Strategy logic with simple goals: Tractable reasoning about strategies

Francesco Belardinelli, Wojciech Jamroga, Vadim Malvone, Aniello Murano

Strategy Logic with Simple Goals: Tractable Reasoning about Strategies

Francesco Belardinelli[1,2] , **Wojciech Jamroga**[3,4] , **Damian Kurpiewski**[3] ,
Vadim Malvone[2] and **Aniello Murano**[5]
[1] Imperial College London, UK
[2] Université d'Evry, France
[3] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[4] Interdisciplinary Centre for Security, Reliability, and Trust, SnT, University of Luxembourg
[5] Università degli studi di Napoli "Federico II", Italy
francesco.belardinelli@imperial.ac.uk, w.jamroga@ipipan.waw.pl, d.kurpiewski@ipipan.waw.pl,
vadim.malvone@univ-evry.fr, murano@na.infn.it

Abstract

In this paper we introduce *Strategy Logic with Simple Goals* (SL[SG]), a fragment of Strategy Logic that strictly extends Alternating-time Temporal Logic ATL by introducing arbitrary quantification over the agents' strategies. Our motivation comes from game-theoretic applications, such as expressing Stackelberg equilibria in games, coercion in voting protocols, as well as module checking for simple goals. We prove that model checking SL[SG] is $\mathbf{P}$-complete, the same as ATL. Thus, the extra expressive power comes at no computational cost as far as verification is concerned.

1 Introduction

Formal verification of multi-agent systems (MAS) has been a thriving area of investigation in the last two decades, which has led to a wealth of logics to specify the temporal, epistemic, and strategic capabilities of agents, including Alternating-time Temporal Logic [Alur *et al.*, 2002], possibly enriched with strategy contexts [Laroussinie and Markey, 2015], irrevocable strategies [Ågotnes *et al.*, 2007], or operators for individual and group knowledge [Jamroga and van der Hoek, 2004; Hoek and Wooldridge, 2003]; Coalition Logic [Pauly, 2002]; and Strategy Logic [Chatterjee *et al.*, 2010; Mogavero *et al.*, 2014]. Besides theoretical results, some model checking tools have also been developed [Alur *et al.*, 2001; Cimatti *et al.*, 2002; Kacprzak *et al.*, 2008; Huang and van der Meyden, 2014; Lomuscio *et al.*, 2015; Cermák *et al.*, 2015; Cermák *et al.*, 2018].

The verification of MAS generates a tension between two conflicting demands. On the one hand, we need an expressive language to capture subtle temporal, epistemic, and game-theoretic notions such as reachability, common and distributed knowledge, and solution concepts in games. On the other hand, we need formalisms with a tractable model checking problem, for which efficient verification algorithms can be implemented. The same tension appears in the seminal paper [Alur *et al.*, 2002] that introduced Alternating-time Temporal Logic. Two variants of the logic are presented: ATL* is rather expressive, but with $\mathbf{2EXPTIME}$ verification complexity. Its fragment ATL has less expressivity, but allows to model check strategic properties in polynomial time. This leads to a nice balance of expressivity and computational complexity – indeed, ATL model checking is supported by a number of tools [Alur *et al.*, 2001; Kacprzak *et al.*, 2008; Lomuscio *et al.*, 2015]. The question is: can we use a language that is more expressive than ATL, and still retains its polynomial-time complexity of model checking? We answer the question affirmatively in this paper.

Our starting point is Strategy Logic (SL) [Mogavero *et al.*, 2014], an expressive extension of ATL* that allows to characterize sophisticated game-theoretic solution concepts, e.g., Nash equilibria, dominant strategies, subgame-perfect equilibria, etc. Unfortunately, its model checking complexity is non-elementary, and hence highly unlikely to enable efficient implementation. Even restricted variants of SL, such as SL with memoryless strategies [Cermák *et al.*, 2014], *nested-goal*, *Boolean-goal*, and *one-goal* SL [Mogavero *et al.*, 2012] do not help much, as their model checking problems range from $\mathbf{2EXPTIME}$-complete to non-elementary [Mogavero *et al.*, 2014; Bouyer *et al.*, 2015; Gardy *et al.*, 2018].

We then propose to take one-goal SL, and further restrict goals to simple LTL formulas of type $\mathsf{X}\,\phi$ (*next* ϕ), $\phi\,\mathsf{U}\,\phi'$ (ϕ *until* ϕ'), and $\phi\,\mathsf{R}\,\phi'$ (ϕ *releases* ϕ'), similarly to the restriction of ATL* to ATL in [Alur *et al.*, 2002]. The result, *Strategy Logic with simple goals* (SL[SG]), can also be seen as the extension of ATL to arbitrary quantification on the agents' strategies. We use SL[SG] to capture Stackelberg equilibria in games, express coercion-resistance in voting protocols [Tabatabaei *et al.*, 2016], and characterize module checking for simple goals [Jamroga and Murano, 2014]. These are all relevant agent-related concepts, that cannot be expressed in ATL. Hence, SL[SG] is provably more expressive than ATL. Most importantly, we show that reasoning about SL[SG] is no more complex as ATL, as it also enjoys polynomial-time model checking. This is achieved by generalising the fixed-point procedures for ATL model checking. In consequence, Strategy Logic with simple goals offers an arguably better balance between expressivity and complexity than ATL. We further show the advantages by means of a case study, based on a simple voting scenario.

2 Logics for Strategies

We begin by recalling Strategy Logic (SL) [Mogavero *et al.*, 2014], as well as its relevant syntactic fragments. Then, we provide an interpretation to these languages by means of concurrent game structures (CGS), as it is customary.

