Towards Partial Order Reductions for Strategic Ability

Wojciech Jamroga and Wojciech Penczek
Institute of Computer Science, Polish Academy of Sciences
01-248 Warsaw, Poland
{jamroga,penczek}@ipipan.waw.pl

Piotr Dembiński and Antoni Mazurkiewicz
Institute of Computer Science, Polish Academy of Sciences
01-248 Warsaw, Poland
{piotr.damaz}@ipipan.waw.pl

ABSTRACT

We propose a general semantics for strategic abilities of agents in asynchronous systems, with and without perfect information. Based on the semantics, we show some general complexity results for verification of strategic abilities in asynchronous interaction. More importantly, we develop a methodology for partial order reduction in verification of agents with imperfect information. We show that the reduction preserves an important subset of strategic properties, both with and without the fairness assumption. Interestingly, the reduction does not work for strategic abilities under perfect information.

KEYWORDS

Alternating-time temporal logic; asynchronous systems; model checking; partial order reduction; Mazurkiewicz traces

1 INTRODUCTION

Alternating-time temporal logic ATL and its fragment ATL [5, 6] extend temporal logic with the notion of strategic ability. They allow to express statements about what agents (or groups of agents) can achieve. For example, ⟨⟨(i)Fwin⟩⟩i says that agent i can eventually win no matter what the other agents do, while ⟨⟨(i,j)Gsafe⟩⟩i,j expresses that agents i and j together can force the system to always remain in a safe state. Such properties can be useful for specification, verification, and reasoning about interaction in agent systems. Moreover, algorithms and tools for verification of strategic abilities have been in constant development for almost 20 years [2, 3, 9, 10, 16, 18, 19, 21, 36, 41, 50, 51, 63]. However, there are two caveats.

First, many tools and algorithmic solutions focus on agents with perfect information. This is clearly unrealistic in all but the simplest multi-agent scenarios. Still, the tendency is somewhat easy to understand, since model checking of ATL variants with imperfect information is $\Delta^2_P$ to PSPACE-complete for agents playing memoryless strategies [13, 38, 66] and EXPTIME-complete to undecidable for agents with perfect recall of the past [11, 28, 34]. Moreover, the imperfect information semantics of ATL does not admit alternation-free fixpoint characterizations [14, 26, 27], which makes incremental synthesis of strategies difficult to achieve [16, 17, 36, 39, 63].

Secondly, the semantics of strategic logics are almost exclusively based on synchronous concurrent game models. That is, one implicitly assumes the existence of a global clock that triggers subsequent global events in the system. At each tick of the clock, all the agents choose their actions, and the system proceeds accordingly with the corresponding global transition. However, many real-life systems are inherently asynchronous. No less importantly, many systems that are synchronous at the implementation level can be more conveniently modeled as asynchronous on a more abstract level.

In this paper, we make the first step towards strategic analysis of such systems. Our contribution is threefold. First, we define a semantics of strategic abilities for agents in asynchronous systems, with and without perfect information. Secondly, we present some general complexity results for verification of strategic abilities in such systems. Thirdly, and most importantly, we adapt partial order reduction (POR) to model checking of strategic abilities for agents with imperfect information. We also demonstrate that POR allows to significantly reduce the size of the model, and thus to make the verification more feasible. In fact, we show that the most efficient variant of POR, defined for linear time logic LTL, can applied be almost directly. The (nontrivial) proof that the LTL reductions work also for the more expressive strategic operators is the main contribution of this paper. Interestingly, the scheme does not work for verification of agents with perfect information.

The outline of the paper is as follows. In Section 2, we introduce the structures to represent and reason about asynchronous multi-agent systems. In Section 3, we define the semantics of ATL for asynchronous systems. In Section 4, we show the general complexity results. Sections 5 and 6 put forward the theoretical foundations and the algorithms for partial order reduction. We conclude in Section 8.

Related work. Relevant related work is relatively scarce. Asynchronous semantics and partial order reduction for distributed systems were extensively studied in [31–33, 43, 45, 57–60, 62]. The most recent approaches include dynamic POR [1, 20, 30] and combine POR with symbolic methods [42, 44]. The only efficient approach to partial order reduction in a MAS context [48, 49] concerns standard temporal-epistemic logics.

Alur, Henzinger and Kupferman mentioned asynchronous systems in their seminal paper on ATL [6], but they modeled them as a special case of synchronous systems. Asynchronous omega-regular games were also considered in [64]. Reactive modules [2–4] feature several modes of asynchronous execution, but – to the best of our knowledge – this aspect has never been given a more systematic analysis. The work that comes closest to our new proposal is [25]
2 MODELS OF MULTI-AGENT SYSTEMS

One can model multi-agent systems as networks of automata that execute asynchronously by interleaving local transitions, and synchronize their moves whenever a shared action is executed [29, 48].

Definition 2.1 (Asynchronous MAS). An asynchronous multi-agent system (AMAS) consists of n agents \( \mathcal{A} = \{ 1, \ldots, n \} \), each associated with a tuple \( A_i = (L_i, I_i, Act_i, P_i, T_i) \) including a set of local states \( L_i = \{ l_1^i, l_2^i, \ldots, l_{|L_i|}^i \} \), an initial state \( I_i \subseteq L_i \), and a set of actions \( Act_i = \{ a_1^i, a_2^i, \ldots, a_{|Act_i|}^i \} \). Notice that the sets \( Act_i \) do not need to be disjoint. \( Act = \bigcup_{i \in \mathcal{A}} Act_i \) is the set of all actions, and \( Loc = \bigcup_{i \in \mathcal{A}} L_i \) is the set of all local states in the system. For each \( a \in Act \), the set \( Agent(a) = \{ i \in \mathcal{A} \mid a \in Act_i \} \) contains the agents which have \( a \) in their sets of actions.

