St, s_0, Act, \tau, AP, V\rangle$ be a CGS, and $\approx \subset St \times St$ an equivalence relation. We write $[s]$ for the equivalence class of $\approx$ that contains s. The* abstract model *of G w.r.t. $\approx$ is defined as the 3-valued CGS $\mathcal{A}(G) = \langle \mathcal{A}(Ag), \mathcal{A}(St), \mathcal{A}(s_0), \mathcal{A}^{may}(Act), \mathcal{A}^{must}(Act), \mathcal{A}^{may}(\tau), \mathcal{A}^{must}(\tau), \mathcal{A}(AP), \mathcal{A}(V)\rangle$, with:*

- *$\mathcal{A}(Ag) = Ag$ and $\mathcal{A}(AP) = AP$.*
- *$\mathcal{A}(St) = \{[s] \mid s \in St\}$ with $\mathcal{A}(s_0) = [s_0]$.*
- *$\mathcal{A}^{may}(Act) = Act$.*
- *$\mathcal{A}^{may}(\tau) = \tau^{may} : \mathcal{A}(St) \times (\mathcal{A}^{may}(Act))^{|\mathcal{A}(Ag)|} \to 2^{\mathcal{A}(St)}$ such that $\tau^{may}([s], \vec{\alpha}) = \{[s_{succ}] \mid \exists s' \in [s]\ \exists s'_{succ} \in [s_{succ}]\ .\ s'_{succ} \in \tau(s', \vec{\alpha})\}$).*
- *$\mathcal{A}^{must}(\tau) = \tau^{must} : \mathcal{A}(St) \times (\mathcal{A}^{may}(Act))^{|\mathcal{A}(Ag)|} \to 2^{\mathcal{A}(St)}$ such that $\tau^{must}([s], \vec{\alpha}) = \{[s_{succ}] \mid \forall s' \in [s]\ \exists s'_{succ} \in [s_{succ}]\ .\ s'_{succ} \in \tau(s', \vec{\alpha})\}$).*
- *$\mathcal{A}^{must}(Act)$ is a maximal[3] set $Act^{must} \subseteq Act$ such that $\forall s \in St\ \forall \vec{\alpha} \in (Act^{must})^{|\mathcal{A}(Ag)|}\ .\ \tau^{must}([s], \vec{\alpha}) \neq \emptyset$. Note that a unique maximal set does not always exist. In such cases, a natural heuristics would be to choose the maximal subset of actions with the largest cardinality, breaking ties lexicographically in case there are still multiple solutions.*
- $\mathcal{A}(V)([s], p) = \begin{cases} \top & \textit{if } V(s', p) = \top \textit{ for all } s' \in [s] \\ \bot & \textit{if } V(s', p) = \bot \textit{ for all } s' \in [s] \\ \mathbf{u} & \textit{otherwise.} \end{cases}$

Note that $\mathcal{A}(G)$ can be computed in polynomial time w.r.t. the size of G, assuming the above heuristics for $\mathcal{A}^{must}(Act)$.

We now prove that the abstraction preserves classical truth values. Given a strategy f in G, we define the set of corresponding may-strategies in $\mathcal{A}(G)$ by $abstr^{may}(f) = \{f^\dagger \mid f^\dagger([s_0], \ldots, [s_n]) = f(s'_0, \ldots, s'_n)$ for some $s'_0 \in [s_0], \ldots, s'_n \in [s_n]\}$. Moreover, $abstr^{must}(f) = abstr^{may}(f) \cap Str^{must}$. Note that $abstr^{may}(f)$ is always nonempty. Also, $abstr^{must}(f)$ is either empty or a singleton.

Conversely, given a (may or $must$) strategy f in $\mathcal{A}(G)$, we define the set of corresponding concrete strategies in G by $concr(f) = \{f^* \mid f^*(s_0, \ldots, s_n) = f([s_0], \ldots, [s_n])\}$. Notice that $concr(f)$ is always a singleton for $must$ strategies, and either empty or a singleton for may strategies. We lift $abstr^{may}, abstr^{must}, concr$ to sets of strategies in the standard way. Clearly, $f \in concr(abstr^{may}(f))$ for any concrete strategy, and $f \in abstr^{must}(concr(f))$ for any $must$-strategy. We lift the notation to assignments analogously. Observe that, in every $\chi^* \in concr(\chi[x \mapsto f])$, x is assigned with $f^* \in concr(f)$.