Strategy Logic. Fix an infinite set *AP* of *atomic propositions* (atoms), a finite set *Ag* of *agents*, and an infinite set *Var* of variables $x_0, x_1, \ldots$ for strategies. *Formulas in Strategy Logic* are defined as follows, for $p \in AP$, $x \in Var$, $a \in Ag$:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \exists x\varphi \mid (x,a)\varphi$$

Next X and *until* U are *temporal operators*. The *strategy quantifier* $\exists x$ reads as "for some strategy x, ...", and the *binding operator* (x,a) intuitively means that "by using strategy x, agent a can achieve ...". Hereafter we use standard abbreviations, e.g., $\forall x\varphi$ for $\neg\exists x\neg\varphi$, as well as temporal operators *releases* R, *globally* G, and *eventually* F. We assume the standard definition of set *free*(φ) of free agents and variables appearing in a formula φ [Mogavero *et al.*, 2014]. Moreover, $shr(x,\varphi)$ denotes the set of agents that use strategy x in the evaluation of formula φ.

Models. Given sets *Ag* of agents and *AP* of atoms, a *concurrent game structure (CGS)* is a tuple $\mathbf{G} = \langle S, s_0, \{Act_a\}_{a\in Ag}, \tau, L\rangle$ such that: S is a non-empty finite set of *states* and $s_0 \in S$ is the *initial state* of $\mathbf{G}$; for every agent $a \in Ag$, Act_a is a finite non-empty set of *actions*, and $ACT = \prod_{a\in Ag} Act_a$ is the set of *joint actions*; $\tau : S \times ACT \to S$ is the *transition function*; finally, $L : S \to 2^{AP}$ is the *labelling*. A *path* is a (finite or infinite) sequence $\pi \in S^* \cup S^\omega$ such that for every $j \geq 1$, $\pi_{j+1} = \tau(\pi_j, \vec{\alpha}_j)$ for some joint action $\vec{\alpha}_j \in ACT$. We distinguish between finite paths, or *histories*, and infinite paths, or *computations*. 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.

Strategies. A *memoryfull strategy* for an agent $a \in Ag$, or a-strategy, is a function $\sigma : S^+ \to Act_a$. The set of all strategies, for all agents, is denoted as $\Sigma(\mathbf{G})$. A *joint strategy* σ_{Ag} assigns a strategy for a to every agent $a \in Ag$. By the above definitions, given a strategy σ for an agent a, if a different agent b is such that the range of strategy σ is a subset of Act_b (i.e., $ran(\sigma) \subseteq Act_b$), then intuitively also agent b can use strategy σ. Finally, an *assignment* is a function $\chi : Var \cup Ag \to \Sigma(\mathbf{G})$ such that for every agent $a \in Ag$, $\chi(a)$ is a strategy for a. For $z \in Var \cup Ag$ and $\sigma \in \Sigma(\mathbf{G})$, 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 $h \in S^+$, an assignment χ defines a unique computation $\lambda(h,\chi) = h \xrightarrow{\langle\chi(a)(h)\rangle_{a\in Ag}} s_1 \xrightarrow{\langle\chi(a)(h\cdot s_1)\rangle_{a\in Ag}} s_2 \ldots$ starting with h and consistent with χ.

Semantics. Given a CGS $\mathbf{G}$, we inductively define the satisfaction relation $(\mathbf{G}, h, \chi) \models \varphi$ where h is a history, φ is a formula, and χ is an assignment such that for every $x \in Var$, $\chi(x)$ is a strategy for all agents in $shr(x,\varphi)$ (we omit the standard clauses for Boolean operators):

$(\mathbf{G}, h, \chi) \models p$ iff $p \in L(last(h))$
$(\mathbf{G}, h, \chi) \models \mathsf{X}\,\varphi$ iff $(\mathbf{G}, \lambda(h,\chi)_{\leq|h|+1}, \chi) \models \varphi$
$(\mathbf{G}, h, \chi) \models \varphi_1\,\mathsf{U}\,\varphi_2$ iff for some $i \geq |h|$, $(\mathbf{G}, \lambda(h,\chi)_{\leq i}, \chi) \models \varphi_2$, and for all j, $|h| \leq j < i$ implies $(\mathbf{G}, \lambda(h,\chi)_{\leq j}, \chi) \models \varphi_1$
$(\mathbf{G}, h, \chi) \models \exists x\varphi$ iff for some strategy σ for every agent in $shr(x,\varphi)$, $(\mathbf{G}, h, \chi^x_\sigma) \models \varphi$
$(\mathbf{G}, h, \chi) \models (x,a)\varphi$ iff $(\mathbf{G}, h, \chi^a_{\chi(x)}) \models \varphi$

We write $(\mathbf{G}, h) \models \varphi$ iff $(\mathbf{G}, h, \chi) \models \varphi$ for every assignment χ, and $\mathbf{G} \models \varphi$ iff $(\mathbf{G}, s_0) \models \varphi$.

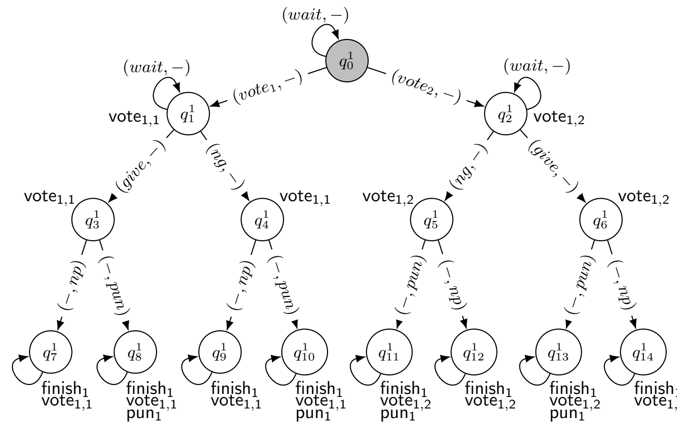

Figure 1: A simple model of voting and coercion

Relevant Fragments. One-Goal Strategy Logic (SL[1G]) is the syntactic fragment of SL where each temporal subformula is preceded by a *binding prefix* that mentions all agents in *Ag*, and by a *quantification prefix*[1] referring to all the preceding strategy variables [Mogavero *et al.*, 2012]. Furthermore, ATL* can be seen as the fragment of SL[1G] that admits at most one alternation of strategic quantifiers (either $\exists\forall$ or $\forall\exists$), and does not allow to bind different agents to the same variable. Finally, ATL is the fragment of ATL* where each group of strategic quantifiers is followed by exactly one temporal operator. Interestingly, SL[1G] has been proved strictly more expressive than ATL*, while having the same complexity of the model checking problem [Mogavero *et al.*, 2014]. In this paper, we look for an analogous extension of ATL.

