Approximate Verification of Strategic Abilities under Imperfect Information Using Local Models

Damian Kurpiewski[1,2], **Wojciech Jamroga**[1,2] and **Yan Kim**[3]
[1]Institute of Computer Science, Polish Academy of Sciences
[2]Nicolaus Copernicus University in Toruń, Poland
[3]Interdisciplinary Centre for Security, Reliability, and Trust, SnT, University of Luxembourg
{d.kurpiewski, w.jamroga}@ipipan.waw.pl, yan.kim@uni.lu

Abstract

Verification of strategic ability under imperfect information is challenging, with complexity ranging from **NP**-complete to undecidable. This is partly because traditional fixpoint equivalences fail in this setting. Some years ago, an interesting idea of *fixpoint approximation* was proposed for model checking of $\mathbf{ATL_{ir}}$, i.e., the logic of strategic ability for agents with imperfect information and imperfect recall. In this paper, we propose a new variant of the approximation, that uses the agent's local model rather than the global model of the system. We prove correctness of the construction, and demonstrate its effectiveness through experimental results on scalable models of voting.

1 Introduction

Logics of strategic ability. Many relevant properties of multi-agent systems (MAS) refer to *strategic abilities* of agents and their groups. Such properties can be conveniently specified in *modal logics of strategic ability*, in particular *Alternating-time Temporal Logic* **ATL** [Alur *et al.*, 2002; Schobbens, 2004] and *Strategy Logic* SL [Mogavero *et al.*, 2010; Mogavero *et al.*, 2014; Berthon *et al.*, 2017]. For example, the **ATL** formula $\langle\langle taxi\rangle\rangle \mathrm{G}\, \neg \mathsf{fatality}$ expresses that the autonomous cab can drive in such a way that no one gets ever killed. Similarly, $\langle\langle taxi, passg\rangle\rangle \mathrm{F}\, \mathsf{destination}$ says that the cab and the passenger have a joint strategy to arrive at the destination, no matter what the other agents do.

Verification of strategic ability. Specifications in agent logics can be used as input to algorithms and tools for *model checking*, that have been in constant development for over 20 years [Alur *et al.*, 1998; Alur *et al.*, 2001; Kacprzak and Penczek, 2004; Lomuscio and Raimondi, 2006; Chen *et al.*, 2013; Busard *et al.*, 2014; Pilecki *et al.*, 2014; Huang and van der Meyden, 2014; Cermak *et al.*, 2014; Lomuscio *et al.*, 2017; Cermák *et al.*, 2015; Belardinelli *et al.*, 2017; Belardinelli *et al.*, 2017; Jamroga *et al.*, 2019; Kurpiewski *et al.*, 2019; Kurpiewski *et al.*, 2021; Kaminski *et al.*, 2024]. However, model checking of strategic abilities is hard, both theoretically and in practice. First, it suffers from the well-known state/transition-space explosion. Moreover, the space of possible strategies is at least exponential *on top of the state-space explosion*, and incremental synthesis of strategies is not possible in general – especially in the realistic case of agents with partial observability. Even for the more restricted (and computation-friendly) logic **ATL**, model checking of its imperfect information variants is $\mathbf{\Delta_2^P}$- to **PSPACE**-complete for agents playing memoryless strategies [Bulling *et al.*, 2010; Schobbens, 2004] and **EXPTIME**-complete to undecidable for agents with perfect recall [Dima and Tiplea, 2011; Guelev *et al.*, 2011]. The theoretical results concur with outcomes of empirical studies on benchmarks [Busard *et al.*, 2015; Jamroga *et al.*, 2019; Lomuscio *et al.*, 2017], as well as attempts at verification of real-life multi-agent scenarios [Jamroga *et al.*, 2020b; Kim *et al.*, 2022; Kurpiewski *et al.*, 2023].

Fixpoint approximation. An interesting idea was proposed in [Jamroga *et al.*, 2019] for model checking of $\mathbf{ATL_{ir}}$, i.e., the variant of **ATL** for agents with imperfect information and imperfect recall. There, we provided a translation of $\mathbf{ATL_{ir}}$ formulas to *alternating epistemic μ-calculus*, such that, whenever the translation of ϕ was true in a given model M, the original formula must also hold there. Thus, in a sense the translation provides a lower bound for the optimal output of verification. The approximation proved conclusive and relatively efficient in many instances of benchmarks [Jamroga *et al.*, 2019], including models of real-life voting procedures [Kurpiewski *et al.*, 2022]. Still, it operates on the *global model of the system*, which means that it still suffers from the state- and transition-space explosion.

Contribution. We observe that, for asynchronous MAS, one can leverage the additional information coming from the modular representation of the system [Jamroga *et al.*, 2020a]. In particular, the fixpoint approximation algorithm in [Jamroga *et al.*, 2019], has the property that, with most fixpoint iterations, it adds or removes whole *epistemic classes*, rather than arbitrary subsets of global states. Since epistemic classes in the global model correspond to local states in the modular representation, one can equivalently do the fixpoint computation on the (exponentially smaller) local model(s) of the given agent(s). We formalize the idea through a translation to appropriate μ-calculus formulas interpreted in the local model, prove its correctness, and evaluate its efficiency through experiments on two scalable benchmarks.

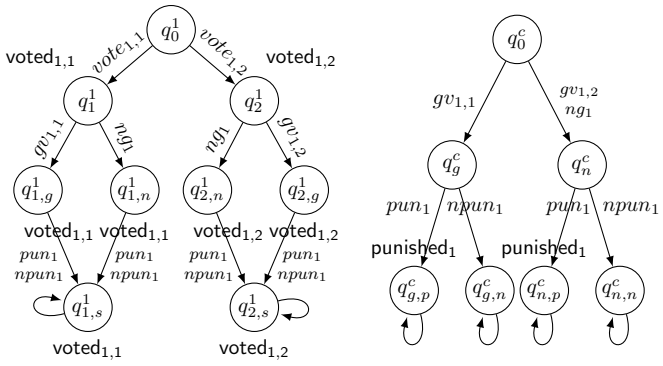

Figure 1: ASV$_1^2$: agents $Voter_1$ (left) and $Coercer$ (right)

In this paper, we only consider individual abilities and observable reachability/safety goals. The case of proper coalitions and non-observable goals is left for future work.

2 Preliminaries

We start by introducing representations of asynchronous MAS, their models, and the relevant logical formalisms. In our presentation, we follow [Jamroga *et al.*, 2021].

2.1 Models of Asynchronous Interaction

Definition 1 (Asynchronous MAS)**.** *An* asynchronous multi-agent system (AMAS) S *consists of* n *agents* $\mathcal{A} = \{1, \ldots, n\}$*, each associated with a tuple* $A_i = (L_i, \iota_i, Evt_i, R_i, T_i, PV_i, V_i)$ *including a set of* local states $L_i = \{l_i^1, l_i^2, \ldots, l_i^{n_i}\}$*, a designated* initial state $\iota_i \in L_i$*, a nonempty finite set of* events $Evt_i = \{\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{m_i}\}$*, and a* repertoire of choices $R_i : L_i \to 2^{2^{Evt_i}}$*. For each* $l_i \in L_i$*,* $R_i(l_i) = \{\text{E}_1, \ldots, \text{E}_m\}$ *is a nonempty list of nonempty choices available to* i *at* l_i*. If the agent chooses* $\text{E}_j = \{\alpha_1, \alpha_2, \ldots\}$*, then only an event in* E_j *can be executed at* l_i *within the agent's module. Moreover,* $T_i \subseteq L_i \times Evt_i \times L_i$ *is a* local transition relation*, where* $(l_i, \alpha, l_i') \in T_i$ *represents that event* $\alpha \in \bigcup R_i(l_i)$ *changes the local state from* l_i *to* l_i'*. Agents are endowed with mutually disjoint, finite and possibly empty sets of* local propositions PV_i*, and their* valuations $V_i : L_i \to 2^{PV_i}$.