Theorem 4.2 (Preservation)**.** *Let G be a CGS and $\mathcal{A}(G)$ its abstraction induced by equivalence relation $\approx$. Then, for every formula ϕ in SL, every (may or must) assignment χ and state s in $\mathcal{A}(G)$, every assignment $\chi^* \in concr(\chi)$, and state $t \in s$ in G, it holds that:*

$$((\mathcal{A}(G), \chi, s) \models^3 \phi) = \top \quad \Rightarrow \quad (G, \chi^*, t) \models^2 \phi \quad (3)$$

$$((\mathcal{A}(G), \chi, s) \models^3 \phi) = \bot \quad \Rightarrow \quad (G, \chi^*, t) \not\models^2 \phi \quad (4)$$

Proof. The proof is by induction on the structure of ϕ.

Induction base ($\phi = p$): $((\mathcal{A}(G), \chi, s) \models^3 \phi) = \top$ iff $\mathcal{A}(V)(s, p) = \top$, iff for all $t \in s$, $V(t, p) = \top$, that is, $(G, \chi^*, t) \models^2 \phi$. The case for $\bot$ is proved similarly. The case of $\phi = \neg p$ is analogous.

Case $\phi = \psi_1 \wedge \psi_2$: $((\mathcal{A}(G), \chi, s) \models^3 \phi) = \top$ iff $((\mathcal{A}(G), \chi, s) \models^3 \psi_1) = \top$ and $((\mathcal{A}(G), \chi, s) \models^3 \psi_2) = \top$. By induction, for all $\chi^* \in concr(\chi)$ and $t \in s$, $(G, \chi^*, t) \models^2 \psi_1$ and $(G, \chi^*, t) \models^2 \psi_2$. Thus, $(G, \chi^*, t) \models^2 \psi_1 \wedge \psi_2$.

Further, $((\mathcal{A}(G), \chi, s) \models^3 \phi) = \bot$ iff $((\mathcal{A}(G), \chi, s) \models^3 \psi_1) = \bot$ or $((\mathcal{A}(G), \chi, s) \models^3 \psi_2) = \bot$. By induction, for all $\chi^* \in concr(\chi)$ and $t \in s$, $(G, \chi^*, t) \not\models^2 \psi_1$ or for all $\chi^* \in concr(\chi)$ and $t \in s$, $(G, \chi^*, t) \not\models^2 \psi_2$. Thus, for all $\chi^* \in concr(\chi)$ and $t \in s$, $(G, \chi^*, t) \not\models^2 \psi_1 \wedge \psi_2$. The case of $\phi = \psi_1 \vee \psi_2$ is analogous.

Case $\phi = \exists x \psi$: $(\mathcal{A}(G), \chi, s \models^3 \phi) = \top$ iff for some $must$-strategy $f \in Str^{must}(s)$, $(\mathcal{A}(G), \chi[x \mapsto f], s \models^3 \psi) = \top$. By induction, for all $\chi^* \in concr(\chi[x \mapsto f])$ and $t \in s$, it holds that $(G, \chi^*, t) \models^2 \psi$. Assume that $concr(\chi[x \mapsto f])$ is nonempty, and consider the sole concrete strategy $f^* \in concr(f)$. Clearly, $\chi^* = \chi^*[x \mapsto f^*]$ for every $\chi^* \in concr(\chi[x \mapsto f])$. Thus, $(G, \chi^*[x \mapsto f^*], t) \models^2 \psi$, and hence also $(G, \chi^*, t) \models^2 \exists x \psi$. Assume now, to the contrary, that $concr(\chi[x \mapsto f])$ is empty. In that case, $(G, \chi^*[x \mapsto f^*], t) \models^2 \psi$ holds vacuously for all $\chi^* = \chi^*[x \mapsto f^*]$, and hence again $(G, \chi^*, t) \models^2 \exists x \psi$.

