SAT-Based ATL Satisfiability Checking

Magdalena Kacprzak Bialystok University of Technology Bialystok, Poland m.kacprzak@pb.edu.pl

Artur Niewiadomski
Siedlce University, Faculty of Exact
and Natural Sciences
Siedlce, Poland
artur.niewiadomski@uph.edu.pl

Wojciech Penczek ICS PAS, Warsaw, Poland penczek@ipipan.waw.pl

ABSTRACT

Synthesis of models and strategies is a very important problem in software engineering. The main element here is checking the satisfiability of formulae expressing the specification of a system to be implemented. This paper puts forward a novel method for deciding the satisfiability of formulae of Alternating-time Temporal Logic (ATL). The method presented expands on one for CTL exploiting SAT Modulo Monotonic Theories solvers. Similarly to the CTL case, our approach appears to be very efficient. The experimental results show that we can quickly test the satisfiability of large ATL formulae that have been out of reach of the existing approaches.

KEYWORDS

ATL; SAT-Based Satisfiability; Monotonic Theory

1 INTRODUCTION

The problem of synthesis is a very important issue in the rapidly-growing field of artificial intelligence and modern software engineering [25, 27, 34]. The aim is to automatically develop highly innovative software, also for AI robots, chatbots or autonomous self-driving vehicles. The problem consists in finding a model satisfying a given property, provided the property is satisfiable. Finally, the model is transformed into its correct implementation.

A convenient formalism to specify the game-like interaction between processes in distributed systems is Alternating-Time Temporal Logic (ATL) [1, 10]. The interpretation of ATL formulae uses the paradigm of multi-agent systems and is defined in models like concurrent game structures or interpreted systems. This logic was introduced to reason about the strategic abilities of agents and their groups. The strategic modalities allow for expressing the ability of agents to force their preferences or to achieve a desired goal and are therefore suitable for describing properties like the existence of a winning strategy. This is particularly important when we study properties and verify the correctness of security protocols or voting systems. There are a lot of papers analysing different versions of ATL [6, 9, 11, 15–17, 22, 23, 35] and other modal logics of strategic ability [12, 30, 31]. However, there is still a need for developing and introducing new and innovative techniques for solving synthesis and satisfiability problems [7, 8, 19, 28, 32]. This is because these problems are hard and their solutions require searching for effective practical algorithms.

1.1 Contribution

In this paper we:

- introduce a novel technique for checking ATL satisfiability, applying for the first time SAT Modulo Monotonic Theories solvers.
- propose a method which is universal in the sense that it can be extended to different classes of multi-agent systems and ATL under different semantics,
- propose a method which allows for testing satisfiability in the class of models that meet given restrictions,
- present a new efficient tool for checking satisfiability of ATL.

1.2 Related Work

The complexity of the ATL satisfiability problem was proven to be EXPTIME-complete by van Drimmelen [20, 36] for a fixed number of agents, and by Walther et al. [37] for systems without this assumption. The satisfiability of ATL* was proved to be 2EXPTIME-complete [33]. A method for testing the satisfiability of ATL was developed by Goranko and Shkatov [21]. Subsequently, this method was extended for checking ATL* [14] and ATEL [5].

In this paper we propose a solution to the first stage of the synthesis problem, which consists in finding a model for a given ATL formula. For this purpose, we adopt the method based on SAT Modulo Monotonic Theories (SMMT) [26] used to search for models of the CTL formulae. This technique was introduced by Bayless et al. in [4] for building efficient lazy SMT solvers for Boolean monotonic theories. Next, Klenze et al. in [26] presented how the SMMT framework can be used to build an SMT solver for CTL model checking theory, and how to perform efficient and scalable CTL synthesis.

In this paper we go one step further by developing an SMMT solver for ATL formulae and show how to construct, often minimal, models for them. We compare the experimental results with the only implementation of the tool for testing ATL satisfiability described in the literature [14]. In that paper, unlike in our work, concurrent game structures were used as models for ATL* with perfect recall and perfect knowledge semantics.

The main advantage of our framework consists in the promising preliminary experimental results and the fact that we can test satisfiability in classes of models under given restrictions on the number of agents, their local states, transition functions, local protocols, and valuation of variables. Restrictions on the number of agents and their local states result directly from the finite model property for ATL [20]. In addition, it is possible to extend our approach to testing different classes of models and different types of strategies.

1.3 Outline

In Sec. 2 we define a multi-agent system and its model, and give the syntax and semantics of ATL. Sec. 3 defines Boolean monotonic Pre-print, arXiv.org, draft paper

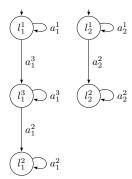


Figure 1: Visualization of an example MAS specification for $\mathcal{A} = \{1, 2\}$. On the left AG_1 , on the right AG_2 .

theory for ATL. In Sec. 4 the approximation algorithm is given and its properties are proved. Sec. 5 introduces the algorithm for deciding ATL satisfiability and model construction. Sec. 6 presents experimental results. Conclusions are in Sec. 7.

2 MAS AND ATL

Alur et al. introduced ATL logic taking into account different model compositions of open systems like turn-based, synchronous, asynchronous, with fairness constraints or Moore game structures. In this paper we follow Moore synchronous models [3], i.e., assume that the state space is the product of local state spaces, one for each agent, all agents proceed simultaneously, and each agent chooses its next local state independently of the moves of the other players. This is a restricted class of models, but it allows for the efficient testing of ATL satisfiability.

2.1 Multi-agent System

We start with defining a multi-agent system following [3, 24].

Definition 2.1. A multi-agent system (MAS) consists of n agents $\mathcal{A} = \{1, \ldots, n\}^1$, where each agent $i \in \mathcal{A}$ is associated with a 5-tuple $AG_i = (L_i, \iota_i, Act_i, P_i, T_i)$ including:

- a set of local states $L_i = \{l_i^1, l_i^2, \dots, l_i^{n_i}\};$
- an initial local state $\iota_i \in L_i$;
- a set of local actions $Act_i = \{\epsilon_i, a_i^1, a_i^2, \dots, a_i^{m_i}\};$
- a local protocol $P_i: L_i \to 2^{Act_i}$ which selects the actions available at each local state; we assume that $P_i(l_i) \neq \emptyset$ for every $l_i \in L_i$;
- a (partial) local transition function $T_i: L_i \times Act_i \to L_i$ such that $T_i(l_i, a)$ is defined iff $a \in P_i(l_i)$ and $T_i(l_i, \epsilon_i) = l_i$ whenever $\epsilon_i \in P_i(l_i)$ for each $l_i \in L_i$.

An example MAS specification is depicted in Figure 1, where $\mathcal{A} = \{1,2\}, AG_1 = \left(\{l_1^1, l_1^2, l_1^3\}, l_1^1, \{\epsilon_1, a_1^1, a_1^2, a_1^3\}, P_1, T_1\right),$ $P_1 = \left\{\left(l_1^1, \{a_1^1, a_1^3\}\right), \left(l_1^2, \{a_1^2\}\right), \left(l_1^3, \{a_1^2, a_1^3\}\right)\right\}, T_1 = \left\{\left((l_1^1, a_1^1), l_1^1\right), \left((l_1^1, a_1^3), l_1^3\right), \left((l_1^2, a_1^2), l_1^2\right), \left((l_1^3, a_1^2), l_1^2\right), \left((l_1^3, a_1^3), l_1^3\right)\right\},$ $AG_2 = \left\{\left(l_2^1, l_2^2\right), l_2^1, \{\epsilon_2, a_2^1, a_2^2\right\}, P_2, T_2\right), P_2 = \left\{\left(l_2^1, \{a_1^2, a_2^2\}\right), \left(l_2^2, \{a_2^2\}\right)\right\},$ $T_2 = \left\{\left((l_2^1, a_1^2), l_2^1\right), \left((l_1^1, a_2^2), l_2^2\right), \left((l_2^2, a_2^2), l_2^2\right)\right\}.$

Magdalena Kacprzak, Artur Niewiadomski, and Wojciech Penczek

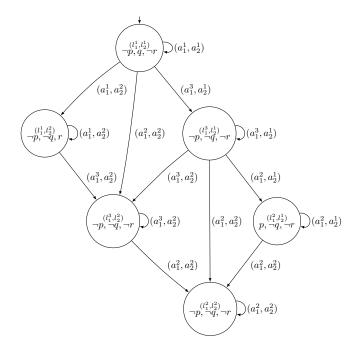


Figure 2: The model for MAS specification of Fig. 1 and $\mathcal{PV} = \{p, q, r\}$.

In our approach we consider *synchronous* multi-agent systems, i.e., systems in which each *global action* is a n-tuple (a^1, \ldots, a^n) , where $a^i \in Act_i$, i.e., each agent performs one local action. Define the set of all global actions as $Act = Act_1 \times \cdots \times Act_n$.

In order to describe the interaction between agents, the model for *MAS* is defined formally below.

Definition 2.2 (Model). Let \mathcal{PV} be a set of propositional variables and MAS be a multi-agent system with n agents. An (induced) model, is a 4-tuple $M = (St, \iota, T, V)$ with

- the set $St = L_1 \times \cdots \times L_n$ of the global *states*,
- an initial state $\iota = (\iota_1, \ldots, \iota_n) \in \mathcal{S}t$,
- the *global transition function* $T: St \times Act \rightarrow St$, such that $T(s_1, a) = s_2$ iff $T_i(s_1^i, a^i) = s_2^i$ for all $i \in \mathcal{A}$, where for global state $s = (l_1, \ldots, l_n)$ we denote the local component of agent i by $s^i = l_i$ and for a global action $a = (a^1, \ldots, a^n)$ we denote the local action of agent i by a^i ;
- a valuation of the propositional variables $V: \mathcal{S}t \to 2^{\mathcal{P}V}$.