Note that each agent "owns" the events affecting its state, but some of the events may be shared with other agents. Those can only be executed synchronously by all the involved parties. Moreover, the agent's strategic choices are restricted by its repertoire function.

Example 1 (Asynchronous Simple Voting)**.** *Consider a simple voting system ASV$_n^k$ with* $n + 1$ *agents (*n *voters and* 1 *coercer). Each* $Voter_i$ *agent can cast her vote for a candidate* $\{1, \ldots, k\}$*, and decide whether to share her vote receipt with the* $Coercer$ *agent. The coercer can choose to punish the voter or refrain from it. A graphical representation of the agents for* $n = 1, k = 2$ *is shown in Fig. 1. We assume that the coercer only registers if the voter hands in a receipt for candidate* 1 *or not. The repertoire of the coercer is defined as* $R_c(q_0^c) = \{\{gv_{1,1}, gv_{1,2}, ng_1\}\}$ *and* $R_c(q_g^c) = R_c(q_n^c) = \{\{pun_1\}, \{npun_1\}\}$*, i.e., the coercer first* receives *the voter's decision regarding the receipt, and then* controls *whether the voter is punished or not. Analogously, the voter's repertoire is given by:* $R_1(q_0^1) = \{\{vote_{1,1}\}, \{vote_{1,2}\}\}$*,* $R_1(q_j^1) = \{\{gv_{1,j}\}, \{ng_1\}\}$ *for* $j = 1, 2$*, and* $R_1(q_{1,g}^1) = R_1(q_{1,n}^1) = R_1(q_{2,g}^1) = R_1(q_{2,n}^1) = \{\{pun_1, npun_1\}\}$.

Notice that the coercer cannot determine which of the events $gv_{1,1}, gv_{1,2}, ng_1$ will occur; this is entirely under the voter's control. This way we model the situation where it is the decision of the voter to show her vote or not. Similarly, the voter cannot avoid punishment by choosing the strategy allowing only $npun_1$, because the choice $\{npun_1\}$ is *not* in the voter's repertoire. She can only execute $\{pun_1, npun_1\}$, and await the decision of the coercer.

The execution semantics is based on interleaving with synchronization on shared events.

Definition 2 (Interleaved interpreted system)**.** *Let* S *be an AMAS with* n *agents. The* interleaved interpreted system $IIS(S)$ *extends* S *with: (i) the initial states* $\iota = (\iota_1, \ldots, \iota_n)$*; (ii) the set of global states* $St \subseteq L_1 \times \ldots \times L_n$ *that collects all the configurations of local states, reachable from* ι *by* T *(see below); (iii) the (partial)* global transition function $T : St \times Evt \rightharpoonup St$*, defined by* $T(g_1, \alpha) = g_2$ *iff* $T_i(g_1^i, \alpha) = g_2^i$ *for all* $i \in Agent(\alpha)$ *and* $g_1^i = g_2^i$ *for all* $i \in \mathcal{A} \setminus Agent(\alpha)$*;*[1] *(iv) the* global valuation *of propositions* $V : St \to 2^{PV}$*, defined as* $V(l_1, \ldots, l_n) = \bigcup_{i \in \mathcal{A}} V_i(l_i)$.

We will sometimes write $g_1 \xrightarrow{\alpha} g_2$ *instead of* $T(g_1, \alpha) = g_2$*. Also, we define relation* $\sim_A = \{(g, g') \in St \times St \mid \exists i \in A \,.\, g^i = g'^i\}$ *to connect states that are indistinguishable for at least one agent* $i \in A$.

Definition 3 (Enabled events)**.** *Let* $A = \{a_1, \ldots, a_k\} \subseteq \mathcal{A} = \{1, \ldots, n\}$ *and* $\overrightarrow{E}_A = (\text{E}_{a_1}, \ldots, \text{E}_{a_k})$ *for some* $k \leq n$*, such that* $\text{E}_i \in R_i(g^i)$ *for every* $i \in A$*. Event* $\beta \in Evt$ *is* enabled by the vector of choices $\overrightarrow{E}_A$ at $g \in St$ *iff, for every* $i \in Agent(\beta) \cap A$*, we have* $\beta \in \text{E}_i$*, and for every* $i \in Agent(\beta) \setminus A$*, it holds that* $\beta \in \bigcup R_i(g^i)$*. That is, the "owners" of* β *in* A *have selected choices that admit* β*, while all the other "owners" of* β might *select choices that do the same. We denote the set of such events by* $enabled(g, \overrightarrow{E}_A)$.

Some combinations of choices enable no events. To account for this, the models of AMAS are augmented with "silent" ϵ-loops, added when no "real" event can occur.

Definition 4 (Undeadlocked IIS)**.** *Let* S *be an AMAS, and assume that no agent in S has* ϵ *in its alphabet of events. The* model of S*, denoted* $IIS^\epsilon(S, I)$*, extends the model* $IIS(S, I)$ *as follows:*

- $Evt_{IIS^\epsilon(S)} = Evt_{IIS(S)} \cup \{\epsilon\}$*, where* $Agent(\epsilon) = \emptyset$*;*
- *For each* $g \in St$*, we add the transition* $g \xrightarrow{\epsilon} g$ *iff there is a selection of all agents' choices* $\overrightarrow{E}_{\mathcal{A}} = (\text{E}_1, \ldots, \text{E}_n)$*, such that* $\text{E}_i \in R_i(g^i)$ *and* $enabled_{IIS(S,I)}(g, \overrightarrow{E}_{\mathcal{A}}) = \emptyset$*. Then, for every* $A \subseteq \mathcal{A}$*, we also fix* $enabled_{IIS^\epsilon(S)}(g, \overrightarrow{E}_A) = enabled_{IIS^\epsilon(S)}(g, \overrightarrow{E}_A) \cup \{\epsilon\}$.

[1] g^i denotes agent i's state in $g = (l_1, \ldots, l_n)$, i.e., $g^i = l_i$.

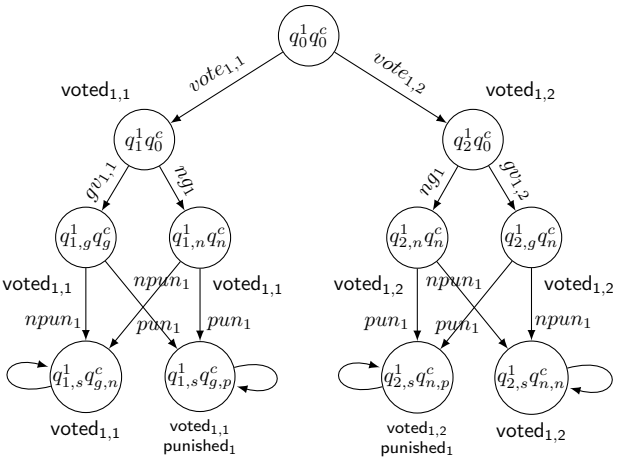

Figure 2: The model $IIS^{\epsilon}(ASV_1^2)$

In other words, an ϵ-loop is enabled whenever E_A allows the grand coalition to collectively block the execution of any "real" event.