Further, $(\mathcal{A}(G), \chi, s \models^3 \phi) = \bot$ iff for every may-strategy $f \in Str^{may}(s)$, $(\mathcal{A}(G), \chi[x \mapsto f], s \models^3 \psi) = \bot$. Take any concrete strategy g in G, and consider any $g^\dagger \in abstr^{may}(f)$. By the above, $(\mathcal{A}(G), \chi[x \mapsto g^\dagger], s \models^3 \psi) = \bot$. Thus, by induction, $(G, \chi^*, t) \not\models^2 \psi$ for all $\chi^* \in concr(\chi[x \mapsto g^\dagger])$ and $t \in s$. Similarly to the previous paragraph, either (i) $concr(\chi[x \mapsto g^\dagger])$ is nonempty and $\chi^*[x \mapsto g] \in \chi^*$, thus $(G, \chi^*[x \mapsto g], t) \not\models^2 \psi$ for all such χ^*, or (ii) the same statement holds vacuously. In both cases, $(G, \chi^*, t) \not\models^2 \exists x \psi$.

The cases $\phi = \forall x \psi$ and $\phi = (a, x)\psi$ are analogous.

Case $\phi = X\psi$: $(\mathcal{A}(G), \chi, s \models^3 \phi) = \top$ iff for all $\pi \in plays^{may}(\chi, s)$, we have $(G, (\chi, \pi)^1) \models^3 \psi) = \top$. By induction, $(G, \chi^*, t) \models^2 \psi$ for every $\pi \in plays^{may}(\chi, s)$,

[3] with respect to set inclusion.

$\chi^* \in concr(\chi_{\pi_{\leq 1}})$ and $t \in (\pi)_1$. Take any state $t' \in s$ and assignment χ' in G such that $\chi'_{\pi^*_{\leq 1}} = \chi^*$ for some $\pi^* \in plays(\chi', t')$. Since may paths in $\mathcal{A}(G)$ overapproximate paths in G, we get that $(G, \chi', t') \models^2 X\psi$.

Further, $(\mathcal{A}(G), \chi, s \models^3 \phi) = \bot$ iff for some $\pi \in plays^{must}(\chi, s)$, we have $(G, (\chi, \pi)^1 \models^3 \varphi) = \bot$. By induction, there is $\pi \in plays^{must}(\chi, s)$ such that $(G, \chi^*, t) \not\models^2 \psi$ for every $\chi^* \in concr(\chi_{\pi_{\leq 1}})$ and $t \in (\pi)_1$. Take any state $t' \in s$ and assignment χ' in G. Since $must$ paths in $\mathcal{A}(G)$ underapproximate paths in G, there must be a path $\pi^* \in plays(\chi', t')$ such that $\chi'_{\pi^*_{\leq 1}} = \chi^*$. Thus, $(G, \chi', t') \not\models^2 X\psi$.

The cases $\underline{\phi = \psi_1 U \psi_2}$ and $\underline{\phi = \psi_1 R \psi_2}$ are analogous. □

Corollary 4.3. *For any CGS G and SL formula ϕ:*

$$(\mathcal{A}(G) \models^3 \phi) = \top \quad \Rightarrow \quad G \models^2 \phi$$
$$(\mathcal{A}(G) \models^3 \phi) = \bot \quad \Rightarrow \quad G \not\models^2 \phi$$

It is easy to see that the above results hold also for the semantic variant of SL based on memoryless strategies.

5 Implementation

We implemented a prototype tool in Java[4], which accepts CGSs and SL properties as input, on top of MCMAS, the *de facto* standard model checker for MAS [Lomuscio *et al.*, 2015]. Specifically, our tool exploits MCMAS as a black-box, for performing the actual verification step. In fact, our tool focuses on the abstraction procedure for the verification of SL formulas (as presented in this paper), while their verification is obtained through MCMAS.

From a practical perspective, there are various aspects to report, that can be summarised as (i) input/output of the tool; (ii) abstraction of the CGS; (iii) verification in MCMAS.

(i) The implementation allows for the definition of CGSs as external JSON[5] formatted input files. In this way, any end user may easily interact with the tool, independently from the CGS's internal representation (*i.e.*, the corresponding data structures). As CGSs, also the definition of the SL formula to check is handled as an external parameter to the tool. Once the verification ends, the outcome is returned to the user.