We say that action $a \in Act$ is enabled at $s \in St$ if T(s, a) = s' for some $s' \in St$. We assume that at each $s \in St$ there exists at least one enabled action, i.e., for all $s \in St$ exist $a \in Act$, $s' \in St$, such that T(s, a) = s'. An infinite sequence of global states and actions $\pi = s_0 a_0 s_1 a_1 s_2 \ldots$ is called a path if $T(s_i, a_i) = s_{i+1}$ for every $i \ge 0$. Let $Act(\pi) = a_0 a_1 a_2 \ldots$ be the sequence of actions in π , and $\pi[i] = s_i$ be the i-th global state of π . $\Pi_M(s)$ denotes the set of all paths in M starting at s.

 $^{^1\}mathrm{The}$ environment component may be added here with no technical difficulty.

2.2 Alternating-time Temporal Logic

Alternating-time temporal logic, ATL [1–3] generalizes the branching-time temporal logic CTL [13] by replacing the path quantifiers E, A with *strategic modalities* $\langle\!\langle \Gamma \rangle\!\rangle$. Informally, $\langle\!\langle \Gamma \rangle\!\rangle\gamma$ expresses that the group of agents Γ has a collective strategy to enforce the temporal property γ . The formulae make use of temporal operators: "X" ("next"), "G" ("always from now on"), U ("strong until").

Definition 2.3 (Syntax of ATL). In vanilla ATL, every occurrence of a strategic modality is immediately followed by a temporal operator. Formally, the language of ATL is defined by the following grammar: $\varphi := p \mid \neg \varphi \mid \varphi \land \varphi \mid \langle\langle \Gamma \rangle\rangle X \varphi \mid \langle\langle \Gamma \rangle\rangle \varphi U \varphi \mid \langle\langle \Gamma \rangle\rangle G \varphi$.

Let M be a model. A *strategy* of agent $i \in \mathcal{A}$ in M is a conditional plan that specifies what i is going to do in any potential situation.

In this paper we focus on memoryless perfect information strategies. Formally, a memoryless perfect information strategy for agent i is a function $\sigma_i : \mathcal{S}t \to Act_i$ st. $\sigma_i(s) \in P_i(s^i)$ for each $s \in \mathcal{S}t$.

A *joint strategy* σ_{Γ} for a coalition $\Gamma \subseteq \mathcal{A}$ is a tuple of strategies, one per agent $i \in \Gamma$. We denote the set of Γ 's collective memoryless perfect information strategies by Σ_{Γ} .

Additionally, let $\sigma_{\Gamma} = (\sigma_1, \dots, \sigma_k)$ be a joint strategy for $\Gamma = \{i_1, \dots, i_k\}$. For each $s \in St$, we define $\sigma_{\Gamma}(s) := (\sigma_1(s), \dots, \sigma_k(s))$.

Definition 2.4 (Outcome paths). The outcome of strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ in state $s \in \mathcal{S}t$ is the set $out_{M}(s, \sigma_{\Gamma}) \subseteq \Pi_{M}(s)$ s. t. $\pi = s_{0}a_{0}s_{1}a_{1} \cdots \in out_{M}(s, \sigma_{\Gamma})$ iff $s_{0} = s$ and $\forall i \in \mathbb{N} \ \forall j \in \Gamma, \ a_{i}^{j} = \sigma_{j}(\pi[i])$.

Intuitively, the outcome of a joint strategy σ_{Γ} in a global state s is the set of all the infinite paths that can occur when in each state of the paths agents (an agent) in Γ execute(s) an action according to σ_{Γ} and agents (an agent) in $\mathcal{A} \setminus \Gamma$ execute(s) an action following their protocols.

The semantics of ATL is defined as follows:

```
M, s \models p \text{ iff } p \in V(s), \text{ for } p \in \mathcal{PV};
```

 $M, s \models \neg \varphi \text{ iff } M, s \not\models [Y]\varphi;$

 $M, s \models \varphi_1 \land \varphi_2 \text{ iff } M, s \models \varphi_1 \text{ and } M, s \models \varphi_2;$

 $M, s \models \langle \langle \Gamma \rangle \rangle X \varphi$ iff there is a strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ such that $out_{M}(s, \sigma_{\Gamma}) \neq \emptyset$ and, for each path $\pi \in out_{M}(s, \sigma_{\Gamma})$, we have $M, \pi \models X \varphi$, i.e., $M, \pi[1] \models \varphi$;

 $M, s \models \langle \langle \Gamma \rangle \rangle \varphi_1 U \varphi_2$ iff there is a strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ such that $out_M(s, \sigma_{\Gamma}) \neq \emptyset$ and, for each path $\pi \in out_M(s, \sigma_{\Gamma})$, we have $M, \pi \models \varphi_1 U \varphi_2$, i.e., $M, \pi[i] \models \varphi_2$ for some $i \geq 0$ and $M, \pi[j] \models \varphi_1$ for all $0 \leq j < i$;

 $M, s \models \langle \langle \Gamma \rangle \rangle G \varphi$ iff there is a strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ such that $out_{M}(s, \sigma_{\Gamma}) \neq \emptyset$ and, for each path $\pi \in out_{M}(s, \sigma_{\Gamma})$, we have $M, \pi \models G \varphi$, i.e., $M, \pi[i] \models \varphi$, for every $i \geq 0$.

We omit the M symbol if it is clear which model is intended.

Definition 2.5. **(Validity)** An ATL formula φ is valid in M (denoted $M \models \varphi$) iff $M, \iota \models \varphi$, i.e., φ is true at the initial state of the model M.

An example ATL formula, which is satisfied by the model depicted in Figure 2, is as follows: $\langle\langle 1,2\rangle\rangle F(p\wedge \neg q\wedge \neg r)\wedge\langle\langle 1\rangle\rangle F(\neg p\wedge q\wedge \neg r)\wedge\langle\langle 1,2\rangle\rangle X(\neg p\wedge \neg q\wedge r)$, where $\langle\langle \Gamma\rangle\rangle F\alpha$ is a short for $\langle\langle \Gamma\rangle\rangle (trueU\alpha)$.

3 BOOLEAN MONOTONIC THEORY FOR ATL

In this section we show how to construct a Boolean monotonic theory for ATL, which allows for building a lazy SMT solver [4] for ATL. The resulting tool, a SAT modulo ATL solver, can be used for testing the satisfiability of the ATL formulae as well as for performing efficient and scalable synthesis.

3.1 Boolean Monotonic Theory

Consider a predicate $P: \{0,1\}^n \mapsto \{0,1\}$. We say that P is Boolean positive monotonic iff $P(s_1,\ldots,s_{i-1},0,s_{i+1},\ldots,s_n)=1$ implies $P(s_1,\ldots,s_{i-1},1,s_{i+1},\ldots,s_n)=1$, for all $1 \le i \le n$. P is called Boolean negative monotonic iff $P(s_1,\ldots,s_{i-1},1,s_{i+1},\ldots,s_n)=1$ implies $P(s_1,\ldots,s_{i-1},0,s_{i+1},\ldots,s_n)=1$, for all $1 \le i \le n$.

The definition of (positive and negative Boolean) monotonicity for a function $F: \{0,1\}^n \mapsto 2^S$ (for some set S) is analogous. F is Boolean *positive monotonic* iff $F(s_1,\ldots,s_{i-1},0,s_{i+1},\ldots,s_n) \subseteq F(s_1,\ldots,s_{i-1},1,s_{i+1},\ldots,s_n)$, for all $1 \le i \le n$. A function F is Boolean *negative monotonic* iff $F(s_1,\ldots,s_{i-1},1,s_{i+1},\ldots,s_n) \subseteq F(s_1,\ldots,s_{i-1},0,s_{i+1},\ldots,s_n)$, for all $1 \le i \le n$. In what follows we refer to Boolean monotonicity simply as to monotonicity.

Definition 3.1 (Boolean Monotonic Theory). A theory T with a signature $\Omega = (S, S_f, S_r, ar)$, where S is a non-empty set of elements called sorts or types, S_f is a set of function symbols, S_r is a set of relation symbols, and ar is arity of the relation and function symbols, is (Boolean) monotonic iff:

- (1) the only sort in Ω is Boolean;
- (2) all predicates and functions in Ω are monotonic.

The authors of [4] introduced techniques for building an efficient SMT solver for Boolean monotonic theories (SMMT). These techniques were further used for checking satisfiability of CTL [26]. In this paper, we extend this approach to ATL. We start with showing a Boolean encoding of the ATL models.

3.2 Boolean Encoding of ATL Models

First, we make some assumptions about MAS. Assume that we are given a set of agents $\mathcal{A}=\{1,\ldots,n\}$, where each agent $i\in\mathcal{A}$ has a fixed set of the local states $L_i=\{l_1^1,\ldots,l_i^{n_i}\}$ and a fixed initial local state $l_i\in L_i$. Since agent i can be in one of its n_i local states, and a local transition function T_i is restricted such that it does not involve actions of the other agents, we can assume, without a loss of generality, that agent i has exactly n_i possible actions, i.e., from each local state it can potentially move to each of its local states. So, assume that the set of local actions for agent i is $Act_i=\{a_i^1,\ldots,a_i^{n_i}\}$ and an action a_i^j can move the agent i from any local state to local state l_i^j . Moreover, we assume that each local protocol P_i satisfies that at least one action is available at each local state. Consequently, the local transition function T_i for agent i is defined as follows: $T_i(l_i^k,a_i^j)=l_i^j$ if $a_i^j\in P_i(l_i^k)$, for any $l_i^k\in L_i$ and $1\leq j\leq n_i$.