Example 2. *The model of ASV_1^2 is shown in Figure 2. Note that it contains no ϵ-transitions, since no choices of the voter and the coercer can cause a deadlock.*

2.2 Reasoning About Strategies

Let PV be a set of propositions and $\mathcal{A}$ the set of all agents. The syntax of *alternating-time logic* **ATL** [Alur *et al.*, 2002; Schobbens, 2004] is given by:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{X}\,\varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{G}\,\varphi \mid \langle\!\langle A\rangle\!\rangle \varphi\,\mathrm{U}\,\varphi.$$

where $\mathsf{p} \in PV$, $A \subseteq \mathcal{A}$, X stands for "next," G for "always from now on," U for "until," and $\langle\!\langle A\rangle\!\rangle\gamma$ for "agent coalition A has a strategy to enforce γ." Derived boolean connectives and constants ($\vee, \top, \bot$) are defined as usual. "Sometime" is defined as $\mathrm{F}\,\varphi \equiv \top\,\mathrm{U}\,\varphi$.

Example 3. Voter enfranchisement *can be expressed by formula* $\bigwedge_{\mathsf{j}\in Candidates} \langle\!\langle Voter\rangle\!\rangle \mathrm{F}\,\mathsf{voted_j}$.

Strategic ability of agents. Following [Jamroga *et al.*, 2021], a *positional imperfect information strategy (*ir-*strategy) for agent i* is defined by a function $\sigma_i\colon L_i \to 2^{Evt_i}$, such that $\sigma_i(l) \in R_i(l)$ for each $l \in L_i$. Note that σ_i is uniform by construction, as it is based on local, and not global states. The set of such strategies is denoted by Σ_i^{ir}. Joint strategies Σ_A^{ir} for $A = \{a_1, \dots, a_k\} \subseteq \mathcal{A}$ are defined as usual, i.e., as tuples of strategies σ_i, one for each agent $i \in A$. By $\sigma_A(g) = (\sigma_{a_1}(g), \dots, \sigma_{a_k}(g))$, we denote the joint choice of coalition A at global state g. An infinite sequence of global states and events $\pi = g_0\alpha_0 g_1\alpha_1 g_2 \dots$ is called a *path* if $g_j \xrightarrow{\alpha_j} g_{j+1}$ for every $j \geq 0$. The set of all paths in model M starting at state g is denoted by $\Pi_M(g)$.

Definition 5 (Standard outcome)**.** *Let $A \subseteq \mathcal{A}$. The* standard outcome *of strategy $\sigma_A \in \Sigma_A^{\mathrm{ir}}$ in state g of model M is the set $out_M^{\mathrm{Std}}(g, \sigma_A) \subseteq \Pi_M(g)$ such that $\pi = g_0\alpha_0 g_1\alpha_1 \dots \in out_M(g, \sigma_A)$ iff $g_0 = g$, and for each $m \geq 0$ we have that $\alpha_m \in enabled_M(g_m, \sigma_A(g_m))$.*

Definition 6 (Reactive outcome)**.** *The* reactive outcome *is the set $out_M^{\mathrm{React}}(g, \sigma_A) \subseteq out_M^{\mathrm{Std}}(g, \sigma_A)$ such that $\pi = g_0\alpha_0 g_1\alpha_1 \dots \in out_M^{\mathrm{React}}(g, \sigma_A)$ iff $\alpha_m = \epsilon$ implies $enabled_M(g_m, \sigma_A(g_m)) = \{\epsilon\}$.*

Intuitively, the standard outcome collects all the paths where agents in A follow σ_A, while the others freely choose from their repertoires. The reactive outcome includes only those outcome paths where the opponents cannot miscoordinate on shared events. Let $x \in \{\mathrm{Std}, \mathrm{React}\}$. We extend the above definitions to subsets of global states $G \subseteq St$ by $out_M^x(G, \sigma_A) = \bigcup_{g\in G} out_M^x(g, \sigma_A)$. Now, the ir-semantics of **ATL** in asynchronous MAS [Alur *et al.*, 2002; Schobbens, 2004; Jamroga *et al.*, 2021] is defined by:

$M, g \models^x \mathsf{p}$ iff $g \in V(\mathsf{p})$, for $\mathsf{p} \in PV$;

$M, g \models^x \neg\varphi$ iff $M, g \not\models^x \varphi$;

$M, g \models^x \varphi_1 \wedge \varphi_2$ iff $M, g \models^x \varphi_1$ and $M, g \models^x \varphi_2$;

$M, g \models^x \langle\!\langle A\rangle\!\rangle \mathrm{X}\,\varphi$ iff there is $s_A \in \Sigma_A^{ir}$ such that, for each path $\lambda \in out_M^x([g]_{\sim_A}, s_A)$, we have $M, \lambda[1] \models^x \varphi$.

$M, g \models^x \langle\!\langle A\rangle\!\rangle \mathrm{G}\,\varphi$ iff there is $s_A \in \Sigma_A^{ir}$ such that, for each $\lambda \in out_M^x([g]_{\sim_A}, s_A)$ and $i \geq 0$, we get $M, \lambda[i] \models^x \varphi$.

$M, g \models^x \langle\!\langle A\rangle\!\rangle \varphi_1\,\mathrm{U}\,\varphi_2$ iff there is $s_A \in \Sigma_A^{ir}$ such that, for each path $\lambda \in out_M^x([g]_{\sim_A}, s_A)$, we have $M, \lambda[i] \models^x \varphi_2$ for some $i \geq 0$ and $M, \lambda[j] \models^x \varphi_1$ for all $0 \leq j < i$.

Example 4. *Let $M = IIS^{\epsilon}(ASV_n^k)$, i.e., the model of the AMAS in Example 1. Note that the* Std *and* React *semantics coincide on M, as it includes no ϵ-transitions. Clearly, $M, (q_0^1, \dots, q_0^n, q_0^c) \models \bigwedge_{\mathsf{j}\in Candidates} \langle\!\langle Voter\rangle\!\rangle \mathrm{F}\,\mathsf{voted_j}$.*

In this paper, we focus on formulas with no next step operators X and no nested strategic modalities. The corresponding "simple" subset of **ATL** is denoted by **sATL**. The restriction is less prohibitive than it seems at a glance. First, the X operator is of little value for asynchronous systems. Secondly, nested strategic modalities would only allow us to express an agent's ability to endow another agent with ability (or deprive the other agent of ability). Such properties are sometimes relevant, but simpler properties like $\langle\!\langle Voter\rangle\!\rangle \mathrm{F}\,\mathsf{voted_j}$ or $\langle\!\langle Voter\rangle\!\rangle \mathrm{G}\,\neg\mathsf{punished}$ are usually of more interest.

2.3 Alternating Mu-Calculus

For strategies with perfect information, **ATL** can be embedded in a variant of μ-calculus with $\langle\!\langle A\rangle\!\rangle$X as the basic modality and no alternation of fixpoint operators [Alur *et al.*, 2002]. At the same time, the analogous variant of μ-calculus for imperfect information has incomparable expressive power to $\mathbf{ATL_{ir}}$ [Bulling and Jamroga, 2011].

