Towards Assume-Guarantee Verification of Strategic Ability

Extended Abstract

Łukasz Mikulski\textsuperscript{1,2}, Wojciech Jamroga\textsuperscript{2,3}, and Damian Kurpiewski\textsuperscript{2,1}

\textsuperscript{1} Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland
\textsuperscript{2} Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
\textsuperscript{3} Interdisciplinary Centre for Security, Reliability and Trust, SnT, University of Luxembourg, Luxembourg

łukasz.mikulski@mat.umk.pl, wojciech.jamroga@uni.lu, d.kurpiewski@ipipan.waw.pl

ABSTRACT

Formal verification of strategic abilities is a hard problem. We propose to use the methodology of assume-guarantee reasoning in order to facilitate model checking of alternating-time temporal logic with imperfect information and imperfect recall.

KEYWORDS

model checking, assume-guarantee reasoning, strategic ability

1 INTRODUCTION

Alternating-time temporal logic ATL\textsuperscript{*} [1, 2, 32] and Strategy Logic SL [10, 29] provide powerful tools to reason about strategic aspects of MAS. Specifications in agent logics can be used as input to algorithms and tools for model checking [3–5, 7–9, 11, 19, 21, 24, 26]. However, verification of strategic abilities suffers both from state-space and strategy-space explosion. Even for the more restricted logic ATL, model checking of its imperfect information variants ranges from \(\Delta^P_5\)–complete to undecidable [6, 14, 16, 20, 32].

In this paper, we make the first step towards compositional model checking of strategic properties in asynchronous multi-agent systems with imperfect information and imperfect recall. To this end, we creatively expand the assume-guarantee framework of [27, 28]. Instead of searching through the states and strategies of the entire model, we “factorize” it and perform most of the search locally, using assume-guarantee reasoning [12, 31]. We illustrate the approach by means of a simple voting scenario. Finally, we evaluate the practical gains through verification experiments.

Related work. Compositional verification dates back to [18, 23, 30], and has been intensively studied for temporal specifications [12, 13, 15, 17, 25, 27, 28, 31]. Our approach is based on [27, 28], where assume-guarantee rules were defined for liveness properties of MAS. A related scheme for ATL with perfect information strategies and aspect-oriented programs was considered in [13].

2 MODELS OF CONCURRENT MAS

We use the MAS representations of [27, 28], which allow for asynchronous and synchronous composition of local transitions.

Figure 1: Module of a voter deciding between two candidates

Modules. Let \(D\) be a domain (for all variables used in the system). For any set of variables \(X\), let \(D^X\) be a set of all valuation functions on \(X\). We follow [28], and divide the variables in a module into state variables and input variables. An agent can read and modify only its state variables. The input variables are not a part of its state, but their values limit the set of executable transitions.

Definition 2.1 (Module [28]). A module is \(M = (X, I, Q, T, \lambda, q_0)\), where: \(X, I\) are finite sets of state and input variables, respectively, with \(X \cap I = \emptyset\); \(Q\) is a finite set of states; \(\lambda : Q \rightarrow D^X\) labels each state with a valuation of the state variables; \(q_0 \in Q\) is the initial state; \(T \subseteq Q \times D^I \times Q\) is a transition relation. We require that if \((q, \alpha, q') \in T, q \neq q'\), then \((q, \alpha, q) \notin T\), and for each pair \((q, \alpha) \in Q \times D^I\) there exists \(q' \in Q\) such that \((q, \alpha, q') \in T\).

Example 2.2. We use a voting scenario, inspired by [21, 22], consisting of modules \(M^{(1)}, \ldots, M^{(m)}\) of voters and \(M^{(e)}\) of a coercer. Every voter has three local variables in \(X_i^{(1)}:\) vote\(^{(i)}\), the vote being cast (\(?, 1, 2\)); reported\(^{(i)}\): the vote value presented to the coercer (\(?, 1, 2\) or \(!\)); pstatus\(^{(i)}\): the punishment status (\(?\), \(\neq\) or \(!\)). Moreover, \(f^{(i)}\) consists of variable pun\(^{(i)}\) controlled by the coercer. The voter first casts her vote, then decides whether to share its value with the coercer. Finally, she waits for the coercer’s decision to punish her or to refrain from punishment. The module is shown in Figure 1.

The coercer has two available actions per voter: to punish the voter or to refrain from punishment. He can execute them in any order, but only after the respective voters decide to share or not.

Modules \(M, M'\) are asynchronous if \(X \cap X' = \emptyset\). Note that all the modules presented in Example 2.2 are asynchronous.

Composition of Modules. The model of a MAS is given by the asynchronous composition \(M = M^{(1)} \cdot \ldots \cdot M^{(n)}\) that combines modules \(M^{(1)}, \ldots, M^{(n)}\) into a single module \(M\) [28]. The composition is standard; it only requires the compliance of the valuations.
3 WHAT AGENTS CAN ACHIEVE

Alternating-time temporal logic ATL\(^*\) [1, 2, 32] introduces strategic modalities \(\langle\langle\cdot\rangle\rangle\)\(y\), expressing that coalition \(C\) can enforce the temporal property \(y\). In this paper, we use the imperfect information/imperfect recall variant without next step operator \(X\) and nested strategic modalities, denoted sATL\(^*\) ("simple ATL\(^*\)").