Example 1 (Simple Voting) *Consider a simple model of voting, inspired by [Jamroga* et al.*, 2017a]. There are k voters $v_1, \ldots, v_k$ and a single coercer c. At the beginning, each voter decides to* wait *or* cast her vote *for one of the n candidates. Then, she can* wait *again,* give her vote receipt *to the coercer or* refuse to give it*. Finally, the coercer can either* punish *the voter, or* refrain from punishment*. The interaction between the coercer and different voters is independent. The CGS $SV_{1,2}$ for $k = 1$ and $n = 2$ is presented in Figure 1.*

An example formula that holds in $SV_{2,2}$ is $\forall x_c \exists x_{v_1} \forall x_{v_2} (x_c, c)(x_{v_1}, v_1)(x_{v_2}, v_2)\,\mathsf{F}(\mathsf{voted_{v,1}} \wedge \neg\mathsf{pun})$*, expressing that, for every strategy of c, the first voter has a counterstrategy ensuring that she can vote for candidate 1 without getting punished, regardless of the other voter. The counterstrategy is simply to vote for 1, and then execute* wait*.*

[1] We will formally introduce both notions in the next section.

3 Strategy Logic with Simple Goals

Inspired by the relationship between the Alternating-time Temporal Logics ATL^* and ATL, we introduce a novel restriction of SL[1G] to "simple" goals.

3.1 The Formal Language

We begin with some terminology. A *binding prefix* over sets $A \subseteq Ag$ of agents and $V \subseteq Var$ of variables is a finite sequence $\flat \in \{(x, a) \mid a \in A \text{ and } x \in V\}^{|A|}$ of length $|\flat| = |A|$, such that every agent $a \in A$ occurs exactly once in $\flat$. In contrast, the same variable $x \in V$ can occur several times in $\flat$, i.e., intuitively, the same strategy denoted by x can be used by several agents in A. A *quantification prefix* over a set $V \subseteq Var$ of variables is a finite sequence $\wp \in \{\exists x, \forall x \mid x \in V\}^{|V|}$ of length $|\wp| = |V|$ such that every variable $x \in V$ occurs exactly once in $\wp$. Then, $Qnt(V) \subset \{\exists x, \forall x \mid x \in V\}^{|V|}$ and $Bnd(A) \subset \{(x, a) \mid a \in A \text{ and } x \in Var\}^{|A|}$ denote the sets of all quantification and binding prefixes over variables in V and agents in A.

Definition 2 (SL[SG]) *The formulas in* Strategy Logic with simple goals *are defined in BNF as follows, where* $\flat \in Bnd(Ag)$, $\wp \in Qnt(free(\flat\varphi))$*:*

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

By Def. 2, SL[SG] restricts SL[1G] to simple LTL objectives of type $\mathsf{X}\,\varphi$, $\varphi\,\mathsf{U}\,\varphi'$, and $\varphi\,\mathsf{R}\,\varphi'$ (the latter can be introduced by using U and dual quantification). We also consider the fragment SL^-[SG], for which Def. 2 is restricted to binding prefixes $\flat$ where every variable occurs at most once. That is, one cannot bind different agents with the same strategy. Further, we define fragments SL_n[SG], for which quantification prefixes $\wp$ are restricted to at most n alternations of existential and universal quantifiers. Finally, we introduce fragments SL^-_n[SG] as the intersection of SL^-[SG] and each SL_n[SG].

Example 3 *Clearly, the formula in Example 1 is a formula of* SL[SG]*, more precisely of* SL^-_2[SG].

In Fig. 2 we summarize the main syntactic inclusions between our fragments. We also remark that ATL can be thought of as the fragment of SL[SG] in which the clauses for formulas of type $\wp\flat\,\mathsf{X}\,\varphi$ and $\wp\flat(\varphi\,\mathsf{U}\,\varphi)$ are restricted to quantification prefixes $\wp$ with at most one alternation, as well as binding prefixes $\flat$ where every variable occurs at most once. In fact, ATL corresponds exactly to SL^-_1[SG], whereas Computation Tree Logic (CTL) corresponds to SL^-_0[SG].

3.2 Expressiveness

An important observation that formally justifies our study is that SL[SG] is strictly more expressive than ATL. That is, SL[SG] allows to characterize all the properties expressible in ATL, but not vice versa. We also show that SL[SG], SL^-[SG], and SL_1[SG] all differ with respect to their expressiveness.

Consider two logical systems $\mathcal{L}_1$ and $\mathcal{L}_2$. $\mathcal{L}_1$ is *at least as expressive as* $\mathcal{L}_2$ (written $\mathcal{L}_2 \preceq_e \mathcal{L}_1$) if every formula of $\mathcal{L}_2$ can be equivalently translated to some formula of $\mathcal{L}_1$. Moreover, $\mathcal{L}_1$ is *at least as distinguishing as* $\mathcal{L}_2$ ($\mathcal{L}_2 \preceq_d \mathcal{L}_1$) if every pair of models that can be distinguished by a formula of $\mathcal{L}_2$ can also be distinguished by some formula of $\mathcal{L}_1$.