Formally, *alternation-free alternating μ-calculus with imperfect information (af-AEμC)* takes the next-time fragment of $\mathbf{ATL_{ir}}$ and adds the least fixpoint operator μ. The greatest fixpoint operator ν is defined as dual to μ. Let $\mathcal{V}ars$ be a set of second-order variables ranging over 2^{St}. The language of af-AEμC is defined by:

$$\phi ::= \mathsf{p} \mid Z \mid \neg\phi \mid \phi \wedge \phi \mid \langle A\rangle\phi \mid \mu Z(\phi) \mid K_a\phi,$$

where $\mathsf{p} \in PV$, $Z \in \mathcal{V}ars$, $a \in \mathcal{A}$, $A \subseteq \mathcal{A}$, and the formulae are Z–positive, i.e., each free occurrence of Z is

in the scope of an even number of negations. We define $\nu Z(\phi(Z)) \equiv \neg\mu Z(\neg\phi(\neg Z))$, where $\phi(\neg Z)$ denotes the result of substituting in ϕ all free occurrences of Z with $\neg Z$. Additionally, every formula of af-AEμC in its negation normal form must contain no occurrences of ν (resp. μ) on any syntactic path from an occurrence of μZ (resp. νZ) to a bound occurrence of Z.

We evaluate the formulae of af-AEμC with respect to the valuations of $\mathcal{V}ars$, i.e., functions $\mathcal{V}\colon \mathcal{V}ars \to 2^{St}$. We denote the set of all the valuations of $\mathcal{V}ars$ by $\mathcal{V}als$. If $X \in \mathcal{V}ars$, $Z \subseteq St$, and $\mathcal{V} \in \mathcal{V}als$, then by $\mathcal{V}[X := Z]$ we denote the valuation of $\mathcal{V}ars$ such that $\mathcal{V}[X := Z](Y) = \mathcal{V}(Y)$ for $Y \neq X$ and $\mathcal{V}[X := Z](X) = Z$. The denotational semantics of af-AEμC assigns to each formula ϕ the set of states $[\![\phi]\!]^M_{\mathcal{V}}$ where ϕ is true under the valuation $\mathcal{V} \in \mathcal{V}als$:

- $[\![\mathsf{p}]\!]^M_{\mathcal{V}} = \mathcal{V}(\mathsf{p})$,
- $[\![Z]\!]^M_{\mathcal{V}} = \mathcal{V}(Z)$,
- $[\![\neg\phi]\!]^M_{\mathcal{V}} = St \setminus [\![\phi]\!]^M_{\mathcal{V}}$,
- $[\![\phi \wedge \psi]\!]^M_{\mathcal{V}} = [\![\phi]\!]^M_{\mathcal{V}} \cap [\![\psi]\!]^M_{\mathcal{V}}$,
- $[\![\langle A\rangle\phi]\!]^M_{\mathcal{V}} = \{q \in St \mid \exists s_A \in \Sigma_A \ \forall \lambda \in out^{ir}_M(q, s_A)\ \lambda[1] \in [\![\phi]\!]^M_{\mathcal{V}}\}$,
- $[\![\mu Z(\phi)]\!]^M_{\mathcal{V}} = \bigcap\{Q \subseteq St \mid [\![\phi]\!]^M_{\mathcal{V}[Z:=Q]} \subseteq Q\}$,

where $\phi \in$af-AEμC, $\mathsf{p} \in PV$, $Z \in \mathcal{V}ars$, $A \subseteq \mathcal{A}$, and $a \in \mathcal{A}$. If ϕ is a sentence, i.e., it contains no free variables, then its validity does not depend on the valuation $\mathcal{V}$, and we write $M, q \models \phi$ instead of $q \in [\![\phi]\!]^M_{\mathcal{V}}$.

3 New Fixed-Point Approximations

In multi-agent systems, the verification of properties often requires analyzing the global model, which can be computationally expensive due to the state space explosion problem. To mitigate this, we propose focusing on the local model of an agent, which is a more tractable approach. By approximating the global model from the perspective of selected agent, we can efficiently verify properties while maintaining a high degree of accuracy.

3.1 Local Approximation Models of Ability

Definition 7 (Local Approximating Model)**.** *The local approximating model for an agent i captures the essential behavior of the agent within the context of the entire system. This model is defined as a tuple $M_i = (L_i, Evt_i, R_i, PV_i, Tapp_i)$, where L_i represents the set of local states, Evt_i the set of events, R_i the repertoire function, PV_i the set of propositions, and $Tapp_i$ the transition relation. The transition relation $Tapp_i$ is particularly crucial as it defines how the agent transitions between local states based on events. Additionally, to account for the potential livelock for i, we introduce an auxliary event symbol τ, such that $Agents(\tau) = \emptyset$.*

In this section, we formalize the new fixpoint approximation by defining the transition relation in the local approximating model and proving its properties. We also introduce the concepts of enabled events, standard outcomes, and reactive outcomes within this model. These definitions lay the groundwork for the execution semantics of the approximated model and the subsequent model checking procedures.

By leveraging the local approximating model, we aim to provide a scalable and efficient method for verifying multi-agent systems, ensuring that the verification process remains feasible even as the complexity of the system increases.

Definition 8 (Transition Relation in Local Approximating Model)**.** *The transition relation $Tapp_i$ is defined as follows:*

- $(l, \epsilon, l) \in Tapp_i$ *if there exist $g \in St$ such that $g^i = l$ and $T(g, \epsilon) = g$.*
- $(l, \tau, l) \in Tapp_i$ *if there exist $g_1, g_2, \ldots, g_{n+1} \in St$ and $\alpha_1, \alpha_2, \ldots, \alpha_n \in Evt \setminus Evt_i$ such that:*
  - $\forall j \in \{1, 2, \ldots, n\}, (g_j)^i = l$ *and* $T(g_j, \alpha_j) = g_{j+1}$,
  - $g_{n+1} = g_1$.

  In other words, there exists a sequence of global states in the global model that starts and ends in the global state containing the local state l, with all transitions in-between labeled by events that are not in the agent i's set of events.
- $(l, \alpha, l') \in Tapp_i$ *for $\alpha \in Evt_i$ if there exist $g_1, g_2, \ldots, g_{n+1} \in St$ and $\alpha_1, \alpha_2, \ldots, \alpha_n \in Evt$ such that:*
  - $\forall j \in \{1, 2, \ldots, n\}, (g_j)^i = l$ *and* $T(g_j, \alpha_j) = g_{j+1}$,
  - $\forall j \in \{1, 2, \ldots, n-1\}, i \notin Agent(\alpha_j)$,
  - $(g_{n+1})^i = l'$ *and* $\alpha_n = \alpha$.

  In other words, there exists a finite path fragment in the global model that runs entirely within the local state l (except possibly the last transition) and ends in a global state containing the local state l', with the last transition labeled by α and no other event executed by i.

The transition relation $Tapp_i$ captures the essential behavior of agent i within the global model. It defines how the agent transitions between local states based on events, ensuring that the local approximating model accurately represents the agent's interactions with the system. Consistent with the standard AMAS S (Definition 1), the transition relation is characterized as a partial transition function, defined for events within the agent's repertoire of choices.

In order to talk about agent's strategies, we need to define the enabled events in the local approximating model. An event is enabled at a local state l if there exists a transition from l to another local state l' labeled with that event. This concept is essential for defining the standard and reactive outcomes of a strategy in the local approximating model.