Next, we represent every single agent i with a given $AG_i = (L_i, \iota_i, Act_i, P_i, T_i)$ by means of a bit vector. In fact, under the condition that the number of the local states is fixed, the initial state is selected, and the rules for defining the local actions and a local transition function are given, we have to encode a local protocol P_i . It can be defined by a Boolean table lp_i of $|L_i| \times |Act_i|$ entries,

where 0 at position (l_i^k, a_i^j) means that the local action a_i^j is not available at the local state l_i^k , and 1 stands for the availability. This table can be represented by a bit vector $tb_i = (lp_i[1], \ldots, lp_i[n_i])^2$, where $lp_i[j]$ stands for the j-th row of the table lp_i , encoding which local actions are available at which local states.

Since the model $M = (St, \iota, T, V)$ induced by a MAS is a product of AG_i for $i \in \mathcal{A}$, the bit vector (tb_1, \ldots, tb_n) determines the synchronous product of the local transition functions of the agents and thus the global transition function T of M.

Finally, we need to define a valuation of the propositional variables. Given a set \mathcal{PV} , a Boolean table of size $|\mathcal{S}t| \times |\mathcal{PV}|$ saves which propositional variables are true in which global states. Then, let $vb = (vb_1, \ldots, vb_k)$ be a bit vector, where $k = |\mathcal{S}t| \cdot |\mathcal{PV}|$, controlling which propositional variables hold in each global state.

In this way, every model can be represented with a bit vector. For a fixed number $|\mathcal{PV}|$ of the propositional variables, a fixed number n of agents, a fixed number n_i of the local states of agent i, for every $i=1,\ldots,n$, the bit vector $v_M=(tb_1,\ldots,tb_n,vb)$ encodes some model induced by MAS without an initial state fixed. Therefore, v_M actually encodes a family of models which differ only in the initial state.

3.3 Predicate Model

From now on, we consider models M defined over the fixed number $|\mathcal{PV}|$ of the propositional variables and a fixed number n of agents with fixed numbers $|L_1|,\ldots,|L_n|$ of local states. Thus, we consider models that can be represented by a bit vector v_M consisting of exactly $n_M = |L_1|^2 + \ldots + |L_n|^2 + |L_1| \cdot \ldots \cdot |L_n| \cdot |\mathcal{PV}|$ bits. In the rest of the work we will use the following notation:

$$V_m = (TB_1, \ldots, TB_n, VB)$$

to denote a vector of Boolean variables, where for $i=1,\ldots,n,TB_i$ is a vector of $|L_i|^2$ variables and VB is a vector of $|L_1|\cdot\ldots\cdot|L_n|\cdot|\mathcal{P}V|$ variables.

For an ATL formula ϕ defined over propositional variables of \mathcal{PV} and over agents of \mathcal{A} , for each global state $g \in \mathcal{S}t$ the following predicate is defined: $Model_{g,\phi}(V_m)$. For the bit vector v_M encoding a model M we define: $Model_{g,\phi}(v_M)=1$ if and only if $M,g\models\phi$. Unfortunately, it turns out that this predicate is not monotonic, i.e. there is an ATL formula ϕ and a global state g for which the predicate $Model_{g,\phi}(V_m)$ is not monotonic w.r.t. V_m .

Theorem 3.2. The predicate $Model_{g,\phi}(V_m)$ is neither positive nor negative monotonic w.r.t V_m .

Proof. Since ATL subsumes CTL, the thesis follows from the similar result for CTL [26]. $\hfill\Box$

However, in some special cases, as we show below, the predicate $Model_{a,\phi}(V_m)$ can be monotonic.

Theorem 3.3. The predicate $Model_{g,\,\phi}(V_m)$ is positive monotonic w.r.t. VB if $\phi \in \{p,p \land q, \langle\!\langle \Gamma \rangle\!\rangle X p, \langle\!\langle \Gamma \rangle\!\rangle Gp, \langle\!\langle \Gamma \rangle\!\rangle pUq\}$, where $p,q \in \mathcal{PV}, \Gamma \subseteq \mathcal{A}$.

PROOF. Let $\phi \in \{p, p \land q, \langle (\Gamma) \rangle X p, \langle (\Gamma) \rangle Gp, \langle (\Gamma) \rangle pUq \}$, where $p, q \in \mathcal{PV}$ and let v_M be a bit vector such that $Model_{g,\phi}(v_M) = 1$. This means that $M, g \models \phi$, for M encoded by v_M , where g is an initial state of M.

Now, let $v_{M'}$ be a vector which differs from v_M only in one value vb_j , for $j=1,\ldots,k$, which is 0 in v_M and 1 in $v_{M'}$. The model M', encoded by $v_{M'}$, has the same states, transitions, and state properties as M, except for one state property which holds in M' but not in M, i.e., one propositional variable holds true in some state t in M' but does not hold in t in M. Thus, if $\phi \in \{p, p \land q\}$ and $M, q \models \phi$, then $M', q \models \phi$ as well.

Consider the case of $\phi \in \{\langle \Gamma \rangle X p, \langle \Gamma \rangle G p, \langle \Gamma \rangle p U q \}$. Since M is a model of ϕ , then there is a strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ such that for each path $\pi \in out_M(s, \sigma_{\Gamma}), \pi \models \psi$ for $\psi \in \{X p, G p, p U q\}$. Clearly, there is the same strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ in M'. Consider a path $\pi' \in out_{M'}(s, \sigma_{\Gamma})$. This path differs from the corresponding path $\pi \in out_M(s, \sigma_{\Gamma})$ such that it may contain more states where p or q holds. Therefore, $\pi' \models \psi$ for $\psi \in \{X p, G p, p U q\}$. So, we have $Model_{q,\phi}(v_{M'}) = 1$.

Theorem 3.4. The predicate $Model_{g,\neg p}(V_m)$ is negative monotonic w.r.t. VB, for $p \in \mathcal{PV}$.

PROOF. Let v_M be a bit vector such that $Model_{g,\neg p}(v_M)=1$. This means that $M,g\models\neg p$ for M, encoded by v_M , with the initial state g. Now, let $v_{M'}$ be a bit vector which differs from v_M only in one value vb_j , for $j=1,\ldots,k$, which is 1 in v_M and 0 in $v_{M'}$. The model M', encoded by $v_{M'}$, has the same states, transitions, and state properties as M, except for one state property which does not hold in M' but holds in M, i.e., one propositional variables is false in some state t in M' but is true in t in M. Thus if $M,g\models\neg p$, then $M',g\models\neg p$ and finally $Model_{g,\neg p}(v_{M'})=1$.

Theorem 3.5. The predicate $Model_{g,\phi}(V_m)$ is both positive and negative monotonic w.r.t. TB_i for each $i \in \mathcal{A}$ if $\phi \in \{p, \neg p, p \land q\}$, where $p, q \in \mathcal{PV}$.

PROOF. Notice that adding or removing transitions (both local or global) does not alter the truthfulness of the formula $\phi \in \{p, \neg p, p \land q\}$ as long as p and q are propositional variables. Therefore, $Model_{g,\phi}(V_m)$ for $\phi \in \{p, \neg p, p \land q\}$ is both positive and negative monotonic w.r.t. TB_i for each $i \in \mathcal{A}$.

Theorem 3.6. The predicate $Model_{g,\phi}(V_m)$ is positive monotonic w.r.t. TB_i for $i \in \Gamma$ if $\phi \in \{\langle\langle \Gamma \rangle\rangle\rangle X_p, \langle\langle \Gamma \rangle\rangle\rangle G_p, \langle\langle \Gamma \rangle\rangle\rangle pUq\}$, where $p, q \in \mathcal{PV}$, $\Gamma \subseteq \mathcal{A}$.

PROOF. Let $\phi \in \{\langle \Gamma \rangle X p, \langle \Gamma \rangle G p, \langle \Gamma \rangle p U q \}$, where $p, q \in \mathcal{P}V$ and let v_M be a bit vector such that $Model_{g,\phi}(v_M) = 1$. This means that $M, g \models \phi$ for M, encoded by v_M , with the initial state g. Now, let $v_{M'}$ be a vector which differs from v_M only in one value tb_i^j , for some $i \in \Gamma$ and $j \in \{1, \ldots, (n_i)^2\}$, which is 0 in v_M and 1 in $v_{M'}$. The model M', encoded by $v_{M'}$, has the same states, state properties, and local transitions of the agents, except for one local transition of one agent from Γ that is enabled in M' but not in M.

If $Model_{g,\phi}(v_M)=1$, then there is a strategy $\sigma_{\Gamma}\in \Sigma_{\Gamma}$ such that for each path $\pi\in out_M(s,\sigma_{\Gamma}),\,\pi\models\psi$ for $\psi\in\{X\,p,Gp,pUq\}$.

Observe that adding one local transition to one agent of Γ results in more strategies of the agents of Γ , but at the same time the

²In what follows, we assume that a sequence of bit vectors is identified with the bit vector composed of its elements.

existing strategies are still in place. Therefore, the strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ is in M' as well. Therefore, $Model_{a,\phi}(v_{M'}) = 1$.