(ii) As presented in the paper, in order to improve the verification performance, the CGS is first abstracted. The abstraction is obtained by clustering multiple states into a single abstract state of the CGS. This step is based on an equivalence relation ($\approx$), as presented in Definition 4.1. An abstract state may be labeled by atoms. As presented in Definition 4.1, an atom holds (resp. does not hold) in the abstract state iff it holds (resp. does not hold) in all the concrete states which have been collapsed into the abstract state. Otherwise, the atom is considered undefined. Note that, since atoms can hold, not hold, or being undefined in a state, they are explicitly labeled in each state. In practice, this is obtained by duplicating each atom p into atoms $p_\top$ and $p_\bot$, which correspond to p holding or not holding in a certain state of the abstract CGS; whereas being undefined can be marked by having neither $p_\top$ nor $p_\bot$ present in the abstract state.

(iii) The abstract CGS is then verified in MCMAS against an SL formula. In more detail, our tool exploits the MCMAS extension for SL[1G], i.e., the *one goal* fragment [Cermák *et al.*, 2015], and the MCMAS extension for SLK, i.e., an epistemic extension of SL, [Cermák *et al.*, 2014].

Note that, to make use of the MCMAS model checker, our CGSs need to be first translated into Interpreted Systems [Fagin *et al.*, 1995]. In fact, MCMAS does not support CGSs, and it expects Interpreted Systems expressed using a domain specific language called Interpreted Systems Programming Language (ISPL). Thus, a pre-processing step before calling MCMAS is always required, where the CGS of interest is first automatically translated into its ISPL representation. This is only a technical detail, since CGSs and Interpreted Systems are equally expressive [Belardinelli *et al.*, 2020; Goranko and Jamroga, 2004].

It is important to report that the ISPL generation is performed on standard CGSs, not on their abstraction. Indeed, the abstract CGSs as described in Definition 4.1 cannot be used in MCMAS straight away, but need to be reprocessed first. To generate a CGS which can then be verified into MCMAS, the tool splits the 3-valued CGS into two CGSs. Such a split is determined by the SL formula under evaluation; that is, given an SL formula φ, we extract two sets of agents, E and U, whose strategies are only existentially and universally quantified in φ, respectively. By using these two sets, we split the 3-valued CGS into two CGSs: one CGS where agents in E use $must$-strategies, while agents in U use may-strategies; one CGS where agents in E use may-strategies, while agents in U use $must$-strategies. The first CGS can be used to prove the satisfaction of φ, while the second CGS can be used to prove the violation of φ. This follows from Definition 2.3, third and fourth bullet points.

As a consequence of how the verification is performed in practice, we remark an important difference between the theory presented in this paper and its implementation: the implementation handles SL formulas with arbitrary alternation of universal ($\forall x$) and existential ($\exists x$) quantifiers, as long as for each agent (a) in the formula, there is one single binding (a, x). Even though at the theoretical level our abstraction method can handle all SL formulas, at the implementation level this is not the case. In fact, our tool is based on MCMAS, and because of that, we cannot handle formulas where the agents need to swap between universally and existentially quantified strategies. This would require to modify MCMAS internally, which we leave as future work.

6 Experiments

We carried out the experiments on a machine with the following specifications: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4 cores 8 threads, 16 GB RAM DDR4. The case study we experimented on consists in a scheduler, where N agents, *i.e.*, processes (called P_i for $1 \leq i \leq N$) compete to get CPU time, while an $Arbiter$ agent decides which process to grant access (one at a time). The full description of the example can be found in [Cermák *et al.*, 2018]. The corre-

[4]https://github.com/AngeloFerrando/3-valuedSL

[5]https://www.json.org/

Processes	CGS				3-valued CGS					
	States	Transitions	Verification Time [sec] (SL[1G])	Verification Time [sec] (SLK)	States	Transitions [Must]	Transitions [May]	Abstraction Time [sec]	Verification Time [sec] (SL[1G])	Verification Time [sec] (SLK)
2	9	40	0.48	0.18	5	16	18	0.01	0.10	0.03
3	21	232	3725.56	2863.75	6	24	27	0.03	0.12	0.13
4	49	1376	T.O	T.O	7	34	38	0.10	0.24	0.19
5	113	7904	T.O	T.O	8	46	51	0.31	0.72	0.69
6	257	43520	T.O	T.O	9	60	66	1.24	2.08	1.12
7	577	230528	T.O	T.O	10	76	83	10.88	5.66	4.99
8	1281	1182208	T.O	T.O	11	94	102	107.89	8.40	6.67
9	2817	5903872	T.O	T.O	12	114	123	1087.14	29.37	26.81

Table 1: Experimental results for the scheduler case study (T.O. stands for Time Out).