Definition 9 (Enabled events in Local Approximating Model)**.** *Event $\alpha \in Evt_i$ is* enabled *at state $l \in L_i$ in the local approximating model M_i if $l \xrightarrow{\alpha}_{M_i} l'$ for some $l' \in L_i$, i.e., $Tapp_i(l, \alpha) = l'$. The set of such events is denoted by $enabled_{M_i}(l)$.*

The standard outcome captures the possible sequences of events that can be executed by the agent according to its strategy, while the reactive outcome further refines this by considering the enabled events at each local state.

Definition 10 (Standard outcome in Local Approximating Model)**.** *Let $i \in \mathcal{A}$. The* standard outcome *of strategy $\sigma_i \in$*

Σ_i^{ir} *in state* $l \in L_i$ *of the local approximating model* M_i *is the set* $out^{\text{Std}}_{M_i}(l, \sigma_i) \subseteq \Pi_{M_i}(l)$ *such that* $\pi = l_0\alpha_0 l_1 \alpha_1 \cdots \in out^{\text{Std}}_{M_i}(l, \sigma_i)$ *iff* $l_0 = l$, *and for each* $m \geq 0$ *we have that* $\alpha_m \in enabled_{M_i}(l_m, \sigma_i(l_m))$.

Definition 11 (Reactive outcome in Local Approximating Model)**.** *Let* $i \in \mathcal{A}$. *The* reactive outcome *of strategy* $\sigma_i \in \Sigma_i^{ir}$ *in state* $l \in L_i$ *of the local approximating model* M_i *is the set* $out^{\text{React}}_{M_i}(l, \sigma_i) \subseteq out^{\text{Std}}_{M_i}(l, \sigma_i)$ *such that* $\pi = l_0\alpha_0 l_1 \alpha_1 \cdots \in out^{\text{React}}_{M_i}(l, \sigma_i)$ *iff* $\alpha_m = l\epsilon$ *for each* $m \geq 0$ *implies* $enabled_{M_i}(l_m, \sigma_i(l_m)) = \{\epsilon\}$.

3.2 Fixpoint Approximation on Local Models

Now we can define the model checking procedure for the local approximating model. The model checking of a formula ϕ in the local approximating model M_i is performed by checking whether the formula holds for all sequences in the outcome of the agent's strategy in the initial state of the model.

Definition 12 (Model Checking Local Approximating Model)**.** *Let* $x \in \{\text{Std}, \text{React}\}$ *and* $i \in \mathcal{A}$. $M_i, l \models^x \langle\langle i \rangle\rangle \phi$ *iff there is a strategy* $\sigma_i \in \Sigma_i^{ir}$ *such that for all* $\pi \in out^x_{M_i}(l, \sigma_i)$ *we have* $M_i, \pi \models^x \phi$.

To utilize the local approximating model for verifying the formula ϕ in the original global model, we must impose certain restrictions on the set of possible formulas. Specifically, all propositions that appear in ϕ must belong to the set PV_i of agent i. This requirement arises because the approximated model retains only the local states of the agent from the original model, omitting information about other agents. Consequently, we focus exclusively on singleton coalitions in our formulas. Additionally, similar to standard model checking for AMAS S, we restrict our attention to formulas that exclude next step operators X and nested strategic modalities. These constraints ensure the feasibility and accuracy of verification within the local approximating model framework.

We use the following translations of simple formulas of $\mathbf{ATL_{ir}}$, that provide the lower approximation.

Definition 13 (Lower Approximation for $\mathbf{sATL}_{ir}$)**.**

1. $tr_L(\langle\langle i \rangle\rangle \mathrm{F}\, \phi) = \mu Z.(\phi \vee \langle i \rangle Z)$;
2. $tr_L(\langle\langle i \rangle\rangle \mathrm{G}\, \phi) = \nu Z.(\phi \wedge \langle i \rangle Z$;
3. $tr_L(\langle\langle i \rangle\rangle \psi\, \mathrm{U}\, \phi) = \mu Z.(\phi \vee (\psi \wedge \langle i \rangle Z))$.

Next, we prove that, if $tr_L(\phi)$ is satisfied in the local model, then ϕ must hold in the corresponding global model.

3.3 Correctness of the Approximation

Here, we present the main theoretical results of the paper, showing that procedure proposed in Section 3 provides a lower approximation of the verification problem for $\mathbf{ATL_{ir}}$. To this end, we introduce the additional concept of a *partial ir-strategy*, which is simply a partial function $\sigma_i \colon L_i \rightharpoonup 2^{Evt_i}$ with the standard constraints. The outcome of such σ_i is defined analogously (note that it can contain infinite as well as finite paths!). Additionally, for $l \in L_i$ and $L \subseteq L_i$, we will write "$g \in l$" instead of "$g^i = l$," and "$g \in L$" instead of "$g \in l$ for some $l \in L$."

Theorem 1 (Approximation of reachability)**.** *Let* $x \in \{\text{Std}, \text{React}\}$, $i \in \mathcal{A}$, *and* $l \in St_{M_i}$. *If* $M_i, l \models^x \mu Z.(\phi \vee \langle i \rangle Z)$ *then, for every* $g \in l$, *we have* $M, g \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

Theorem 1 is a direct consequence of the following lemma.

Lemma 1. *Let* $Sat^x_n(M_i, \mu Z.(\phi \vee \langle i \rangle Z))$ *be the set of states in* M_i, *computed by the standard least fixpoint algorithm for* $\mu Z.(\phi \vee \langle i \rangle Z)$ *with at most* n *fixpoint iterations (equivalently: with at most* n *executions of the preimage operation for* $\langle i \rangle$*).*

We claim that, for every $n \in \mathbb{N}$, *if* $L = Sat^x_n(M_i, \mu Z.(\phi \vee \langle i \rangle Z))$ *then, for every* $g \in L$, *we have* $M, g \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

Proof. Proof by induction on n.

Base Case, $n = 0$**:** If $n = 0$, then $L = Sat^x_0(M_i, \mu Z.(\phi \vee \langle i \rangle Z))$ contains only the states where ϕ is satisfied. Therefore, for every $g \in St_M$ such that $g^i = l \in L$, we have $M, g \models^x \phi$. Hence, $M, g \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

Inductive Step: Assume that the lemma holds for some $n \geq 0$. We need to show that it also holds for $n + 1$. Let $L_{n+1} = Sat^x_{n+1}(M_i, \mu Z.(\phi \vee \langle i \rangle Z))$. By the definition of the least fixpoint algorithm, L_{n+1} is the set of states where either ϕ is satisfied or there exists a transition to a state in L_n. Formally, $L_{n+1} = \{l \mid M_i, l \models \phi \vee \exists_{l' \in L_n}(l, \alpha, l') \in Tapp_i\}$.

Consider any $g \in St_M$ such that $g^i = l \in L$. We have two cases to consider:

1. $M_i, l \models \phi$: In that case, $M, g \models^x \phi$, and hence $M, g \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

2. $\exists_{l' \in L_n}(l, \alpha, l') \in Tapp_i$: By the inductive hypothesis, for every $g' \in St_M$ st. $g'^i = l'$, we have $M, g' \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

Since $(l, \alpha, l') \in Tapp_i$, there exists a corresponding set of transitions in the original model M from state g to some states g' such that $g'^i = l'$. Since l' was added in the current iteration of the least fixpoint algorithm, it means that there exists an event $\alpha \in Evt_i$ such that $(l, \alpha, l') \in Tapp_i$. Therefore, there exists a partial strategy in M for agent i to ensure that ϕ is eventually satisfied starting from state g. By following this strategy, agent i can navigate through the states in M to reach a state where ϕ holds, thus satisfying the reachability condition in the original model. Hence, $M, g \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

Formally, we proceed as follows:

(i) By the induction lemma, we obtain a partial strategy σ_a defined on L_n, such that for every global state $g \in L_n$ and path $\lambda \in out^x(g, \sigma_a)$, the path λ is infinite and there exists an index $i \in \mathbb{N}$ such that $M, \lambda[i] \models \phi$.