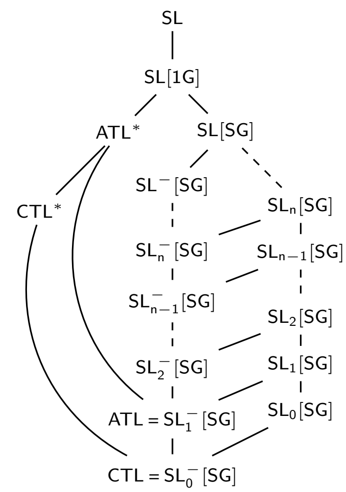

Figure 2: Syntactic inclusions for fragments of SL

Theorem 4 SL^-_2[SG] *(and thus also* SL[SG]*) has strictly greater expressive and distinguishing power than* ATL.

Proof. The embedding of ATL into SL^-_2[SG] is straightforward. To show that ATL does not cover the distinguishing power (and hence also the expressive power) of SL^-_2[SG], we can use the counterexample from the proof of [Mogavero *et al.*, 2014, Theorem 4.3]. □

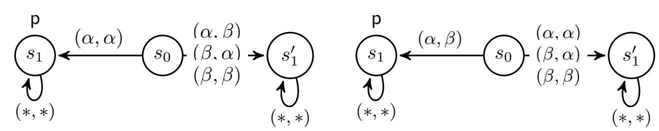

Figure 3: Models M_1 and M_2 in the proof of Theorem 5

Interestingly, we also have that for every $n \in \mathbb{N}$, SL_n[SG] is strictly more expressive than SL^-_n[SG]. That is, being able to bind the same strategy to different agents strictly increases the expressive power of Strategy Logic with simple goals.

Theorem 5 *For every* $n \geq 0$*, we have:*

1. SL^-_n[SG]$\prec_e \mathsf{SL}_\mathsf{n}$[SG]*, and therefore* SL^-_n[SG]$\prec_d \mathsf{SL}_\mathsf{n}$[SG]*;*
2. SL^-[SG]$\prec_e$ SL[SG]*, and therefore* SL^-[SG] $\prec_d$ SL[SG]*;*

Proof. We adapt the proof of [Belardinelli *et al.*, 2018, Lemma 32.1]. Clearly, SL^-_n[SG] $\preceq_e$ SL_n[SG] and therefore also SL^-_n[SG] $\preceq_d$ SL_n[SG]. Since SL_0[SG] $\preceq_d$ SL_n[SG] and SL^-_n[SG] $\preceq_d$ SL^-[SG], to show that SL_n[SG] $\not\preceq_d$ SL^-_n[SG], we prove that:

$$\mathsf{SL}_0[\mathsf{SG}] \not\preceq_d \mathsf{SL}^-[\mathsf{SG}] \tag{1}$$

To do so, we provide two models that satisfy the same formulas in SL^-[SG] but are distinguished by a formula of SL_0[SG]. Consider the CGS's M_1 and M_2 with $Ag = \{1, 2\}$, depicted in Fig. 3. Since SL^-[SG] can be shown invariant under renaming of action labels, both models satisfy the same formulas of SL^-[SG]. However, M_1 satisfies the SL_0[SG]-formula $\exists x(x, 1)(x, 2)\ \mathsf{X}\, p$, while M_2 does not. Thus, SL_0[SG] $\not\preceq_d$ SL^-[SG] and therefore also SL_n[SG] $\not\preceq_d$ SL^-_n[SG] and SL_n[SG] $\not\preceq_e$ SL^-_n[SG] for every $n \in \mathbb{N}$.

As regards 2., clearly, $\mathsf{SL}^-[\mathsf{SG}] \preceq_e \mathsf{SL}[\mathsf{SG}]$, and therefore $\mathsf{SL}^-[\mathsf{SG}] \preceq_d \mathsf{SL}[\mathsf{SG}]$. Since $\mathsf{SL}_0[\mathsf{SG}] \preceq_d \mathsf{SL}[\mathsf{SG}]$ and $\mathsf{SL}_0[\mathsf{SG}] \not\preceq_d \mathsf{SL}^-[\mathsf{SG}]$ (item (1)), then $\mathsf{SL}[\mathsf{SG}] \not\preceq_d \mathsf{SL}^-[\mathsf{SG}]$. So, $\mathsf{SL}^-[\mathsf{SG}] \prec_e \mathsf{SL}[\mathsf{SG}]$, and thus $\mathsf{SL}^-[\mathsf{SG}] \prec_d \mathsf{SL}[\mathsf{SG}]$. □

The discussion above highlights logics $\mathsf{SL}_0[\mathsf{SG}]$ and $\mathsf{SL}_1[\mathsf{SG}]$ as variants of $\mathsf{CTL} = \mathsf{SL}_0^-[\mathsf{SG}]$ and $\mathsf{ATL} = \mathsf{SL}_1^-[\mathsf{SG}]$ that allow for strategy sharing. Such extensions have not been considered in the literature. Nonetheless, by Theorem 5, they are strictly more expressive than CTL and ATL respectively.

3.3 Motivating Examples and Applications

Strategy Logic enables to address complex interaction between strategies of different players, who may pursue adversarial, collaborative, or uncorrelated goals. Also, the goals can be based on sophisticated temporal patterns. This expressivity comes at the expense of tractability, or even decidability of decision problems for SL. At the other end, we have ATL where only a single alternation is allowed in strategy quantification. Moreover, the agents' goals can only be phrased in terms of reachability. This results in a somewhat rigid specification language, albeit one with a big advantage: the model checking problem becomes tractable.

With SL[SG], we extend ATL by allowing arbitrarily many alternations of strategy quantifiers. We argue that the extra expressivity is useful, i.e., it allows for specification of *relevant* properties of agent interaction in multi-agent systems. To this end, we propose a number of motivating examples.