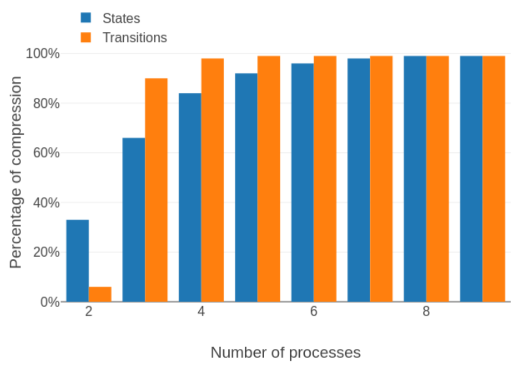

Figure 1: CGS compression in the scheduler case study.

sponding CGS can be parameterised over the number N of processes. Naturally, such parameter largely influences the size and complexity of the resulting CGS. Table 1 reports experimental results we obtained by applying our tool to the scheduler case study. We considered the verification of the same SL formula φ verified in [Cermák *et al.*, 2018], that is:

$$\varphi = \forall x, \vec{y}(Arbiter, x)(P_1, y_1)\dots(P_n, y_n)$$
$$G \neg \bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} rs_i \wedge rs_j$$

Intuitively, φ asserts that at most one process (P_i) owns the resource (rs) at any given point in time. In Table 1, each row refers to a fixed number of processes, from 2 to 9, used to generate the corresponding CGS. Each row also reports the number of states and transitions of the CGS, and the time required to perform its verification in MCMAS, both on the original CGS and its 3-valued abstraction, for comparison. For the latter, the time required to generate such an abstraction is reported. For the experiments with the scheduler, the abstraction is assumed to be guided by an expert of the system. In more detail, all states where at least one process is waiting to be selected by the scheduler are clustered together. This choice, as apparent in Table 1, largely reduces the number of states and transitions of the CGS. Nonetheless, this does not prevent the verification process to correctly conclude the satisfaction of φ on both the CGS and its 3-valued version, *i.e.*, the abstraction does not remove any information necessary to determine the satisfaction of φ. Table 1 also reports the execution time required for the actual verification of both the CGS and its 3-valued abstraction. As we can observe, without the abstraction step, the verification of the CGS times out when 3 processes are considered. In fact, MCMAS cannot model check φ in less than 3 hours, which was set as the time out (both for the SL[1G] and SLK extensions of MCMAS). Instead, thanks to the abstraction, the verification can be performed for up to 9 (a more realistic number of processes). Note that, the verification of the 3-valued CGS could have been performed for even larger numbers of processes. However, the CGS with 10 processes did not fit into the available memory of the machine used for the experiments; so, it was not possible to apply our technique to generate its 3-valued abstraction. Nonetheless, we expect the tool to handle even the case with 10 processes via abstraction. Figure 1 reports the data compression obtained in the scheduler case study. It is immediate to observe the huge compression obtained via abstraction. Indeed, the larger is the number of processes involved, the more significant is such compression. Note that, for more than 6 processes, the abstraction produces a CGS with $\sim$99% less states and transitions. Besides φ, we experimented with other specifications as well. Specifically, we carried out experiments over a large set of randomly-generated SL formulas. The goal of these experiments is to understand how many times our tool would return a conclusive answer (*i.e.*, not $\mathbf{u}$). We automatically synthesised 10,000 different SL formulas and verified them in the scheduler case study; where we kept the same abstraction as for Table 1. Over the 10,000 different SL formulas, the tool was capable of providing a defined truth value (either true or false) in the 83% of cases. Of course, this is a preliminary evaluation, which needs to be corroborated through additional experiments, also involving further real-world scenarios. Nonetheless, the results we obtained are promising, and allow us to empirically show the effectiveness of our approach, not only from a data-compression perspective, but also from a computational one.

7 Conclusion

The high complexity of the verification problem for Strategy Logic (being non-elementary) hinders the development of practical model checking tools and therefore its application in critical, real-life scenarios. As a consequence, it is of upmost importance to develop techniques to alleviate this computational burden and allow the use of Strategy Logic in concrete use cases, such as the scheduler scenario here analysed. This contribution is meant to be the first step in this direction.