(ii) From the $(n + 1)$-th iteration, we derive a partial strategy σ'_a defined on $L_{n+1} \setminus L_n$, such that for every global state $g \in L_{n+1} \setminus L_n$ and path $\lambda \in out^x(g, \sigma'_a)$, the path λ is finite and $last(\lambda) \in L_n$.

(iii) Since σ_a and σ'_a are defined on disjoint subsets of locations, they can be combined into a new strategy $\sigma''_a = \sigma_a \cup \sigma'_a$. Consequently, every path $\lambda'' \in out^x(g, \sigma''_a)$ is infinite and consists of the concatenation of a finite prefix $\lambda' \in out^x(g, \sigma'_a)$ with an infinite path $\lambda \in out^x(\hat{g}, \sigma_a)$ for some $\hat{g} \in L_n$. Therefore, there must exist some index $j \in \mathbb{N}$ such that $M, \lambda[j] \models \phi$, which completes the proof that $M, g \models^x \langle\langle i \rangle\rangle \mathrm{F}\, \phi$.

Therefore, by induction, the claim holds for all $n \in \mathbb{N}$. □

Theorem 2 (Approximation of safety)**.** *Let $x \in \{\text{Std}, \text{React}\}$, $i \in \mathcal{A}$, and $l \in St_{M_i}$. If $M_i, l \models^x \nu Z.(\phi \wedge \langle i \rangle Z)$ then, for every $g \in l$, we have $M, g \models^x \langle\langle i \rangle\rangle \mathrm{G}\, \phi$.*

Theorem 2 is a direct consequence of Lemma 2, that uses the *"bounded always"* operator G^n with semantics given by:

$M, g \models^x \langle\langle A \rangle\rangle \mathrm{G}^n \varphi$ iff there is a strategy $s_A \in \Sigma^{ir}_A$ such that, for each path $\lambda \in out^x_M(g, s_A)$, we have $M, \lambda[i] \models^x \varphi$ for all $0 \leq i \geq n$.

Lemma 2. *Let $Sat^x_n(M_i, \nu Z.(\phi \wedge \langle i \rangle Z))$ be the set of states in M_i, computed by the standard greatest fixpoint algorithm for $\nu Z.(\phi \wedge \langle i \rangle Z)$ with at most n fixpoint iterations (equivalently: at most n executions of the preimage for $\langle i \rangle$).*

We claim that, for every $n \in \mathbb{N}$, if $L = Sat^x_n(M_i, \nu Z.(\phi \wedge \langle i \rangle Z))$ then, for every $g \in L$, we have $M, g \models^x \langle\langle i \rangle\rangle \mathrm{G}^n \phi$.

Proof. Proof by induction on n.

Base Case, $n = 0$: If $n = 0$, then $L = Sat^x_0(M_i, \nu Z.(\phi \wedge \langle i \rangle Z))$ contains only the states where ϕ is satisfied. Therefore, for every $g \in St_M$ such that $g^i = l \in L$, we have $M, g \models^x \phi$. Hence, $M, g \models^x \langle\langle i \rangle\rangle \mathrm{G}^0 \phi$.

Inductive Step: Assume that the lemma holds for some $n \geq 0$. We need to show that it also holds for $n + 1$.

Let $L_{n+1} = Sat^x_{n+1}(M_i, \nu Z.(\phi \wedge \langle i \rangle Z))$. By the definition of the greatest fixpoint algorithm, $L_{n+1} \subseteq L_n$, i.e. all the states from L_n from which transition going back to L_n can't be enforced, were removed. Formally: $L_{n+1} = L_n \setminus \{l \mid L_n, \forall_{\alpha \in R_i(l)} \exists_{l' \notin L_n} (l, \alpha, l') \in Tapp_i\} = \{l \mid L_n, \exists_{\alpha \in R_i l} \forall_{l' \in M_i} ((l, \alpha, l') \in Tapp_i) \implies l' \in L_n\}$.

Consider any $g \in St_M$ such that $g^i = l \in L_{n+1}$. We need to show that $\exists_{\alpha \in R_i(l)} \forall_{l' \in M_i} ((l, \alpha, l') \in Tapp_i) \implies l' \in L_n$: by the inductive hypothesis, for every $g' \in St_M$ such that $g'^i = l' \in L_n$, we have $M, g' \models^x \langle\langle i \rangle\rangle \mathrm{G}^n \phi$.

By the induction lemma, we obtain a partial strategy σ_a defined on L_n, such that for every global state $g \in L_n$ and path $\lambda \in out^x(g, \sigma_a)$, the path λ is of length at least n and for all $i \leq n$ we have $M, \lambda[i] \models \phi$.

From the $(n+1)$-th iteration, we derive a partial strategy σ'_a defined on L_{n+1}, such that for every global state $g \in L_{n+1}$ and path $\lambda \in out^x(g, \sigma'_a)$, the path λ is of length at least 1 and for all $i \leq n + 1$ we have $\lambda[i] \in L_{n+1}$, and thus also $\lambda[i] \models \phi$. Since $L_{n+1} \subset L_n$, we have that for all $i \leq n + 1$ we have $\lambda[i] \in L_n$. Therefore, by the inductive hypothesis, we have $M, \lambda[i] \models \phi$ for all $i \leq n + 1$. Hence, we can use σ'_a to demonstrate that $M, g \models^x \langle\langle i \rangle\rangle \mathrm{G}^{n+1} \phi$.

Therefore, by induction, the claim holds for all $n \in \mathbb{N}$. □

The characterization for "until" formulas only slightly extends that of reachability (Theorem 1). We only mention the theorem and the main lemma here, as the rest of the proof is completely analogous.

Theorem 3 (Approximation of until)**.** *Let $x \in \{\text{Std}, \text{React}\}$, $i \in \mathcal{A}$, and $l \in St_{M_i}$. If $M_i, l \models^x \mu Z.(\phi \vee (\psi \wedge \langle i \rangle Z))$ then, for every $g \in l$, we have $M, g \models^x \langle\langle i \rangle\rangle \psi \,\mathrm{U}\, \phi$.*

Lemma 3. *Let $Sat^x_n(M_i, \mu Z.(\phi \vee (\psi \wedge \langle i \rangle Z)))$ be the set of states in M_i, computed by the standard least fixpoint algorithm for $\mu Z.(\phi \vee (\psi \wedge \langle i \rangle Z))$ with at most n fixpoint iterations (equivalently: with at most n executions of the preimage operation for $\langle i \rangle$).*

We claim that, for every $n \in \mathbb{N}$, if $L = Sat^x_n(M_i, \mu Z.(\phi \vee (\psi \wedge \langle i \rangle Z))$ then, for every $g \in L$, we have $M, g \models^x \langle\langle i \rangle\rangle \psi \,\mathrm{U}\, \phi$.

4 Experimental Evaluation

In this section we evaluate the proposed method on the asynchronous simple voting protocol of [Jamroga *et al.*, 2020a].