Theorem 3.7. The predicate $Model_{g,\phi}(V_m)$ is negative monotonic w.r.t. TB_i for $i \in \mathcal{A} \setminus \Gamma$ if $\phi \in \{\langle\langle \Gamma \rangle\rangle Xp, \langle\langle \Gamma \rangle\rangle Gp, \langle\langle \Gamma \rangle\rangle pUq\}$, where $p, q \in \mathcal{PV}, \Gamma \subseteq \mathcal{A}$.

PROOF. Let $\phi \in \{\langle\langle \Gamma \rangle\rangle X p, \langle\langle \Gamma \rangle\rangle G p, \langle\langle \Gamma \rangle\rangle p U q\}$, where $p, q \in \mathcal{PV}$, $\Gamma \subseteq \mathcal{A}$, and v_M be a bit vector s.t. $Model_{g,\phi}(v_M) = 1$. This means that $M, g \models \phi$ for M, encoded by v_M , with the initial state g.

Now, let $v_{M'}$ be a bit vector which differs from v_M only in one value tb_i^j , for some $i \in \mathcal{A} \setminus \Gamma$ and $j = 1, \dots, (n_i)^2$, which is 1 in v_M and 0 in $v_{M'}$. The model M', encoded by $v_{M'}$, has the same states, state properties, and local transitions of the agents, except for one local transition of one agent of $\mathcal{A} \setminus \Gamma$ that is enabled in M but not in M'. Observe that deleting one local transition of some agent of $\mathcal{A} \setminus \Gamma$ results in the same number of strategies of the agents of Γ , but for each strategy the number of paths in its outcome may be lower. The protocol function ensures that at least one action and thereby at least one transition must remain (not all can be deleted). Thus, for any strategy of the agents of Γ , the number of transitions consistent with this strategy cannot be reduced to zero. If $Model_{a,\phi}(v_M) = 1$, then there is a strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ such that for each path $\pi \in out_M(g, \sigma_{\Gamma}), \pi \models \psi$ for $\psi \in \{Xp, Gp, pUq\}.$ Therefore, the strategy $\sigma_{\Gamma} \in \Sigma_{\Gamma}$ is in M'. Since $\emptyset \neq out_{M'}(q, \sigma_{\Gamma}) \subseteq$ $out_M(g, \sigma_{\Gamma})$, we have $Model_{g, \phi}(v_{M'}) = 1$.

3.4 Function solve

In order to compute the value of the predicate $Model_{g,\phi}(v_M)$ for a given M, we define a new function, called $solve_{\phi}(V_m)$. This function returns a set of states of M such that $g \in solve_{\phi}(v_M)$ iff $Model_{g,\phi}(v_M) = 1$, i.e., $M,g \models \phi$. The monotonicity properties also apply to the function $solve_{\phi}$, as every state returned by this function can be viewed as an initial state of the model M. Thus, the theorem below follows directly from Theorems 3.3 – 3.7.

Theorem 3.8. The function $solve_{\phi}(V_m)$ is

- positive monotonic w.r.t. VB for $\phi \in \{p, p \land q, \langle\langle \Gamma \rangle\rangle X p, \langle\langle \Gamma \rangle\rangle Gp, \langle\langle \Gamma \rangle\rangle pUq\}$,
- negative monotonic w.r.t. VB for $\phi = \neg p$,
- positive and negative monotonic w.r.t. TB_i for $i \in \mathcal{A}$ if $\phi \in \{p, \neg p, p \land q\}$,
- positive monotonic w.r.t. TB_i for each $i \in \Gamma$ if $\phi \in \{\langle\langle \Gamma \rangle\rangle\rangle X_p$, $\langle\langle \Gamma \rangle\rangle G_p$, $\langle\langle \Gamma \rangle\rangle pUq\}$,
- negative monotonic w.r.t. TB_i for $i \in \mathcal{A} \setminus \Gamma$ if $\phi \in \{\langle\langle \Gamma \rangle\rangle\rangle X p$, $\langle\langle \Gamma \rangle\rangle\rangle Gp$, $\langle\langle \Gamma \rangle\rangle\rangle pUq$ }, where $p, q \in \mathcal{PV}$, $\Gamma \subseteq \mathcal{A}$.

Moreover, to compute $solve_{\phi}(V_m)$ for each ATL formula ϕ , a new evaluation function $solve_{op}(Y_1,V_m)$ is defined for an unary operator op and $solve_{op}(Y_1,Y_2,V_m)$ for a binary operator op, and $Y_1,Y_2\subseteq St$. This function evaluates the operator op on sets of states Y_1,Y_2 instead of the formulae holding in these states. If $\phi=p\in \mathcal{PV}$, then for a given model M, $solve_p(v_M)$ returns the set of states of M in which p holds. Otherwise, $solve_{\phi}(v_M)$ takes the top-most operator op of ϕ and solves its argument(s) recursively using the function $solve_{op}$ and applying $solve_{op}(Y_1,v_M)$ ($solve_{op}(Y_1,Y_2,v_M)$) to the returned set(s) of states.

Now, Theorem 3.8 can be rewritten by replacing propositional variables p and q by sets of states satisfying these variables.

Theorem 3.9. The function $solve_{op}(Y_1, V_m)$ for an unary operator op and $solve_{op}(Y_1, Y_2, V_m)$ for a binary operator op is

- positive monotonic w.r.t. VB for $op \in \{\land, \langle\langle \Gamma \rangle\rangle X, \langle\langle \Gamma \rangle\rangle G, \langle\langle \Gamma \rangle\rangle U\}$,
- negative monotonic w.r.t. VB for $op = \neg$,
- positive and negative monotonic w.r.t. TB_i for $i \in \mathcal{A}$ and $op \in \{\neg, \land\}$,
- positive monotonic w.r.t. TB_i for $i \in \Gamma$ and $op \in \{\langle\langle \Gamma \rangle\rangle\rangle X$, $\langle\langle \Gamma \rangle\rangle G$, $\langle\langle \Gamma \rangle\rangle U$,
- negative monotonic w.r.t. TB_i for $i \in \mathcal{A} \setminus \Gamma$ and $op \in \{\langle\langle \Gamma \rangle\rangle X$, $\langle\langle \Gamma \rangle\rangle G$, $\langle\langle \Gamma \rangle\rangle U$.

To compute $solve_{\phi}(v_M)$, $solve_{op}(Y_1, v_M)$, and $solve_{op}(Y_1, Y_2, v_M)$ the model checking algorithms described in [29] are applied.

4 APPROXIMATING ATL MODELS

In this section we show how to approximate models for ATL in order to solve the satisfiability problem using SAT modulo monotonic theories. First, the construction of over and under approximations of a model are given. Then, the approximation algorithm is defined together with the proofs of its properties.

4.1 Construction of M_{over} and M_{under}

Given a set of agents $\mathcal{A} = \{1, \dots, n\}$, we fix for each $i \in \mathcal{A}$ a set of local states L_i , an initial state ι_i , and a set of local actions defined like in Def. 2.1. Next we define a function, called a *partial protocol*:

$$CP_i: L_i \times Act_i \rightarrow \{0, 1, undef\}.$$

By a partial MAS, denoted MAS_{CP} , we mean a MAS in which each agent is associated with a partial protocol rather than with a protocol. Then, a model induced by MAS_{CP} together with a partial valuation of the propositional variables

$$CV: St \times PV \rightarrow \{0, 1, undef\}$$

is called a *partial model*, denoted by M_{par} . Both a partial protocol and a partial valuation can be extended to total functions. The intention behind these definitions is to give requirements on the models.

For each partial model, total models $M_{under}^{\mathcal{A}}$ and M_{over}^{Γ} , for $\Gamma \subseteq \mathcal{A}$, are constructed. First, for every agent $i \in \mathcal{A}$ we define: a necessary local protocol $\underline{P_i}: L_i \to 2^{Act_i}$ and a possible local protocol $\overline{P_i}: L_i \to 2^{Act_i}$, where:

- (1) if $CP_i(l_i, a_i) = 1$ then $a_i \in \underline{P_i}(l_i)$ and $a_i \in \overline{P_i}(l_i)$,
- (2) if $CP_i(l_i, a_i) = 0$ then $a_i \notin P_i(l_i)$ and $a_i \notin \overline{P_i}(l_i)$,
- (3) if $CP_i(l_i, a_i) = undef$ then $a_i \notin P_i(l_i)$ and $a_i \in \overline{P_i}(l_i)$.

Notice that the possible local protocol is an extension of the necessary local protocol, i.e., the following condition holds: for every local state l_i , $\underline{P_i}(l_i) \subseteq \overline{P_i}(l_i)$. In a similar way, total valuations of the propositional variables are defined: a necessary valuation $\underline{V}: \mathcal{S}t \to 2^{\mathcal{P}V}$ and a possible valuation $\overline{V}: \mathcal{S}t \to 2^{\mathcal{P}V}$ such that:

- (1) if CV(g, p) = 1 then $p \in \underline{V}(g)$ and $p \in \overline{V}(g)$,
- (2) if CV(g, p) = 0 then $p \notin \underline{V}(g)$ and $p \notin \overline{V}(g)$,
- (3) if CV(g, p) = undef then $p \notin \underline{V}(g)$ and $p \in \overline{V}(g)$.

Observe that for every global state $g \in St$ we have $\underline{V}(g) \subseteq \overline{V}(g)$.

The model $M^{\mathcal{A}}_{under}$ is defined as in Def. 2.2 of all agents $i \in \mathcal{A}$ with $AG_i = (L_i, \iota_i, Act_i, \underline{P_i}, T_i)$ and for the valuation of the propositional variables \underline{V} . The model M^{Γ}_{over} is defined as in Def. 2.2 of agents $i \in \Gamma$ with $AG_i = (L_i, \iota_i, Act_i, \overline{P_i}, T_i)$, agents $j \in \mathcal{A} \setminus \Gamma$ with $AG_j = (L_j, \iota_j, Act_j, \underline{P_j}, T_j)$, and for the valuation of the propositional variables \overline{V} .