Example 6 (Stackelberg equilibria) Stackelberg games *concern scenarios where one player (the* leader*) exposes his strategy first, and the other player (the* follower*) adapts to that strategy by choosing her best response. For example, coercion in voting often has a Stackelberg structure: the coercer must lay out his coercion strategy in a believable way to force the voter to vote as requested. A desirable property in voting systems is that the voter can resist coercion, i.e., for every strategy of the coercer (c) there exists a response of the voter (v) such that for every run of the environment (e) the voter will have voted as she intended without being punished [Tabatabaei* et al.*, 2016]. This is captured by the following formula of* SL[SG]*:*

$$\forall x_c \exists x_v \forall x_e\ (x_c, c)(x_v, v)(x_e, e)\, \mathsf{F}(\mathsf{voted_{v,i}} \wedge \neg \mathsf{punished}).$$

Example 7 (Coercion in voting, ctd.) *A more realistic model of coercion allows the coercer to split his strategy into a public and a private part. We can simulate that by splitting the model of the coercer into two agents: cpub and cprv, each responsible for different parts of the coercion strategy. Moreover, some voting protocols include a process that can add* decoy votes *after the election to deceive the coercer. This can be incorporated into our specification of coercion-resistance as follows:*

$$\forall x_{c_1} \exists x_v \forall x_{c_2} \exists x_d \forall x_e\ (x_{c_1}, cpub)(x_v, v)(x_{c_2}, cprv)$$
$$(x_d, d)(x_e, e)\mathsf{F}\,(\mathsf{voted_{v,i}} \wedge \neg \mathsf{punished}).$$

This specification can intuitively be read as: for every public strategy of the coercer (x_{c_1}), there exists a response by the voter (x_v) such that, no matter what the coercer does privately (x_{c_2}), the decoy process d *can use decoy votes to make sure that the voter votes as she likes without being punished.*

Example 8 (Module checking for strategic abilities) *The problem of* module checking *asks, for an open system embedded in a nondeterministic environment, whether the correctness specification holds for every possible strategy of the environment*[2]*. It has been proved that module checking of either temporal or strategic specifications cannot be expressed in* ATL *[Jamroga and Murano, 2014]. The ability of an autonomous taxi (t) to serve an unbounded stream of customers (e) so that no accident occurs, regardless of what the agents $a_1, \ldots, a_k$ do, can be captured as*

$$\forall x_e \exists x_t \forall x_{a_1} ... \forall x_{a_k}\ (x_e, e)(x_t, t)(x_{a_1}, a_1)...(x_{a_k}, a_k)\ \mathsf{G}\,\neg \mathsf{crash}.$$

A careful reader may notice that the results in Section 3.2 indeed apply to the above specifications. That is, the formulas in Examples 6–8 cannot be equivalently expressed in ATL.

4 Model Checking

Given a CGS $\mathbf{G}$ and a formula ϕ in language L, the *model checking problem w.r.t.* L consists in determining whether $\mathbf{G} \models \phi$. In this section we study model checking for Strategy Logic with simple goals and its fragments. These results are essential for applications of SL[SG] to the verification of multi-agent systems, including the specifications in Examples 6–8. Our main theoretical result is that verification of SL[SG] is tractable with respect to the size of the CGS (i.e., the number of its transitions) and the length of the formula.

We first prove that the fixed-point characterisation of ATL carry over to SL[SG].

Proposition 9 *The following formula is a validity in* SL[SG]*:*

$$\wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2) \quad \leftrightarrow \quad \varphi_2 \vee (\varphi_1 \wedge \wp\flat\,\mathsf{X}\,\wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2))$$

Proof. The proof makes use of the preimage function $Pre()$ that is defined in Algorithm 2 and will be described in detail in the proof of Theorem 10. As regards the implication from left to right, suppose that $\wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$ holds in some state s. Then, if φ_2 does not hold in s, φ_1 does. Moreover, consider set $Y \subseteq S$ such that $s \in Pre(\wp, \flat, Y, \epsilon)$, i.e., the successors of s that are consistent with the quantification and binding prefixes. Since $(\mathbf{G}, s) \models \wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$ and $(\mathbf{G}, s) \not\models \varphi_2$, for every $s' \in Y$, we have $(\mathbf{G}, ss') \models \wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$: indeed every strategy σ_i witnessing an existential quantifier Q_i in $\wp$ at s, also witnesses Q_i in history ss'. In particular, $(\mathbf{G}, ss') \models \wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$, and finally, $(\mathbf{G}, s) \models \wp\flat\,\mathsf{X}\,\wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$ by the way s' was chosen.

As for the implication from right to left, if φ_2 holds at state s, then $\wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$ is also the case. On the other hand, suppose that $(\mathbf{G}, s) \models \varphi_1 \wedge \wp\flat\,\mathsf{X}\,\wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$. In particular, every existential quantifier Q_i in $\wp$ is witnessed in s by some strategy σ_i, and in every "successor" state $s' \in Y$, Q_i is witnessed in history ss' by some (possibly different) strategy σ'_i. Then, define strategies σ''_i such that $\sigma''_i(s) = \sigma_i(s)$ and $\sigma''_i(s \cdot h) = \sigma'_i(h)$ for all $h \in S^+$. In particular, each σ''_i witnesses Q_i at s, that is, $(\mathbf{G}, s) \models \wp\flat(\varphi_1 \,\mathsf{U}\, \varphi_2)$. Observe the essential use of perfect recall in the construction of strategy σ''_i from σ_i and σ'_i. □

[2]Admittedly, to represent module checking properly we need CGSs with non-deterministic transitions. Nonetheless, the characterization of module checking in SL[SG] would remain the same.

Algorithm 1 SL[SG] Model Checking

```
procedure ModelChecking(G, φ)
  for all φ' in Sub(φ) do
    case φ' = p: [φ'] := Reg(p);
    case φ' = ¬θ: [φ'] := S \ [θ];
    case φ' = θ1 ∧ θ2: [φ'] := [θ1] ∩ [θ2];
    case φ' = ℘♭ X θ: [φ'] := Pre(℘, ♭, [θ], ϵ);
    case φ' = ℘♭(θ1 U θ2):
      Y := ∅; Z := [θ2];
      while Z ⊈ Y do
        Y := Y ∪ Z;
        Z := Pre(℘, ♭, Y, ϵ) ∩ [θ1];
      end while
      [φ'] := Y;
  end for
  return [φ];
end procedure
```

By using Prop. 9 we can prove the main theoretical result.