4.1 Verification

We conducted two sets of experiments to evaluate the performance of our approach. The first set focused on the standard Asynchronous Simple Voting model, while the second set examined the same model with the addition of revoting. In the experiments, we compared the verification time of fixpoint approximation on local approximation models vs. the performance of the fixpoint approximation on standard global model, implemented in the state-of-art model checker STV [Kurpiewski *et al.*, 2024], that operates on explicit representation of global states. We used STV due to its applicability to $\mathbf{ATL_{ir}}$ (i.e., imperfect information strategies), unlike most other model checkers, such as MCMAS and PRISM-GAMES, that focus on perfect-information strategies. The results (discussed further) show astonishing gains. One may notice, however, that in order to run the approximate verification, we should first generate the local approximating model. The code for the experiments is available at https://tinyurl.com/sup7888.

4.2 Generating Approximated Models

The naive approach would be to generate the full global model, and then project it to local states and transitions of a given agent a. To avoid this, we tried a smarter procedure, that translates the existence of a sequence of transitions between two locations l, l' into a **CTL** verification problem, and runs a dedicated **CTL** model checker to get the answer. For the experiments, we employed the UPPAAL model checker as a sub-routine to generate the approximated local model, which was then used with the STV model checker for verification of strategic ability [Behrmann *et al.*, 2004].

All calls to UPPAAL were managed by an auxiliary script that reads the model specification (provided in .xta format with event labels assigned to edges) and constructs a corresponding pair of model and query in a BFS manner, starting from the initial local state. For every candidate triplet (s, α, s')—composed of source local state s, event label α, and target local state s'—the script processes the model.

In the study, each event available to the given agent is associated with an auxiliary Boolean variable, which evaluates to true only when that event was the last to occur at a current (global) state. For each source local state, every outgoing edge with the given event label is duplicated with an additional guard (matching the valuation from the source local state) and update assignments (matching the last occurred

#V	Model generation			Verification		
	Global	Approx. Standard	Approx. Optimized	Global	Approx.	Result
2	0.04	6.60	6.54	<0.01	<0.01	TRUE
3	0.10	6.62	6.60	0.29	<0.01	TRUE
4	1.22	6.93	6.91	30.15	<0.01	TRUE
5	35.80	8.71	8.70	2659	<0.01	TRUE
6	1206	36.95	29.42	timeout	<0.01	TRUE
7	timeout	282.48	280.62	——	<0.01	TRUE
8	timeout	5539	4046	——	<0.01	TRUE
9	timeout			——	——	——

Table 1: Results for Asynchronous Simple Voting

#V	Model generation			Verification		
	Global	Approx. Standard	Approx. Optimized	Global	Approx.	Result
2	0.82	19.43	19.27	8.20	<0.01	TRUE
3	131.61	26.44	19.28	timeout	<0.01	TRUE
4	timeout	524.93	19.25	——	<0.01	TRUE
5	timeout		19.34	——	<0.01	TRUE
6	timeout		19.40	——	<0.01	TRUE
7	timeout		19.41	——	<0.01	TRUE
8	timeout		19.43	——	<0.01	TRUE
9	timeout		19.44	——	<0.01	TRUE

Table 2: Results for Asynchronous Simple Voting with Revoting

event). Fragments outside of the potential predecessors are truncated, and remaining edges are appended with the opposing update assignment. To enforce progressive semantics, all locations are set to the committed type.

The obtained model is then queried for the reachability of the target local state counterpart, with the auxiliary variable from the last occurred event matching the one from the triplet. Next, the computed local states are verified for livelocks [Ashcroft, 1975]—the existence of a cycle of global states and events that neither leads to a change in the given local state nor involves events from the studied agent's repertoire—using the original model specification.

The straightforward application of UPPAAL for generating the approximated local model was inefficient. When the query with the triplet has no counterpart in the global model, the verifier needs to generate the entire state space to conclude that. However, we implemented several optimizations to improve efficiency. Firstly, we noted the symmetry within the homogeneous modules of voters; none of the voters have transitive interactions with each other—they only interact with the coercer. This semantic analysis allowed us to conclude that the order in which different voters perform their actions does not affect the composition of their local models. Consequently, we enhanced the generation of the local model by using UPPAAL's specific process priority feature, assigning the given voter the highest priority among all other voters, and adding an edge connecting the target local state with an auxiliary sink location, which has a single self-loop.

4.3 Results

The results of the experiments are summarized in Tables 1 and 2. The first column indicates the number of voters considered. In all scenarios, there were two candidates and one coercer. The tables present the model generation times for the standard global model and two approximated models: one without optimizations (standard) and one with optimizations (optimized) in UPPAAL. Verification was performed using a fixpoint algorithm, and the verified formula was $\phi_1 = \langle\!\langle Voter_1 \rangle\!\rangle \mathrm{F}\,(vote_{1,1} \wedge \neg give_1)$. The approximated model was generated for agent $Voter_1$. All times are reported in seconds, with a timeout set to two hours.

All experiments were conducted on a machine equipped with a 3.0 GHz 8-core AMD Ryzen 7 5700X3D CPU and 64 GB of RAM. The model generation and verification times were measured using the `time` command in a Linux environment to ensure precise and consistent timing measurements.

As observed, the model generation times for few voters are lower for the global model compared to the approximate models. However, this trend reverses as the number of voters increases. The optimized version of the approximate model is consistently faster than the standard version, although its time still increases exponentially for the model without revoting. In the scenario with revoting, the optimized approach exhibits linear time growth, allowing us to generate a model for even 50 voters in under one minute. The verification time for the global model also increases exponentially, whereas the verification time for the approximate model remains constant. This is because the size of the approximated model does not change, as the local model of the voter remains the same regardless of the number of voters in the system.

Interestingly, the optimized version yields a more substantial reduction in model generation time for the case with revoting. Without revoting, the exponential growth in time can be mainly attributed to the detection of potential livelocks. In the absence of a witnessing livelock cycle, the model checker must generate the entire fragment of the source local state's predecessor states to reach a conclusion.

Note that in both the straightforward and optimized versions, there is room for further improvement in efficiency through parallel computation of the reachable successors from the already discovered states of the local model. This would allow more efficient generation of the approximated model, especially for larger numbers of voters.

5 Conclusions

The model checking problem for $\mathbf{ATL_{ir}}$ is indeed challenging. Nevertheless, some feasible approaches for tackling this problem exist. We have taken the first step in optimizing fixpoint approximations, by focusing on the local model of the agent whose strategic abilities we aim to verify. In consequence, the verified model grows much slower, and in some cases might even remain constant, regardless of the number of agents in the scenario. Consequently, the verification time also decreases. However, the approximate model needs to be generated. We have shown that existing tools like UPPAAL can be used for this task, and with proper optimizations, the generation times for the approximated model can increase linearly rather than exponentially.

The present method has several limitations: observable safety/reachability goals, only individual strategies, no nesting of strategic modalities. They arise from technical feasibility (e.g., scalability and semantic alignment with experiments), rather than theoretical constraints. Our experiments show very promising results, and we plan to extend the method to handle general non-observable properties, proper coalitions, and nested strategic reasoning in future work.

Acknowledgments

The work has been supported by NCBR Poland and FNR Luxembourg under the PolLux/FNR-CORE project SpaceVote (POLLUX-XI/14/SpaceVote/2023 and C22/IS/17232062/SpaceVote). For the purpose of open access, and in fulfilment of the grant agreement, the authors have applied CC BY 4.0 license to any Author Accepted Manuscript version arising from this submission.