4.2 Algorithm SApp

We say that the model $M = (St, \iota, T, V)$ induced by agents $\mathcal{A} = \{1, \ldots, n\}$ with $AG_i = (L_i, \iota_i, Act_i, P_i, T_i)$ for $i \in \mathcal{A}$ and the propositional variables $\mathcal{P}V$ is *compatible* with a partial model M_{par} induced by the same sets of agents and propositional variables, and determined by the given partial protocols CP_i for $i \in \mathcal{A}$, and a partial valuation CV if P_i is consistent with CP_i for every $i \in \mathcal{A}$, and V satisfies all conditions determined by CV. Formally:

```
(1) if CP<sub>i</sub>(l<sub>i</sub>, a<sub>i</sub>) = 1 then a<sub>i</sub> ∈ P<sub>i</sub>(l<sub>i</sub>),
(2) if CP<sub>i</sub>(l<sub>i</sub>, a<sub>i</sub>) = 0 then a<sub>i</sub> ∉ P<sub>i</sub>(l<sub>i</sub>),
(1) if CV(g, p) = 1 then p ∈ V(g),
(2) if CV(g, p) = 0 then p ∉ V(g).
```

Observe that $M_{under}^{\mathcal{A}}$ and M_{over}^{Γ} are compatible with M_{par} . What is more, for any model M compatible with M_{par} we have:

$$\forall i \in \mathcal{A} \quad P_i(l_i) \subseteq P_i(l_i) \subseteq \overline{P_i}(l_i), \quad \underline{V}(g) \subseteq V(g) \subseteq \overline{V}(g).$$

Theorem 4.1. Let v_M , $v_{M^{\mathcal{A}}_{under}}$, $v_{M^{\Gamma}_{over}}$, for some $\Gamma \subseteq \mathcal{A}$, be bit vectors encoding models M, $M^{\mathcal{A}}_{under}$, and M^{Γ}_{over} , respectively, then we have:

```
 \begin{split} \bullet \ v_{M_{under}^{\mathcal{A}}} & [tb_{i}[j_{i}]] \leq v_{M}[tb_{i}[j_{i}]] \leq v_{M_{over}^{\Gamma}}[tb_{i}[j_{i}]] \\ & \text{for all } i \in \Gamma \text{ and for all } 1 \leq j_{i} \leq n_{i}, \text{ for short} \\ & v_{M_{under}^{\mathcal{A}}} & [TB_{i}] \leq v_{M}[TB_{i}] \leq v_{M_{over}^{\Gamma}}[TB_{i}] \text{ for } i \in \Gamma, \text{ and} \\ \bullet \ v_{M_{under}^{\mathcal{A}}} & [vb_{j}] \leq v_{M}[vb_{j}] \leq v_{M_{over}^{\Gamma}}[vb_{j}] \text{ for all } 1 \leq j \leq k, \\ & \text{for short } v_{M^{\mathcal{A}}} & [VB] \leq v_{M}[VB] \leq v_{M_{over}^{\Gamma}}[VB]. \end{split}
```

Proof. Follows from the definitions of $M_{under}^{\mathcal{A}}$ and M_{over}^{Γ} . \square

This means that each transition in $M^{\mathcal{A}}_{under}$ is also a transition in M, and each transition in M is a transition in M^{Γ}_{over} . Similarly for the propositional variables, if some propositional variable holds true at state g of $M^{\mathcal{A}}_{under}$, then it also holds true at the same state of M, and if some propositional variable holds true at g of M, then it also holds true at the same state of M^{Γ}_{over} .

Now, a new function $SApp_{M_{par}}(\phi,V_{m_1},V_{m_2})$ over two separate assignments of transitions and states $V_{m_1}=(TB_1^1,\ldots,TB_n^1,VB^1)$ and $V_{m_2}=(TB_1^2,\ldots,TB_n^2,VB^2)$ is defined. For a given partial model M_{par} and two models M_1 and M_2 compatible with M_{par} , the output of the function is determined by the following algorithm.

```
Algorithm SApp_{M_{par}}(\phi, v_{M_1}, v_{M_2})

1: if \phi \in \mathcal{PV} then

2: return \{g \in St : M_1, g \models \phi\}

3: else if \phi = op(\psi) then

4: if op is \neg then // negative monotonic

5: Y := SApp_{M_{par}}(\psi, v_{M_2}, v_{M_1})

6: return solve_{op}(Y, v_{M_2})

7: else // op \in \{\langle \Gamma \rangle \rangle X, \langle \langle \Gamma \rangle \rangle G\}
```

```
8:
                   Y := SApp_{M_{par}}(\psi, v_{M_1}, v_{M_2})
                    \begin{aligned} \textbf{if} \ v_{M_1} &= v_{M_{under}}^{\mathcal{A}} \ \textbf{then} \\ & \textbf{return} \ solve_{op}(Y, v_{M_1}) \end{aligned} 
9:
10:
11:
                   else return solve_{op}(Y, v_{M_{over}^{\Gamma}})
12: else if \phi \in \{\langle\langle \Gamma \rangle\rangle \psi_1 U \psi_2, \psi_1 \wedge \psi_2\}
13:
                Y_1 := SApp_{M_{par}}(\psi_1, v_{M_1}, v_{M_2})
14:
                Y_2 := SApp_{M_{par}}(\psi_2, v_{M_1}, v_{M_2})
15:
                if \phi is \langle\langle \Gamma \rangle\rangle \psi_1 U \psi_2 then
16:
                       if v_{M_1} = v_{M_{under}}^{\mathcal{A}} then
17:
                               return solve_{op}(Y_1, Y_2, v_{M_1})
                       else return solve_{op}(Y_1, Y_2, v_{M_{over}^{\Gamma}})
18:
19:
                       return solve_{op}(Y_1, Y_2, v_{M_1})
20:
```

Theorem 4.2. The function $SApp_{M_{par}}(\phi, V_{m_1}, V_{m_2})$ is

- positive monotonic w.r.t. TB¹_i for i ∈ A and VB¹ and negative monotonic w.r.t. TB²_i for i ∈ A and VB² for φ ∈ {p, ¬p, p ∧ q}, and
- positive monotonic w.r.t. TB_i^1 for $i \in \Gamma$ and VB^1 and negative monotonic w.r.t. TB_i^2 for $i \in \Gamma$ and VB^2 for $\phi \in \{\langle\langle \Gamma \rangle\rangle\rangle Xp$, $\langle\langle \Gamma \rangle\rangle\rangle Gp$, $\langle\langle \Gamma \rangle\rangle\rangle DUq\}\}$.

PROOF. By a structural induction on a formula ϕ . Let M_{11} , M_{12} , M_{21} , M_{22} be models compatible with M_{par} such that $v_{M_{11}}[VB] \leq v_{M_{12}}[VB]$, $v_{M_{11}}[TB_i] \leq v_{M_{12}}[TB_i]$ for $i \in \mathcal{A}$, and $v_{M_{22}}[VB] \leq v_{M_{21}}[VB]$, $v_{M_{22}}[TB_i] \leq v_{M_{21}}[TB_i]$ for $i \in \mathcal{A}$.

The base case. If $\phi = p \in \mathcal{PV}$, then from the definition of the algorithm, $SApp_{M_{par}}(p,v_{M_{11}},v_{M_{21}})$ returns the set of the states satisfying p in M_{11} and $SApp_{M_{par}}(p,v_{M_{12}},v_{M_{21}})$ returns the set of the states satisfying p in M_{12} . Since $v_{M_{11}}[VB] \leq v_{M_{12}}[VB]$ then $SApp_{M_{par}}(p,v_{M_{11}},v_{M_{21}}) \subseteq SApp_{M_{par}}(p,v_{M_{12}},v_{M_{21}})$ and $SApp_{M_{par}}(\phi,V_{m_1},V_{m_2})$ is positive monotonic w.r.t. VB^1 . Observe that the output of $SApp_{M_{par}}(\phi,V_{m_1},V_{m_2})$ depends only on values of variables VB^1 , thus the function is also positive monotonic w.r.t. TB_i^1 for $i \in \mathcal{A}$ and negative monotonic w.r.t. VB^2 and TB_i^2 for $i \in \mathcal{A}$

The induction step. We show the proof for the unary operators \neg , $\langle\!\langle \Gamma \rangle\!\rangle X$, $\langle\!\langle \Gamma \rangle\!\rangle G$. The proofs for the binary operators Until and \land are similar.

Induction assumption (IA): the thesis holds for a formula ψ . Induction hypothesis (IH): the thesis holds for $\phi = op \ \psi$.

• If $\phi = \neg \psi$.