Theorem 10 *Model checking* SL[SG] *is* $\mathbf{P}$*-complete.*

Proof. Recall that CTL corresponds to the SL_0^-[SG] fragment of SL[SG]. Then, the lower bound follows immediately from the $\mathbf{P}$-hardness of model checking CTL [Katoen, 2008].

As for the upper bound, Algorithm 1 shows a procedure for model checking SL[SG], which manipulates sets of states in S. The procedure is inspired by the standard model checking algorithm for ATL [Alur *et al.*, 2002], but the preimage operator for the ATL modality $\langle\langle A\rangle\rangle$X is now replaced by the Pre operator in Algorithm 2 for an arbitrary block of strategy quantifiers. Specifically, Algorithm 1 uses the following primitive operations:

- The function Sub returns a sequence of syntactic subformulas of a given formula φ.
- The function Reg returns the set $\{s \in S \mid p \in L(s)\}$ of states $s \in S$ labelled with a given atom $p \in AP$.
- The function Pre, when given a quantification prefix $\wp = Q_1x_1 \ldots Q_kx_k$, a binding prefix $\flat = (x_1, a_{1,1}) \ldots (x_1, a_{1,m_1}) \ldots (x_k, a_{k,1}) \ldots (x_k, a_{k,m_k})$[3], a set $Y \subseteq S$ of states, and a tuple $\vec{\alpha}$ of actions for all agents $a \in Ag$ such that (x, a) appears in $\flat$ and x does not appear in $\wp$, returns the set of states $s \in S$ from which there exist transitions consistent with $\wp$, $\flat$, and $\vec{\alpha}$, ending up in Y. Formally, Pre is a recursive function defined on the length of $\wp$. For the base case $\wp = \epsilon$, notice that $\vec{\alpha}$ is a joint action for all agents in Ag, and therefore Pre returns the set of states $s \in S$ such that $\tau(s, \vec{\alpha}) \in Y$. As for the recursive step, $s \in Pre(\wp, \flat, Y, \vec{\alpha})$ iff for $\wp_1 = \exists$ (resp. $\forall$), for some (resp. every) action β, it is the case that $s \in Pre(\wp_{>1}, \flat, Y, update(\flat, \vec{\alpha}, var(\wp_1), \beta))$, where function $update(\flat, \vec{\alpha}, x, \beta)$, given a binding $\flat$, a tuple $\vec{\alpha}$ of actions, variable x, and action β, returns a tuple $\vec{\alpha}'$ of actions such that for every $a \in Ag$, if (x, a) appears in $\flat$ then $\vec{\alpha}'_a = \beta$, otherwise $\vec{\alpha}'_a = \vec{\alpha}_a$. For more details on function Pre, see Algorithm 2.
- Union, intersection, difference, and inclusion test for sets of states.

[3]The parameter m_i represents the number of agents assigned to variable x_i, for each $1 \leq i \leq k$.

Algorithm 2 Preimage of a set Y of states

procedure $Pre(\wp, \flat, Y, \vec{\alpha})$
  if $\wp = \epsilon$ **then return** $\{s \in S \mid \tau(s, \vec{\alpha}) \in Y\}$;
  else if $\wp_1 = \exists$ **then return**
    $\bigcup_{\beta \in \bigcap_{(\mathsf{var}(\wp_1), a) \in \flat} Act_a} Pre(\wp_{>1}, \flat, Y, update(\flat, \vec{\alpha}, \mathsf{var}(\wp_1), \beta))$;
  else return
    $\bigcap_{\beta \in \bigcap_{(\mathsf{var}(\wp_1), a) \in \flat} Act_a} Pre(\wp_{>1}, \flat, Y, update(\flat, \vec{\alpha}, \mathsf{var}(\wp_1), \beta))$;
  end if
end procedure

Algorithm 1 works bottom-up on the structure of the formula; the cases of interest are for strategy formulas. For $\varphi' = \wp\flat\, \mathsf{X}\, \theta$, the procedure calls function Pre to compute the set of states that are "bound" to end up in satisfaction set $[\theta]$. As regard $\varphi' = \wp\flat(\theta_1 \,\mathsf{U}\, \theta_2)$, the procedure computes the least fixed-point of operator $F(Z) = [\theta_2] \cup ([\theta_1] \cap \wp\flat\, \mathsf{X}\, \wp\flat(Z))$. We observe that, since F is monotone, such a fixed-point always exists. Further, Algorithm 1 runs in polynomial time in the size of the CGS and the size of the formula[4]. Termination of Algorithm 1 is guaranteed, as the state space S is finite. Soundness and completeness can be proved by induction on the structure of the input formula φ by using Proposition 9 for the case $\varphi' = \wp\flat(\theta_1 \,\mathsf{U}\, \theta_2)$. □

By Theorem 10 we immediately obtain that for all fragments of SL[SG] in Fig. 2 model checking is $\mathbf{P}$-complete.

5 Experimental Evaluation

With SL[SG], we propose a logic that enhances the expressivity of ATL while enjoying the same, polynomial-time complexity of model checking. Alternatively, we sacrifice some expressiveness of SL[1G] in order to reduce the verification complexity from highly intractable (**2EXPTIME**-complete) to tractable. In consequence, one would expect that the verification algorithm for SL[SG], presented in Section 4, should perform better than the general algorithm that handles the whole "one-goal" fragment of SL [Cermák *et al.*, 2015].

However, theoretical complexity results rely on *worst case complexity*. Thus, it can be that the practical performance of SL[1G] model checking is much better than the **2EXPTIME** classification suggests. In particular, one may ask if the specialized SL[SG] algorithm of the previous section performs significantly better than the general algorithm in [Cermák *et al.*, 2015] on the formulas of SL[SG]. To answer this question, we have conducted a series of experiments with a scalable benchmark, based on the simple voting and coercion scenario of Examples 1 and 6.