References

[Alur *et al.*, 1998] Rajeev Alur, Thomas Henzinger, Freddy Y.C. Mang, Shaz Qadeer, Sriram Rajamani, and Serdar Tasiran. MOCHA: Modularity in model checking. In *Proceedings of Computer Aided Verification (CAV)*, volume 1427 of *Lecture Notes in Computer Science*, pages 521–525. Springer, 1998.

[Alur *et al.*, 2001] R. Alur, L. de Alfaro, R. Grossu, T.A. Henzinger, M. Kang, C.M. Kirsch, R. Majumdar, F.Y.C. Mang, and B.-Y. Wang. jMocha: A model-checking tool that exploits design structure. In *Proceedings of International Conference on Software Engineering (ICSE)*, pages 835–836. IEEE Computer Society Press, 2001.

[Alur *et al.*, 2002] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[Ashcroft, 1975] Edward A Ashcroft. Proving assertions about parallel programs. *Journal of Computer and System Sciences*, 10(1):110–135, 1975.

[Behrmann *et al.*, 2004] G. Behrmann, A. David, and K.G. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-Time Systems: SFM-RT*, number 3185 in LNCS, pages 200–236. Springer, 2004.

[Belardinelli *et al.*, 2017] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Verification of broadcasting multi-agent systems against an epistemic strategy logic. In *Proceedings of IJCAI*, pages 91–97, 2017.

[Berthon *et al.*, 2017] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi. Strategy logic with imperfect information. In *Proceedings of LICS*, pages 1–12, 2017.

[Bulling and Jamroga, 2011] N. Bulling and W. Jamroga. Alternating epistemic mu-calculus. In *Proceedings of IJCAI-11*, pages 109–114, 2011.

[Bulling *et al.*, 2010] N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.

[Busard *et al.*, 2014] Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. Improving the model checking of strategies under partial observability and fairness constraints. In *Formal Methods and Software Engineering*, volume 8829 of *Lecture Notes in Computer Science*, pages 27–42. Springer, 2014.

[Busard *et al.*, 2015] Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. *Information and Computation*, 242:128–156, 2015.

[Cermak *et al.*, 2014] Petr Cermak, Alessio Lomuscio, Fabio Mogavero, and Aniello Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Proc. of Computer Aided Verification (CAV)*, volume 8559 of *Lecture Notes in Computer Science*, pages 525–532. Springer, 2014.

[Cermák *et al.*, 2015] Petr Cermák, Alessio Lomuscio, and Aniello Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *Proceedings of AAAI*, pages 2038–2044, 2015.

[Chen *et al.*, 2013] Taolue Chen, Vojtech Forejt, Marta Kwiatkowska, David Parker, and Aistis Simaitis. PRISM-games: A model checker for stochastic multi-player games. In *Proceedings of Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 7795 of *Lecture Notes in Computer Science*, pages 185–191. Springer, 2013.

[Dima and Tiplea, 2011] Catalin Dima and Ferucio L. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[Guelev *et al.*, 2011] Dimitar P. Guelev, Catalin Dima, and Constantin Enea. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.

[Huang and van der Meyden, 2014] Xiaowei Huang and Ron van der Meyden. Symbolic model checking epistemic strategy logic. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 1426–1432, 2014.

[Jamroga *et al.*, 2019] Wojciech Jamroga, Michał Knapik, Damian Kurpiewski, and Łukasz Mikulski. Approximate verification of strategic abilities under imperfect information. *Artificial Intelligence*, 277, 2019.

[Jamroga *et al.*, 2020a] W. Jamroga, W. Penczek, T. Sidoruk, P. Dembiński, and A. Mazurkiewicz. Towards partial order reductions for strategic ability. *Journal of Artificial Intelligence Research*, 68:817–850, 2020.

[Jamroga *et al.*, 2020b] Wojciech Jamroga, Yan Kim, Damian Kurpiewski, and Peter Y. A. Ryan. Towards model checking of voting protocols in uppaal. In *Proceedings of E-Vote-ID*, volume 12455 of *Lecture Notes in Computer Science*, pages 129–146. Springer, 2020.

[Jamroga *et al.*, 2021] Wojciech Jamroga, Wojciech Penczek, and Teofil Sidoruk. Strategic abilities of asynchronous agents: Semantic side effects and how to tame them. In *Proceedings of KR 2021*, pages 368–378, 2021.

[Kacprzak and Penczek, 2004] M. Kacprzak and W. Penczek. Unbounded model checking for alternating-time temporal logic. In *Proceedings of International*

Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 646–653. IEEE Computer Society, 2004.

[Kaminski *et al.*, 2024] Mateusz Kaminski, Damian Kurpiewski, and Wojciech Jamroga. STV+KH: towards practical verification of strategic ability for knowledge and information flow. In *Proceedings of AAMAS*, pages 2812–2814. ACM, 2024.

[Kim *et al.*, 2022] Yan Kim, Wojciech Jamroga, and Peter Y.A. Ryan. Verification of the socio-technical aspects of voting: The case of the Polish postal vote 2020. In *Proceedings of STAST*, 2022. To appear, available at https://arxiv.org/abs/2210.10694.

[Kurpiewski *et al.*, 2019] Damian Kurpiewski, Wojciech Jamroga, and Michał Knapik. STV: Model checking for strategies under imperfect information. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019*, pages 2372–2374. IFAAMAS, 2019.

[Kurpiewski *et al.*, 2021] Damian Kurpiewski, Witold Pazderski, Wojciech Jamroga, and Yan Kim. STV+Reductions: Towards practical verification of strategic ability using model reductions. In *Proceedings of AAMAS*, pages 1770–1772. ACM, 2021.

[Kurpiewski *et al.*, 2022] Damian Kurpiewski, Wojciech Jamroga, Lukasz Masko, Lukasz Mikulski, Witold Pazderski, Wojciech Penczek, and Teofil Sidoruk. Verification of multi-agent properties in electronic voting: A case study. In *Advances in Modal Logic*, pages 531–556. College Publications, 2022.

[Kurpiewski *et al.*, 2023] Damian Kurpiewski, Wojciech Jamroga, and Teofil Sidoruk. Towards modelling and verification of social explainable AI. In *Proceedings of ICAART*, pages 396–403, 2023.

[Kurpiewski *et al.*, 2024] Damian Kurpiewski, Mateusz Kamiński, Yan Kim, Łukasz Maśko, Witold Pazderski, Wojciech Jamroga, and Łukasz Mikulski. STV – StraTegic Verifier. code repository, 2024. https://github.com/blackbat13/stv_v2.

[Lomuscio and Raimondi, 2006] A. Lomuscio and F. Raimondi. MCMAS : A model checker for multi-agent systems. In *Proceedings of Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 4314 of *Lecture Notes in Computer Science*, pages 450–454. Springer, 2006.

[Lomuscio *et al.*, 2017] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 19(1):9–30, 2017.

[Mogavero *et al.*, 2010] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning about strategies. In *Proceedings of FSTTCS*, pages 133–144, 2010.

[Mogavero *et al.*, 2014] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic*, 15(4):1–42, 2014.

[Pilecki *et al.*, 2014] J. Pilecki, M.A. Bednarczyk, and W. Jamroga. Synthesis and verification of uniform strategies for multi-agent systems. In *Proceedings of CLIMA XV*, volume 8624 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2014.

[Schobbens, 2004] Pierre-Yves Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.