If $Y = SApp_{M_{par}}(\psi, v_{M_{21}}, v_{M_{11}})$ and $Y' = SApp_{M_{par}}(\psi, v_{M_{21}}, v_{M_{12}})$, then $Y' \subseteq Y$ since $SApp_{M_{par}}$ is negative monotonic w.r.t. VB^2 and TB_i^2 for $i \in \mathcal{A}$, from IA. Next, $solve_{\neg}(Y, v_{M_{21}}) \subseteq solve_{\neg}(Y', v_{M_{21}})$ since $solve_{\neg}$ returns the compliment of Y and Y', respectively. Thus, $SApp_{M_{par}}(\phi, v_{M_{11}}, v_{M_{21}}) \subseteq SApp_{M_{par}}(\phi, v_{M_{12}}, v_{M_{21}})$ and the function is positive monotonic w.r.t. VB^1 and TB_i^1 for $i \in \mathcal{A}$. If $Y = SApp_{M_{par}}(\psi, v_{M_{21}}, v_{M_{11}})$ and $Y' = SApp_{M_{par}}(\psi, v_{M_{22}}, v_{M_{11}})$, then $Y' \subseteq Y$ since $SApp_{M_{par}}$ is positive monotonic w.r.t. VB^1 and TB_i^1 for $i \in \mathcal{A}$, from IA. Next, $solve_{\neg}(Y, v_{M_{21}}) \subseteq solve_{\neg}(Y', v_{M_{21}})$ since $solve_{\neg}$ returns the compliment of Y and Y', respectively,

and $solve_{\neg}(Y', v_{M_{21}}) \subseteq solve_{\neg}(Y', v_{M_{22}})$ since $solve_{\neg}$ is negative

monotonic w.r.t. VB and TB_i for $i \in \mathcal{A}$. Thus, $solve_{\neg}(Y, v_{M_{21}})$

 $\subseteq solve_{\neg}(Y',v_{M_{22}})$, i.e., $SApp_{M_{par}}(\phi,v_{M_{11}},v_{M_{21}}) \subseteq SApp_{M_{par}}(\phi,v_{M_{11}},v_{M_{22}})$ and the function is negative monotonic w.r.t. VB^2 and TB_i^2 for $i \in \mathcal{A}$.

• If $\phi = op \ \psi$ with $op \in \langle\langle \Gamma \rangle\rangle X$, $\langle\langle \Gamma \rangle\rangle G$.

Now, let M_{11} , M_{12} , M_{21} , M_{22} be models such that $v_{M_{11}}[TB_i] \leq v_{M_{12}}[TB_i]$ and $v_{M_{22}}[TB_i] \leq v_{M_{21}}[TB_i]$ for $i \in \Gamma \subseteq \mathcal{A}$. The other restrictions remain the same. If $Y = SApp_{M_{par}}(\psi, v_{M_{11}}, v_{M_{21}})$ and $Y' = SApp_{M_{par}}(\psi, v_{M_{12}}, v_{M_{21}})$, then $Y \subseteq Y'$ since $SApp_{M_{par}}$ is positive monotonic w.r.t. VB^1 and TB_i^1 for $i \in \Gamma$, from IA. Next, $SApp_{M_{par}}(\phi, v_{M_{11}}, v_{M_{21}})$ is $solve_{op}(Y, v_{M_1})$ where M_1 is $M_{under}^{\mathcal{A}}$ or M_{over}^{Γ} . Similarly, $SApp_{M_{par}}(\phi, v_{M_{12}}, v_{M_{21}})$ is $solve_{op}(Y', v_{M_2})$ where M_2 is $M_{under}^{\mathcal{A}}$ or M_{over}^{Γ} . In all the cases, if $v_{M_{11}}(VB) \leq v_{M_{12}}(VB)$ and $v_{M_{11}}(TB_i) \leq v_{M_{12}}(TB_i)$ for $i \in \Gamma$ then $v_{M_1}(VB) \leq v_{M_2}(VB)$ and $v_{M_1}(TB_i) \leq v_{M_2}(TB_i)$ for $i \in \Gamma$. Now observe that if $Y \subseteq Y'$ then $solve_{op}(Y, v_{M_1}) \subseteq solve_{op}(Y', v_{M_1})$.

Next, $solve_{op}(Y', v_{M_1}) \subseteq solve_{op}(Y', v_{M_2})$ since $solve_{op}$ is positive monotonic w.r.t. VB and TB_i for $i \in \Gamma$. Finally, $solve_{op}(Y, v_{M_1}) \subseteq solve_{op}(Y', v_{M_2})$, i.e., $SApp_{M_{par}}(\phi, v_{M_{11}}, v_{M_{21}}) \subseteq SApp_{M_{par}}(\phi, v_{M_{12}}, v_{M_{21}})$, and the function is positive monotonic w.r.t. VB^1 and TB_i^1 for $i \in \Gamma$.

If $Y = SApp_{M_{par}}(\psi, v_{M_{11}}, v_{M_{21}})$ and $Y' = SApp_{M_{par}}(\psi, v_{M_{11}}, v_{M_{22}})$, then $Y \subseteq Y'$ since $SApp_{M_{par}}$ is negative monotonic w.r.t. VB^2 and TB_i^2 for $i \in \Gamma$, from IA. The rest of the proof proceeds similarly like in the case above. Finally, $SApp_{M_{par}}(\phi, v_{M_{11}}, v_{M_{21}}) \subseteq SApp_{M_{par}}(\phi, v_{M_{11}}, v_{M_{22}})$, and the function is negative monotonic w.r.t. VB^2 and TB_i^2 for $i \in \Gamma$.

The algorithm $SApp_{M_{par}}(\phi,V_{m_1},V_{m_2})$, for a model M compatible with M_{par} , computes over and under-approximation of $solve_{\phi}(v_M)$. More precisely, $SApp_{M_{par}}(\phi,v_{M_{over}^{\mathcal{A}}},v_{M_{under}^{\mathcal{A}}})$ returns a set of states represented by a bit vector, which is an over-approximation of $solve_{\phi}(v_M)$ for a model M compatible with M_{par} . This means that if $\iota \in solve_{\phi}(v_M)$, then $\iota \in SApp_{M_{par}}(\phi,v_{M_{over}^{\mathcal{A}}},v_{M_{under}^{\mathcal{A}}})$. Clearly, if $\iota \notin SApp_{M_{par}}(\phi,v_{M_{over}^{\mathcal{A}}},v_{M_{under}^{\mathcal{A}}})$, then there is no model M extending M_{par} such that $M,\iota \models \phi$. Similarly, $SApp_{M_{par}}(\phi,v_{M_{under}^{\mathcal{A}}},v_{M_{over}^{\mathcal{A}}})$ computes an under-approximation of $solve_{\phi}(v_M)$. This means that if $\iota \in SApp_{M_{par}}(\phi,v_{M_{under}^{\mathcal{A}}},v_{M_{over}^{\mathcal{A}}})$ then $\iota \in solve_{\phi}(v_M)$.

Theorem 4.3. Let M_{par} be a partial model and M be a model compatible with M_{par} . Then, for any ATL formulae ϕ, ψ_1, ψ_2 and $p \in \mathcal{PV}$, we have:

$$\begin{array}{l} (1) \ \ for \ \phi \in \{p, \neg \psi_1, \psi_1 \wedge \psi_2\} \ \ and \ each \ \Gamma \subseteq \mathcal{A} \colon \\ SApp_{M_{par}}(\phi, v_{M^{\mathcal{A}}_{under}}, v_{M^{\Gamma}_{over}}) \subseteq solve_{\phi}(v_M); \\ solve_{\phi}(M) \subseteq SApp_{M_{par}}(\phi, v_{M^{\Gamma}_{over}}, v_{M^{\mathcal{A}}_{under}}); \\ (2) \ \ for \ \phi \in \{\langle\langle \Gamma \rangle\rangle X \ \psi_1, \langle\langle \Gamma \rangle\rangle G\psi_1, \langle\langle \Gamma \rangle\rangle \psi_1 U\psi_2\} \colon \\ SApp_{M_{par}}(\phi, v_{M^{\mathcal{A}}_{under}}, v_{M^{\Gamma}_{over}}) \subseteq solve_{\phi}(v_M); \\ solve_{\phi}(M) \subseteq SApp_{M_{par}}(\phi, v_{M^{\Gamma}_{over}}, v_{M^{\mathcal{A}}_{over}}). \end{array}$$

PROOF. By a structural induction on a formula. We prove that (a) $solve_{\phi}(v_M) \subseteq SApp_{M_{par}}(\phi, v_{M_{over}^{\Gamma}}, v_{M_{under}^{\mathcal{A}}})$ and (b) $SApp_{M_{par}}(\phi, v_{M_{under}^{\mathcal{A}}}, v_{M_{over}^{\Gamma}}) \subseteq solve_{\phi}(v_M)$.

The base case. If $\phi=p\in\mathcal{PV}$, then $solve_{\phi}(v_M)$ returns the set of states satisfying p in M, $SApp_{M_{par}}(\phi,v_{M_{over}^{\Gamma}},v_{M_{under}^{\mathcal{A}}})$ returns the set of states satisfying p in M_{over}^{Γ} , and $SApp_{M_{par}}(\phi,v_{M_{under}^{\mathcal{A}}})$ returns the set of states satisfying p in $M_{over}^{\mathcal{A}}$ returns the set of states satisfying p in $M_{under}^{\mathcal{A}}$. The thesis holds by Theorem 4.1 which implies that $v_{M_{under}^{\mathcal{A}}}[VB] \leq v_{M}[VB] \leq v_{M}[VB] \leq v_{M_{over}^{\Gamma}}[VB]$, i.e., the states satisfying p in M are included in the states satisfying p in M_{over}^{Γ} and the states satisfying p in $M_{under}^{\mathcal{A}}$ are included in the states satisfying p in M. Notice that this does not depend on the set of agents Γ since the models M, M_{over}^{Γ} , $M_{under}^{\mathcal{A}}$ have the same states and the transitions do not affect the values of the propositional variables in the states.

The induction step. We show the proof for **(a)** and for $\phi \in \{\neg \psi, \langle \langle \Gamma \rangle \rangle X \psi, \langle \langle \Gamma \rangle \rangle G \psi \}$. The rest of the proof proceeds similarly. Induction assumption (IA): the thesis holds for a formula ψ . Induction hypothesis (IH): the thesis holds for $\phi = op \psi$.