We consider models $ESV_{k,n}$ (Extended Simple Voting with k voters and n candidates), constructed as follows. The initial state q_0 is handled by a new *election authority ea*, who decides whether to implement *low* or *high* anti-coercion protection. The former choice leads to a copy of the $SV_{k,n}$ CGS. The latter proceeds to a modified copy of the same model, with proposition $\mathsf{pun_i}$ satisfied only in the states where the coercer has decided to punish the voter *and the voter gave him her voting receipt*.

[4]Notice, however, that the number of transitions in a CGS can be exponential w.r.t. the number of agents.

#v	#states	SL[SG]		SL[1G]	SLK
		tgen	tverif	tg+tv	tg+tv
1	29	0.001	0.002	0.42	0.04
2	395	0.02	0.17	27.76	0.08
3	5573	0.98	45.31	5247.65	530.40
4	79187	30.40	8502.12	timeout	timeout
5	1130669	805.38	timeout	timeout	timeout

Table 1: Experimental results for the simple voting model

Moreover, we consider the formula:

$$\exists x_{ea} \forall x_c \exists x_{v_1} \forall x_{v_2} \ldots \forall x_{v_k} (x_{ea}, ea)(x_c, c)(x_{v_1}, v_1) \ldots (x_{v_k}, v_k) \\ \mathsf{F}(\mathsf{finish}_1 \wedge \mathsf{voted}_{1,1} \wedge \neg \mathsf{pun}_1)$$

expressing that the election authority can make sure that, for every strategy of the coercer, voter 1 has a strategy to eventually complete her participation in the election, having voted for candidate 1 without getting punished. Clearly, the formula is true in every model $ESV_{k,n}$ (the strategy for e being high protection, and for the voter to never give away her vote). In the experiments, we have only used models with $n = 2$. Thus, the number of voters (k) was the sole scaling factor.

We implemented the model checking algorithm in Section 4 in Python 3, using explicit representation of states and transitions. The tool is available on-line[5]. For the general SL[1G] algorithm, we used the MCMAS-SL[1G] extension of MCMAS [Cermák *et al.*, 2015] and the MCMAS-SLK extension. The experiments were conducted on an Intel Core i7-6700 CPU with dynamic clock speed of 2.60–3.50 GHz and 32 GB RAM, running under 64bit Windows 10. The timeout was set to *5 hours*. The experimental results are presented in Tab. 1. All times are given in seconds. The first two columns describe the model configuration and its size. The next two columns show the model generation time and the verification time for our algorithm. Finally, the last column presents the performance of the general SL[1G] tool.

The results show significant improvement in running time when using the specialized algorithm (by orders of magnitude). Moreover, our implementation was able to verify models of up to nearly $80,000$ states, while the more general tool handled only 10 times smaller models.

6 Conclusions

In this paper we advanced the state of the art on tractable logics for strategic reasoning in MAS. Inspired by the restriction of the Alternating-time Temporal Logic ATL* to ATL, we introduced Strategy Logic with simple goals (SL[SG]), a fragment of Strategy Logic that only allows for simple goals of form $\mathsf{X}\,\phi$, $\phi \,\mathsf{U}\, \phi'$, and $\phi \,\mathsf{R}\, \phi'$. Alternatively, SL[SG] can be seen as the extension of ATL to arbitrary quantification on the agents' strategies. Such an extension is motivated by relevant specification requirements in Game Theory, voting protocols, and formal verification. In particular, we showed that subtle notions, such as Stackelberg equilibria, coercion-resistance, module checking, can all be naturally represented in SL[SG].

Then, we analysed the comparative expressivity of SL[SG], and introduced the family of logics $\mathsf{SL_n}$[SG] and $\mathsf{SL_n^-}$[SG], for $n \in \mathbb{N}$, inbetween CTL and SL[SG]. We pointed out that ATL is strictly less expressive than SL[SG] and even $\mathsf{SL_2^-}$[SG] (Theorem 4). We showed that strategy binding strictly increases the expressive power of our fragments (Theorem 5). Most importantly, the enhanced expressivity of SL[SG] w.r.t. ATL comes at no extra computational cost: model checking SL[SG] is $\mathbf{P}$-complete, the same complexity as ATL (Theorem 10). Because of its tractable model checking problem, SL[SG] lends itself to an efficient implementation. As an evidence, in Section 5 we presented an implementation of the proposed SL[SG] model checking algorithm and conducted a series of experiments that have showed striking improvements in running time with respect to the implementation of the SL[1G] algorithm in [Cermák *et al.*, 2015].

Related Work. Logics for strategies are a powerful tool for strategic reasoning in MAS. Their major bottleneck is the high complexity of the related decision problems. To overcome this, two main lines of research have been followed recently. One deals with semantical restrictions, mainly by limiting the number of strategies to evaluate [Chatterjee *et al.*, 2014; Vester, 2013; Brihaye *et al.*, 2009; Jamroga *et al.*, 2017b; Bruyère *et al.*, 2013]; while the other deals with the syntax, e.g., by looking at "simpler" fragments [Alur and La Torre, 2004; Pauly, 2002; Mogavero *et al.*, 2014]. In this paper, we focused on this second line. Among such fragments, one of the most studied is ATL, a proper sublanguage of ATL* [Alur *et al.*, 2002]. Recently, the syntactic restriction to simple goals behind the success of ATL has also been investigated in the context of more powerful logics for strategic reasoning. In [Malvone *et al.*, 2018] the authors consider the simple vanilla fragment of Graded SL[1G] ($GSL[1G]$), but for two agents only. Hence, their contribution is orthogonal to ours: it is more general, as they can count strategies, but also more restricted, since it only considers two agents. Further, in [Laroussinie and Markey, 2015] the authors consider the ATL-like restriction of ATL* with strategy contexts. However, this restriction does not reduce the expressive power nor the complexity of the related decision problems.