Acknowledgements. W. Jamroga acknowledges the support of NCBR Poland, NCN Poland, and FNR Luxembourg under projects STV (POLLUX-VII/1/2019), SpaceVote (POLLUX-XI/14/SpaceVote/2023), and SAI (2020/02/Y/ST6/00064). A. Murano acknowledges the support of the PNNR FAIR project, the InDAM project "Strategic Reason- ing in Mechanism Design" and the PRIN 2020 Project RIPER.

References

[Alur *et al.*, 2002] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *JACM*, 49(5):672–713, 2002.

[Aminof *et al.*, 2012] B. Aminof, O. Kupferman, and A. Murano. Improved model checking of hierarchical systems. *Inf. Comput.*, 210:68–86, 2012.

[Aminof *et al.*, 2016] B. Aminof, A. Murano, S. Rubin, and F. Zuleger. Prompt alternating-time epistemic logics. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 258–267, 2016.

[Aminof *et al.*, 2018] B. Aminof, V. Malvone, A. Murano, and S. Rubin. Graded modalities in strategy logic. *Inf. Comput.*, 261:634–649, 2018.

[Aminof *et al.*, 2019] B. Aminof, M. Kwiatkowska, B. Maubert, A. Murano, and S. Rubin. Probabilistic strategy logic. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 32–38. ijcai.org, 2019.

[Ball and Kupferman, 2006] T. Ball and O. Kupferman. An abstraction-refinement framework for multi-agent systems. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS06)*, pages 379–388. IEEE, 2006.

[Belardinelli and Lomuscio, 2017] F. Belardinelli and A. Lomuscio. Agent-based abstractions for verifying alternating-time temporal logic with imperfect information. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1259–1267. ACM, 2017.

[Belardinelli and Malvone, 2020] F. Belardinelli and V. Malvone. A three-valued approach to strategic abilities under imperfect information. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 89–98, 2020.

[Belardinelli *et al.*, 2018] F. Belardinelli, A. Lomuscio, and V. Malvone. Approximating perfect recall when model checking strategic abilities. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, pages 435–444. AAAI Press, 2018.

[Belardinelli *et al.*, 2019] F. Belardinelli, A. Lomuscio, and V. Malvone. An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In *Proceedings of AAAI*, 2019.

[Belardinelli *et al.*, 2020] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with public actions against strategy logic. *Artif. Intell.*, 285:103302, 2020.

[Belardinelli *et al.*, 2022] F. Belardinelli, A. Lomuscio, V. Malvone, and E. Yu. Approximating perfect recall when model checking strategic abilities: Theory and applications. *J. Artif. Intell. Res.*, 73:897–932, 2022.

[Belardinelli *et al.*, 2023] F. Belardinelli, A. Ferrando, and V. Malvone. An abstraction-refinement framework for verifying strategic properties in multi-agent systems with imperfect information. *Artif. Intell.*, 316:103847, 2023.

[Berthon *et al.*, 2021] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi. Strategy logic with imperfect information. *ACM Trans. Comput. Log.*, 22(1):5:1–5:51, 2021.

[Bouyer *et al.*, 2019] P. Bouyer, O. Kupferman, N. Markey, B. Maubert, A. Murano, and G. Perelli. Reasoning about quality and fuzziness of strategic behaviours. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1588–1594. ijcai.org, 2019.

[Bruns and Godefroid, 1999] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Proceedings of Computer Aided Verification (CAV)*, pages 274–287, 1999.

[Bruns and Godefroid, 2000] G. Bruns and P. Godefroid. Generalized model checking: Reasoning about partial state spaces. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR00)*, volume 1877 of *LNCS*, pages 168–182. Springer-Verlag, 2000.

[Cermák *et al.*, 2014] P. Cermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Proceedings of the 26th International Conference on Computer Aided Verification (CAV14)*, volume 8559 of *Lecture Notes in Computer Science*, pages 525–532. Springer, 2014.

[Cermák *et al.*, 2015] P. Cermák, A. Lomuscio, F. Mogavero, and A. Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15)*, pages 2038–2044. AAAI Press, 2015.

[Cermák *et al.*, 2018] P. Cermák, A. Lomuscio, F. Mogavero, and A. Murano. Practical verification of multi-agent systems against SLK specifications. *Inf. Comput.*, 261:588–614, 2018.

[Chatterjee *et al.*, 2010] K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy Logic. *IC*, 208(6):677–693, 2010.