• If $\phi = \neg \psi$, then $solve_{\phi}(v_M) = solve_{\neg}(Y, v_M) \text{ for } Y = solve_{\psi}(v_M) \text{ and}$ $SApp_{M_{par}}(\phi, v_{M_{over}^{\Gamma}}, v_{M_{under}^{\mathcal{A}}}) = solve_{\neg}(Y', v_{M_{under}^{\mathcal{A}}}) \text{ for } Y' = SApp_{M_{par}}(\psi, v_{M_{under}^{\mathcal{A}}}, v_{M_{over}^{\Gamma}}) \text{ and any } \Gamma.$

Observe that $solve_{\neg}(Y,v_M)$ returns the compliment of Y, i.e. $St \setminus Y$. Thus, $solve_{\neg}(Y,v_M) \subseteq solve_{\neg}(Y',v_M)$ since $Y' \subseteq Y$ from IA. Next, $solve_{\neg}(Y',v_M) \subseteq solve_{\neg}(Y',v_M)$ since function $solve_{\neg}(Y,V_m)$ is negative monotonic w.r.t. VB and TB_i for $i \in \mathcal{A}$ from Theorem 3.9 and $v_{M^{\mathcal{A}}_{under}}[VB] \leq v_M[VB]$ and $v_{M^{\mathcal{A}}_{under}}[TB_i] \leq v_M[TB_i]$ for $i \in \mathcal{A}$ from Theorem 4.1. Finally, $solve_{\neg}(Y,v_M) \subseteq solve_{\neg}(Y',v_{M^{\mathcal{A}}_{under}})$ and thus $solve_{\phi}(v_M) \subseteq SApp_{M_{par}}(\phi,v_{M^{\Gamma}_{over}},v_{M^{\mathcal{A}}_{under}})$.

• If $\phi = op \ \psi$ with $op \in \{\langle\langle \Gamma \rangle\rangle X, \langle\langle \Gamma \rangle\rangle G\}$, then $solve_{\phi}(v_M) = solve_{op}(Y, v_M)$ for $Y = solve_{\psi}(v_M)$ and $SApp_{M_{par}}(\phi, v_{M_{over}}^{\Gamma}, v_{M_{under}}^{\mathcal{A}}) = solve_{op}(Y', v_{M_{over}}^{\Gamma})$, where $Y' = SApp_{M_{par}}(\psi, v_{M_{over}}^{\Gamma}, v_{M_{under}}^{\mathcal{A}})$.

Since $Y \subseteq Y'$ from IA, $solve_{op}(Y, V_m)$ is positive monotonic w.r.t. VB and TB_i for $i \in \Gamma$ from Theorem 3.9, and $v_M[VB] \le v_{M_{over}^{\Gamma}}[VB]$ and $v_M[TB_i] \le v_{M_{over}^{\Gamma}}[TB_i]$ for $i \in \Gamma$ from Theorem 4.1, we have $solve_{op}(Y, v_M) \subseteq solve_{op}(Y', v_M) \subseteq solve_{op}(Y', v_{M_{over}^{\Gamma}})$ and thus $solve_{\phi}(v_M) \subseteq SApp_{M_{par}}(\phi, v_{M_{over}^{\Gamma}}, v_{M_{ouder}^{R}})$.

5 SATISFIABILITY AND SYNTHESIS

Since the basic predicate $Model_{g,\phi}(V_m)$ is not monotonic we consider an alternative one: $MApprox_{q,\phi}(V_{m_1},V_{m_2})$.

For two bit vectors v_{M_1} and v_{M_2} encoding models M_1 and M_2 compatible with a partial model M_{par} , we have:

$$\begin{split} \mathit{MApprox}_{g,\phi}(v_{M_1},v_{M_2}) = 1 \text{ iff } g \in \mathit{SApp}_{M_{par}}(\phi,v_{M_1},v_{M_2}). \end{split}$$
 The following corollary follows directly from Theorem 4.2.

Corollary 5.1. $MApprox_{q, \phi}(V_{m_1}, V_{m_2})$ is

- positive monotonic w.r.t. TB_i^1 for $i \in \mathcal{A}$ and VB^1 and negative monotonic w.r.t. TB_i^2 for $i \in \mathcal{A}$ and VB^2 for $\phi \in \{p, \neg p, p \land q\}$, and
- positive monotonic w.r.t. TB_i^1 for $i \in \Gamma$ and VB^1 and negative monotonic w.r.t. TB_i^2 for $i \in \Gamma$ and VB^2 for $\phi \in \{\langle\langle \Gamma \rangle\rangle\rangle Xp$, $\langle\langle \Gamma \rangle\rangle Gp$, $\langle\langle \Gamma \rangle\rangle pUq\}\}$.

Given a monotonic predicate we can design and apply an efficient SAT-modulo-ATL solver which uses SAT Modulo Monotonic Theories (SMMT). This gives us an efficient procedure for ATL satisfiability and synthesis.

The described approach shows that if M is a model of a formula ϕ , then the initial state of *M* belongs to the set of states determined by $SApp_{M_{par}}(\phi,v_{M_{over}^{\Gamma}},v_{M_{under}^{\mathcal{A}}})$. Thus, given an over and under approximation of the set of states satisfying ϕ , we can check whether the initial state of M belongs to this approximation. If not, M is not a model of ϕ . Such an approximation can be computed by a partial assignment built by an SMT solver. In conclusion, the following theorem follows from Theorems 4.2 and 4.3.

Theorem 5.2. Let ϕ be an ATL formula, M be a model with an initial state ι such that $M, \iota \models \phi$, and M is compatible with a partial model M_{par} . Then, we have: $\iota \in SApp_{M_{par}}(\phi, v_{M_{over}^{\mathcal{A}}}, v_{M_{over}^{\mathcal{A}}})$.

The following corollary results directly from this theorem.

 $\begin{array}{ll} \text{Corollary 5.3. } \textit{If } \iota \not \in \textit{SApp}_{M_{par}}(\phi, \upsilon_{M_{over}^{\mathcal{A}}}, \upsilon_{M_{under}^{\mathcal{A}}}), \textit{then } \textit{M}, \iota \not \models \phi. \\ \textit{If } \textit{SApp}_{M_{par}}(\phi, \upsilon_{M_{over}^{\mathcal{A}}}, \upsilon_{M_{under}^{\mathcal{A}}}) &= \emptyset, \textit{then there is no model compatible with } \textit{M}_{par} \textit{such that } \textit{M}, \iota \models \phi. \end{array}$

Now, we are ready to give a procedure for testing satisfiability of the ATL formulae. Basing on the SMMT framework, we have implemented the MsAtl tool - a lazy SMT solver for ATL theory. That is, our implementation exploits a slightly modified MiniSAT[18] as a SAT-solving core, and SApp algorithm as the (main part of the) theory solver for ATL. Due to lack of space we are unable to describe our implementation in detail. However, we sketch below (in a semi-formal way) how our tool works in general.

Input: (a) an ATL formula ϕ , (b) model requirements fixing the number of propositional variables (not less than those appearing in the formula), the number of agents (not less than those appearing in the formula), the number of local states for every agent, an initial local state for every agent, and protocol requirements (if there are any). The requirements determine a partial model M_{par} .

Output: a model satisfying ϕ , which meets the requirements of M_{par} or the answer that such a model does not exist.

Let *d* be an integer variable for tracking the decision depth of the solver, and asg(i) denote the variable assigned at the i-th step.

- (1) Let d := 0.
- (2) Compute $SApp_{M_{par}}(\phi, v_{M_{over}^{\mathcal{A}}}, v_{M_{under}^{\mathcal{A}}})$. (3) If $\iota \in SApp_{M_{par}}(\phi, v_{M_{over}^{\mathcal{A}}}, v_{M_{under}^{\mathcal{A}}})$, then
 - (a) if all variables of V_m are assigned, then return the model.
 - (b) otherwise: d := d + 1, and SAT-solver core, according to its decision policy, assigns a value to the variable $asg(d) \in V_m$. In this way the class of the considered models is narrowed down, and the tool looks for a valuation which encodes a model satisfying ϕ . Go to step (2).
- (4) If $u \notin SApp_{M_{par}}(\phi, v_{M_{over}^{\mathcal{A}}}, v_{M_{under}^{\mathcal{A}}})$, then
 - (a) if d > 0, then compute conflict clause, analyse conflict, undo recent decisions until appropriate depth c, d := c, assign the opposite value to the variable asg(c), and go to step (2).
 - (b) if d = 0 there is no model which meets the requirements and satisfies ϕ . Return UNSAT.

EXPERIMENTAL RESULTS

In order to evaluate the efficiency of our tool we have implemented an ATL formulae generator. Given the number of agents, groups, and propositional variables, and the depth of the formula, the generator (using the normal distribution) draws a random ATL formula up to the given depth. We have compared our preliminary results with TATL [14] - a tableaux-based tool for ATL satisfiability testing. Despite the fact that our implementation is at the prototype stage, and there is a lot of space for further optimizations,³ we have observed several interesting facts. First of all, for small formulae both tools run rather quickly, in fractions of a second. When the size of the formula grows, especially when the number of nested strategy operators increases, the computation time consumed by both tools also grows very quickly. Moreover, we have found that for unsatisfiable formulae our tool runs quite long, especially for a large number of states. This is a typical behaviour for SAT-based methods, which could still be improved by introducing symmetry reductions preventing the exploration of many isomorphic models. However, we have found a class of formulae for which our tool outperforms TATL. These are formulae satisfied by very simple - and often even trivial - models. Table 1 presents the results for a set of such formulae generated with the following parameter values: $|\mathcal{P}V| = 3$, $|\mathcal{A}| = 3$, and number of groups equals 4. The table rows have the following meaning (from top to bottom). The first three rows contain a formula id, the depth of the formula, i.e., the maximal number of nested strategy operators, and the total number of Boolean connectives, respectively. The last two rows present computation times consumed by both tools, in seconds. The experiments have been performed using a PC equipped with Intel i5-7200U CPU and 16GB RAM running Linux.