Future Work. A natural direction for future work is to look for more expressive, yet tractable, formalisms. For instance, [Bulling and Jamroga, 2010] introduced ATL^+ that sits between ATL and ATL*. This fragment includes only formulas where each temporal operator is followed by a state formula, and allows cooperation modalities to be followed by a Boolean combination of path formulas. Notably, model checking ATL^+ is PSPACE-complete. We envisage an extension of SL[SG] similar to ATL^+, hopefully with a PSPACE-complete model checking problem too. We also plan to release our model checking prototype as a verification tool.

Acknowledgements

F. Belardinelli acknowledges the support of the ANR JCJC Project SVeDaS (ANR-16-CE40-0021). W. Jamroga and D. Kurpiewski acknowledge the support of the National Centre for Research and Development NCBiR, Poland, and the National Research Fund FNR, Luxembourg, under the PolLux/FNR-INTER project VoteVerif (POLLUX-IV/1/2016).

[5] https://github.com/slsgijcai19/StrategyLogicSimpleGoals.

References

[Ågotnes *et al.*, 2007] T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-Time Temporal Logics with Irrevocable Strategies. In *TARK'07*, pages 15–24, 2007.

[Alur and La Torre, 2004] R. Alur and S. La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.

[Alur *et al.*, 2001] R. Alur, L. de Alfaro, R. Grosu, T. Henzinger, A. Thomas, M. Kang, C. Kirsch, R. Majumdar F. Mang, and B-Y. Wang. jMocha: A model checking tool that exploits design structure. In *ICSE'01*, pages 835–836. IEEE, 2001.

[Alur *et al.*, 2002] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[Belardinelli *et al.*, 2018] F. Belardinelli, C. Dima, and A. Murano. Bisimulations for logics of strategies: A study in expressiveness and verification. In *KR'18*, pages 425–434, 2018.

[Bouyer *et al.*, 2015] P. Bouyer, P. Gardy, and N. Markey. Weighted strategy logic with boolean goals over one-counter games. In *FSTTCS'15*, pages 69–83, 2015.

[Brihaye *et al.*, 2009] T. Brihaye, A. Da Costa Lopes, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In *LFCS'09*, pages 92–106, 2009.

[Bruyère *et al.*, 2013] V. Bruyère, E. Filiot, M. Randour, and J. F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *CoRR*,abs/1309.5439, 2013.

[Bulling and Jamroga, 2010] N. Bulling and W. Jamroga. Verifying agents with memory is harder than it seemed. *AI Communications*, 23:380–403, 2010.

[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 *CAV'14*, pages 525–532, 2014.

[Cermák *et al.*, 2015] P. Cermák, A. Lomuscio, and A. Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *AAAI'15*, pages 2038–2044, 2015.

[Cermák *et al.*, 2018] P. Cermák, A. Lomuscio, F. Mogavero, and A. Murano. Practical verification of multi-agent systems against SLK specifications. *I&C*, 261(Part 3):588–614, 2018.

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

[Chatterjee *et al.*, 2014] K. Chatterjee, L. Doyen, S. Nain, and M. Y. Vardi. The complexity of partial-observation stochastic parity games with finite-memory strategies. In *ICFSSCS'14*, pages 242–257, 2014.

[Cimatti *et al.*, 2002] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV2: An open-source tool for symbolic model checking. In *CAV'02*, pages 359–364, 2002.

[Gammie and van der Meyden, 2004] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *CAV'04*, pages 479–483, 2004.

[Gardy *et al.*, 2018] P. Gardy, P. Bouyer, and N. Markey. Dependences in strategy logic. In *STACS'18*, pages 34:1–34:15, 2018.

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

[Huang and van der Meyden, 2014] X. Huang and R. van der Meyden. Symbolic model checking epistemic strategy logic. In *AAAI'14*, pages 1426–1432, 2014.

[Jamroga and Bulling, 2011] W. Jamroga and N. Bulling. Comparing variants of strategic ability. In *IJCAI'11*, pages 252–257, 2011.

[Jamroga and Murano, 2014] W. Jamroga and A. Murano. On module checking and strategies. In *AAMAS'14*, pages 701–708, 2014.

[Jamroga and van der Hoek, 2004] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 62:1–35, 2004.

[Jamroga *et al.*, 2017a] W. Jamroga, M. Knapik, and D. Kurpiewski. Fixpoint approximation of strategic abilities under imperfect information. In *AAMAS'17*, pages 1241–1249, 2017.

[Jamroga *et al.*, 2017b] W. Jamroga, V. Malvone, and A. Murano. Reasoning about natural strategic ability. In *AAMAS'17*, pages 714–722, 2017.

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

[Katoen, 2008] C. Baier J. P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[Laroussinie and Markey, 2015] F. Laroussinie and N. Markey. Augmenting atl with strategy contexts. *I&C*, (245):98–123, 2015.

[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.

[Malvone *et al.*, 2018] V. Malvone, F. Mogavero, A. Murano, and L. Sorrentino. Reasoning about graded strategy quantifiers. *I&C*, 259:390 – 411, 2018.

[Mogavero *et al.*, 2012] F. Mogavero, A. Murano, G. Perelli, and M. Vardi. What makes ATL* decidable? a decidable fragment of strategy logic. In *CONCUR'12*, 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 Trans. Comput. Log.*, 15(4):34:1–34:47, 2014.

[Mogavero *et al.*, 2017] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: on the satisfiability problem. *Logical Methods in Computer Science*, 13(1), 2017.

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

[Tabatabaei *et al.*, 2016] M. Tabatabaei, W. Jamroga, and Peter Y. A. Ryan. Expressing receipt-freeness and coercion-resistance in logics of strategic ability: Preliminary attempt. In *PrAISe@ECAI 2016*, pages 1:1–1:8, 2016.

[Vester, 2013] S. Vester. Alternating-time temporal logic with finite-memory strategies. In *GandALF'13*, pages 194–207, 2013.