Syntax. Formally, the syntax of sATL\(^*\) is defined by:

\[
\phi ::= p(Y) \mid \neg \phi \mid \phi \wedge \phi \mid \langle\langle\cdot\rangle\rangle\gamma; \quad \gamma ::= p(Y) \mid \neg \gamma \mid \gamma \wedge \gamma \mid \gamma U \gamma.
\]

where \(p : Y \to D\) for some subset of domain variables \(Y \subseteq X\). Which is, each atomic statement refers to the valuation of a subset of variables used in the system. \(U\) is the "strong until" operator of LTL. The "sometime" and "always" operators \(F\) and \(G\) can be defined as usual by \(Fy \equiv \top U y\) and \(Gy \equiv \neg (\top U \neg y)\).

Semantics. A memoryless imperfect information strategy for agent \(i\) is a function \(s_i : Q_i \to Ti\). We say that a trace \(\sigma\) (word derived with \(\sigma\)) implements a strategy \(s_i\) if for any \(j\) where \(q^{(i)}_j \neq q^{(i)}_{j+1}\) we have \(s_i(q^{(i)}_j) = (q^{(i)}_{j+1} \cdot a_j^{(i)}(q^{(i)}_j), \lambda(q_j^{(i)}))\). Let \(C \subseteq \{1, \ldots, n\}\) be a set of agent indices. We define joint strategies for \(C\) as tuples of individual strategies, one per \(i \in C\). The semantics of strategic operators is given by the following clause:

\[
M, q \models \langle\langle\cdot\rangle\rangle\gamma \quad \text{if there exists a joint strategy } s_C \text{ for } C \text{ such that, for any word } w \text{ that implements } s_C, \text{ we have } M, w \models \gamma.
\]

4 ASSUMPTIONS AND GUARANTEES

We propose an assume-guarantee scheme, where one can reduce the complexity of model checking sATL\(^*\) by verifying individual strategic abilities of single agents against overapproximating abstractions of their environment, i.e., the rest of the system. The general idea is that if an agent has a successful strategy in a more non-deterministic environment, then it can use the same strategy to succeed in the original model. Moreover, it often suffices to prepare the abstraction based only of the modules that are connected with the agent by at most \(k\) synchronization steps.

Assumptions and Guarantees. The environmental abstractions are formalized by assumptions \(A = (M_A, F)\), where \(M_A\) is a module and \(F\) is a set of accepting states that provide B"uchi-style accepting rules for infinite traces derived by \(M\). The assumption should be constructed so that it guarantees that the set of computations accepted by \(A\) covers the sequences of changes in the input variables \(I_M\) of module \(M\). We capture those changes by the notion of curtailment. Formally, a sequence \(v = v_1v_2\ldots\) over \(D^X\) is a curtailment of sequence \(u = u_1u_2\ldots\) over \(D^X\) (where \(Y \subseteq X\)) if there exists an infinite sequence of indices \(j_1 < j_2 < \ldots\) with \(j_1 = 1\) such that \(Y \setminus \{j_1, \ldots, j_k\} \supseteq u_k \setminus Y\).

Table 1: Results of assume-guarantee verification for simple voting (times given in seconds; timeout=2h)

<table>
<thead>
<tr>
<th>#tr</th>
<th>DFS</th>
<th>Apprx</th>
<th>#tr</th>
<th>DFS</th>
<th>Apprx</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>529</td>
<td>&lt;0.1</td>
<td>161</td>
<td>528</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>3</td>
<td>1.22e4</td>
<td>&lt;0.1</td>
<td>1127</td>
<td>7830</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>4</td>
<td>2.79e5</td>
<td>&lt;0.1</td>
<td>30/Y</td>
<td>7889</td>
<td>1.08e5</td>
</tr>
<tr>
<td>5</td>
<td>6.63e6</td>
<td>&lt;0.1</td>
<td>161</td>
<td>1.45e6</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>6</td>
<td>timeout</td>
<td>timeout</td>
<td>3.86e5</td>
<td>1.92e7</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>7</td>
<td>timeout</td>
<td>timeout</td>
<td>135/Y</td>
<td>135/Y</td>
<td></td>
</tr>
</tbody>
</table>

5 EXPERIMENTS

Here, we present preliminary experimental results for the assume-guarantee rule proposed in Section 4, using the voting scenario of Example 2.2 as the benchmark. The assumptions are provided by a simplified model of the coe¨er, where he only waits for the value reported by Voter\(_j\), no matter how he reacts to other voters’ choices. The algorithms have been implemented in Python, and run on a server with 2.40 GHz Intel Xeon Platinum 8260 CPU, 991 GB RAM, and 64-bit Linux.

The verified formula was \(\phi = \langle\langle \\text{Voter}_1 \rangle\rangle G(\neg \text{status}_1 \vee \text{voted}_1 = 1)\). The results are presented in Table 1. The first column describes the configuration of the benchmark, i.e., the number of the voters. Then, we report the performance of model checking algorithms that operate on the explicit model of the whole system vs. assume-guarantee verification. DFS is a straightforward implementation of depth-first strategy synthesis. Apprx refers to the method of fixpoint-approximation [21]; besides the time, we also report if the approximation was conclusive.

6 CONCLUSION

In this paper, we sketch how assume-guarantee reasoning can be extended for verification of strategic abilities. The main idea is to factorize coalitional abilities by the abilities of the coalition members, and to verify the individual abilities against B"uchi-style abstractions of the agents’ environment of action. Preliminary experimental evaluation has produced very promising results, showing noticeable improvement in the verification of large models consisting of asynchronous agents with independent goals.

ACKNOWLEDGMENTS

We acknowledge the support of the National Centre for Research and Development, Poland (NCBR), and the Luxembourg National Research Fund (FNR), under the PolLux/FNR-CORE project STV (POLLEX-VII/1/2019 – C18/IS/12685695/IS/STV/Ryan).
REFERENCES