Table 1: Preliminary experimental results

Id	1	2	3	4	5	6	7	8
Depth	9	13	17	20	23	26	30	33
Con.	13	19	25	31	35	41	49	55
MsAtl[s]	0.22	0.23	0.24	0.31	0.32	0.34	0.38	0.43
TATL[s]	0.58	6.2	29.7	74.6	229	552	1382	3948

Due to lack of space we do not show here all formulae⁴ but only the shortest one. The formula 1 of Table 1 is as follows: $\langle\langle 0\rangle\rangle X(\neg p_0 \lor p_0)$ $\langle\!\langle 1 \rangle\!\rangle G(\neg p_1 \vee \langle\!\langle 0, 1 \rangle\!\rangle F(\neg p_1 \vee \langle\!\langle 0, 1 \rangle\!\rangle F(\neg p_0 \vee \langle\!\langle 2 \rangle\!\rangle F\langle\!\langle 0 \rangle\!\rangle X(\neg p_0 \vee \langle\!\langle 1 \rangle\!\rangle G(\neg p_0 \vee \langle\!\langle 1 \rangle\!) G(\neg p_0 \vee \langle\!\langle 1 \rangle\!\rangle G(\neg p_0 \vee \langle\!\langle 1 \rangle\!) G(\neg p_0 \vee \langle\!\langle 1 \rangle$ $\neg p_1 \lor \langle \langle 0, 1 \rangle \rangle G(\langle \langle 0 \rangle F \neg p_0))))))$. The subsequent formulae are similar but longer.

It is easy to observe that while scaling the depth of the formulae, the computation time of MonoSatATL grows very slowly, almost imperceptibly, contrary to TATL for which it increases significantly.

CONCLUSIONS

The paper introduced a new method exploiting SMMT solvers for (bounded) testing of ATL satisfiability and for constructing (in

³We plan to increase the efficiency of our tool by introducing several optimizations, like, e.g., symmetry reductions, formulae caching, and smart clause-learning.

⁴Additional resources, including a prototype version of our tool, the benchmarks, can be accessed at the (anonymous free hosting) website http://monosatatl.epizy.com

many cases minimal) ATL models. Despite the fact that we apply the method to a restricted class of models for ATL under the standard semantics, our method can be adapted to other classes of multi-agent systems as well as to other ATL semantics including imperfect information. Although our implementation is rather at the preliminary stage, the experimental results show a high potential for this approach.

REFERENCES

- R. Alur, T. A. Henzinger, and O. Kupferman. 1997. Alternating-Time Temporal Logic. In Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS'97). IEEE Computer Society, 100–109.
- [2] R. Alur, T. A. Henzinger, and O. Kupferman. 1998. Alternating-Time Temporal Logic. LNCS 1536 (1998), 23–60.
- [3] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. J. ACM 49(5) (2002), 672–713.
- [4] S. Bayless, N. Bayless, H.H. Hoos, and A.J. Hu. 2015. SAT Modulo Monotonic Theories. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15). AAAI Press, 3702–3709.
- [5] F. Belardinelli. 2014. Reasoning about Knowledge and Strategies: Epistemic Strategy Logic. In Proceedings 2nd International Workshop on Strategic Reasoning, SR 2014, Grenoble, France, April 5-6, 2014. 27–33.
- [6] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. 2019. Imperfect Information in Alternating-Time Temporal Logic on Finite Traces. In PRIMA 2019: Principles and Practice of Multi-Agent Systems - 22nd International Conference, Turin, Italy, October 28-31, 2019, Proceedings. 469–477.
- [7] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. 2012. Synthesis of Reactive(1) designs. J. Comput. Syst. Sci. 78 (2012), 911–938.
- [8] R. Bloem, R. Könighofer, and M. Seidl. 2014. SAT-based synthesis methods for safety specs. In *International Conference on Verification, Model Checking, and Abstract Interpretation*. Springer, 1–20.
- [9] N. Bulling, J. Dix, and W. Jamroga. 2010. Model Checking Logics of Strategic Ability: Complexity. In Specification and Verification of Multi-Agent Systems, M. Dastani, K. Hindriks, and J.-J. Meyer (Eds.). Springer, 125–159.
- [10] N. Bulling, V. Goranko, and W. Jamroga. 2015. Logics for Reasoning About Strategic Abilities in Multi-player Games. In Models of Strategic Reasoning -Logics, Games, and Communities. 93–136.
- [11] N. Bulling and W. Jamroga. 2011. Alternating Epistemic Mu-Calculus. In Proceedings of IJCAI-11. 109–114.
- [12] K. Chatterjee, T.A. Henzinger, and N. Piterman. 2010. Strategy logic. Inf. Comput. 208, 6 (2010), 677–693.
- [13] E.M. Clarke and E.A. Emerson. 1981. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. In Proceedings of Logics of Programs Workshop (Lecture Notes in Computer Science), Vol. 131. 52–71.
- [14] A. David. 2015. Deciding ATL* Satisfiability by Tableaux. In International Conference on Automated Deduction. Springer, 214–228.
- [15] C. Dima, B. Maubert, and S. Pinchinat. 2014. The Expressive Power of Epistemic μ-Calculus. CoRR abs/1407.5166 (2014).
- [16] C. Dima, B. Maubert, and S. Pinchinat. 2015. Relating Paths in Transition Systems: The Fall of the Modal Mu-Calculus. In *Proceedings of MFCS (Lecture Notes in Computer Science)*, Vol. 9234. Springer, 179–191. https://doi.org/10.1007/978-3-662-48057-1_14
- [17] C. Dima and F.L. Tiplea. 2011. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. CoRR abs/1102.4225 (2011).
- [18] N. Eén and N. Sörensson. 2003. An extensible SAT-solver. In International conference on theory and applications of satisfiability testing. Springer, 502–518.
- [19] B. Finkbeiner and S. Schewe. 2013. Bounded synthesis. International Journal on Software Tools for Technology Transfer 15, 5-6 (2013), 519–539.
- [20] V. Goranko and G. Van Drimmelen. 2006. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science* 353, 1-3 (2006), 93–117.
- [21] V. Goranko and D. Shkatov. 2009. Tableau-based decision procedures for logics of strategic ability in multiagent systems. ACM Trans. Comput. Log. 11, 1 (2009), 3:1–3:51.
- [22] D.P. Guelev, C. Dima, and C. Enea. 2011. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal* of Applied Non-Classical Logics 21, 1 (2011), 93–131.
- [23] W. Jamroga, M. Knapik, and D. Kurpiewski. 2017. Fixpoint Approximation of Strategic Abilities under Imperfect Information. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017. 1241–1249.
- [24] W. Jamroga, W. Penczek, P. Dembiński, and A. Mazurkiewicz. 2018. Towards Partial Order Reductions for Strategic Ability. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18).

- 156-165
- [25] A. V. Jones, M. Knapik, W. Penczek, and A. Lomuscio. 2012. Group synthesis for parametric temporal-epistemic logic. In *International Conference on Autonomous* Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes). 1107–1114.
- [26] T. Klenze, S. Bayless, and A.J. Hu. 2016. Fast, Flexible, and Minimal CTL Synthesis via SMT. In Computer Aided Verification, S. Chaudhuri and A. Farzan (Eds.). Springer International Publishing, 136–156.
- [27] P. Kouvaros, A. Lomuscio, and E. Pirovano. 2018. Symbolic Synthesis of Fault-Tolerance Ratios in Parameterised Multi-Agent Systems. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. 324–330.
- [28] O. Kupferman and M.Y. Vardi. 2005. Safraless decision procedures. In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05). IEEE, 531–540.
- [29] A. Lomuscio, H. Qu, and F. Raimondi. 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer* 19, 1 (2017), 9–30.
- [30] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. 2012. A Decidable Fragment of Strategy Logic. CoRR abs/1202.1309 (2012).
- [31] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. 2012. What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic. In CONCUR 2012 Concurrency Theory 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings. 193–208.
- [32] A. Pnueli and R. Rosner. 1989. On the Synthesis of a Reactive Module. In Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '89). ACM, 179–190.
- [33] S. Schewe. 2008. ATL* Satisfiability Is 2EXPTIME-Complete. In Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations. 373–385.
- [34] S. Schewe and B. Finkbeiner. 2007. Distributed Synthesis for Alternating-Time Logics. In Automated Technology for Verification and Analysis, 5th International Symposium, ATVA 2007, Tokyo, Japan, October 22-25, 2007, Proceedings. 268–283.
- [35] P. Y. Schobbens. 2004. Alternating-Time Logic with Imperfect Recall. Electronic Notes in Theoretical Computer Science 85, 2 (2004), 82–93.
- [36] G. van Drimmelen. 2003. Satisfiability in alternating-time temporal logic. In 18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings. IEEE, 208–217.
- [37] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. 2006. ATL satisfiability is indeed EXPTIME-complete. Journal of Logic and Computation 16, 6 (2006), 765– 787.