[Clarke *et al.*, 1994] E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[Fagin *et al.*, 1995] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[Ferrando and Malvone, 2021] A. Ferrando and V. Malvone. Strategy RV: A tool to approximate ATL model checking under imperfect information and perfect recall. In *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pages 1764–1766. ACM, 2021.

[Ferrando and Malvone, 2022] A. Ferrando and V. Malvone. Towards the combination of model checking and runtime verification on multi-agent systems. In *Proc. of PAAMS 2022*, volume 13616 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2022.

[Ferrando and Malvone, 2023] A. Ferrando and V. Malvone. Towards the verification of strategic properties in multi-agent systems with imperfect information. In *22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023 (to appear)*, 2023.

[Godefroid and Jagadeesan, 2003] P. Godefroid and R. Jagadeesan. On the expressiveness of 3-valued models. In *Proceedings of the 4th International Conference on Verification, Model Checkig, and Abstract Interpretation (VMCAI03)*, volume 2575 of *LNCS*, pages 206–222. Springer-Verlag, 2003.

[Goranko and Jamroga, 2004] V. Goranko and W. Jamroga. Comparing semantics for logics of multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[Grumberg *et al.*, 2007] O. Grumberg, M. Lange, M. Leucker, and S. Shoham. When not losing is better than winning: Abstraction and refinement for the full mu-calculus. *Inf. Comput.*, 205(8):1130–1148, 2007.

[Halpern and Shoham, 1986] J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. In *Proceedings 1st Annual IEEE Symposium on Logic in Computer Science, LICS86, Cambridge, MA, USA, 16–18 June 1986*, pages 279–292, Washington, DC, 1986. IEEE Computer Society Press.

[Huth *et al.*, 2001] M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: A foundation for three-valued program analysis. *ACM Transactions on Programming Languages and Systems*, pages 155–169, 2001.

[Jamroga *et al.*, 2016] W. Jamroga, B. Konikowska, and W. Penczek. Multi-valued verification of strategic ability. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, pages 1180–1189, 2016.

[Jamroga *et al.*, 2020] W. Jamroga, B. Konikowska, D. Kurpiewski, and W. Penczek. Multi-valued verification of strategic ability. *Fundam. Informaticae*, 175(1-4):207–251, 2020.

[Kouvaros and Lomuscio, 2017] P. Kouvaros and A. Lomuscio. Parameterised verification of infinite state multi-agent systems via predicate abstraction. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3013–3020. AAAI Press, 2017.

[Lomuscio and Michaliszyn, 2014] A. Lomuscio and J. Michaliszyn. An abstraction technique for the verification of multi-agent systems against ATL specifications. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR14)*, pages 428–437. AAAI Press, 2014.

[Lomuscio and Michaliszyn, 2015] A. Lomuscio and J. Michaliszyn. Verifying multi-agent systems by model checking three-valued abstractions. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS15)*, pages 189–198, 2015.

[Lomuscio and Michaliszyn, 2016] A. Lomuscio and J. Michaliszyn. Verification of multi-agent systems via predicate abstraction against ATLK specifications. In *Proc. of the 15th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS16)*, 2016.

[Lomuscio *et al.*, 2015] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. *Software Tools for Technology Transfer*, 2015. http://dx.doi.org/10.1007/s10009-015-0378-x.

[Łukasiewicz, 1920] Jan Łukasiewicz. O logice trojwartosciowej [On three-valued logic]. *Ruch Filozoficzny*, 5:170–171, 1920.

[Mogavero *et al.*, 2012] F. Mogavero, A. Murano, G. Perelli, , and M.Y. Vardi. What makes ATL* decidable? a decidable fragment of strategy logic. In *Proceedings of CONCUR*, pages 193–208, 2012.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions in Computational Logic*, 15(4):34:1–34:47, 2014.

[Raimondi and Lomuscio, 2005] F. Raimondi and A. Lomuscio. The complexity of symbolic model checking temporal-epistemic logics. In *Proceedings of Concurrency, Specification & Programming (CS&P)*, pages 421–432. Warsaw University, 2005.

[Shoham and Grumberg, 2004] S. Shoham and O. Grumberg. Monotonic abstraction-refinement for CTL. In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS04)*, volume 2988 of *Lecture Notes in Computer Science*, pages 546–560. Springer, 2004.

[Shoham and Grumberg, 2007] S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. *ACM Trans. Comput. Log.*, 9(1):1, 2007.