A local protocol \( P_i : L_i \rightarrow 2^{Act_i} \) selects the actions available at each local state. Moreover, \( T_i : L_i \times Act_i \rightarrow L_i \) is a (partial) local transition function such that \( T_i(l_i, a) \neq \text{undef} \) iff \( a \in P_i(l_i) \).

Example 2.2 (TGC). Figure 1 presents the Train-Gate-Controller (TGC) benchmark [3, 35]. The system consists of three agents: a controller \( c \) and two trains \( t_1, t_2 \). The trains run on separate circular tracks that jointly pass through a narrow tunnel. Each train can be waiting for the permission to enter (state \( W \)) with a tuple \( (TGC) \) benchmark \( \mathcal{T} \). To understand the interaction between asynchronous agents, we use global states and global transitions, defined formally below.

Definition 2.3 (Interleaved Interpreted System). Let \( \mathcal{P}V \) be a set of propositional variables. An interleaved interpreted system (IIS), or a model, is an asynchronous MAS extended with the following elements: a set \( St \subseteq L_1 \times \cdots \times L_n \) of global states, an initial state \( s \in St \), a global transition function \( T : St \times Act \rightarrow St \), and a valuation of propositions \( V : St \rightarrow 2^{\mathcal{P}V} \). For state \( s = (l_1, \ldots, l_n) \), we denote the local component of agent \( i \) by \( g_i = l_i \). Also, we will sometimes write \( g_1 \xrightarrow{a} g_2 \) instead of \( T(g_1, a) = g_2 \).

We say that action \( a \in Act \) is enabled at \( s \in St \) if \( g \xrightarrow{a} g' \) for some \( g' \in St \). The global transition function is assumed to be total, i.e., at each \( g \in St \) there exists at least one enabled action.

2.1 Interleaved Interpreted Systems

To understand the interaction between asynchronous agents, we use global states and global transitions, defined formally below.

An infinite sequence of global states and actions \( \pi = g_0a_0g_1a_1g_2 \ldots \) is called a path if \( g_i \xrightarrow{a} g_{i+1} \) for every \( i \geq 0 \). \( Act(\pi) = a_0a_1a_2 \ldots \) is the sequence of actions in \( \pi \), and \( \pi[i] = g_i \) is the \( i \)-th global state of \( \pi \). \( \Pi_M(g) \) denotes the set of all paths in \( M \) starting at \( g \).

IIS can be used to provide an execution semantics to AMAS.

Definition 2.4 (Canonical IIS). Let \( S \) be an asynchronous MAS with \( n \) agents. Its canonical model \( IIS_T(S) \) extends \( S \) with the valuation \( V \), global states \( St = L_1 \times \cdots \times L_n \), initial state \( s = (l_1, \ldots, l_n) \), and transition function \( T \) defined as follows: \( T(g_1, a) = g_2 \) iff \( T(g_1', a) = g_2' \) for all \( i \in Agent(a) \) and \( g_1' = g_2' \) for all \( i \in S \setminus Agent(a) \).

The state/transition structure of the canonical interleaved interpreted system for TGC is depicted in Figure 2a. Additionally, let us assume \( \mathcal{P}V = \{ \text{in}1, \text{in}2 \} \) with \( \text{in}1 \in V(g) \) iff \( g = T \). That is, proposition \( \text{in}1 \) denotes that train \( t_1 \) is currently in the tunnel. It is easy to see that the global state space grows exponentially with the number of agents. In some cases, it suffices to consider a subset of states and transitions, i.e., a submodel of \( IIS(S) \).

Definition 2.5 (Submodel). Let \( M, M' \) be two models extending the same AMAS, such that \( St' \subseteq St \), \( s \in St' \), \( T \) is an extension of \( T' \), and \( V' = V|_{St'} \). Then, we write \( M' \subseteq M \) and call \( M' \) a submodel of \( M \) or a reduced model of \( M \).

An example submodel of the IIS for TGC is shown in Figure 2b. It is easy to see that, for each \( g \in St' \), we have \( \Pi_M(g) \subseteq \Pi_{M'}(g) \).

In order to generate reduced models, we need a notion of invisibility and independency of actions. Intuitively, an action is invisible iff it does not change the valuations of the propositions. Note that this concept of invisibility is technical, and is not connected to the view of any agent in the sense of [52]. Additionally, we can designate a subset of agents \( A \) whose actions are visible by definition. Furthermore, two actions are weakly independent iff they are not actions of the same agent, and strongly independent iff they are weakly independent and at least one of them is invisible.

Definition 2.6 (Invisible actions). Consider a model \( M \), a subset of agents \( A \subseteq \mathcal{A} \), and a subset of propositions \( PV \subseteq \mathcal{P}V \). An action \( a \in Act \) is invisible wrt. \( A \) and \( PV \) if \( Agent(a) \cap A = \emptyset \) and for each two global states \( g, g' \in St \) we have that \( g \xrightarrow{a} g' \) implies...
V(g) \cap PV = V(g') \cap PV$. The set of all invisible actions for $A, PV$ is denoted by $Inv_{IA, PV}$, and its closure $- \circ$ of visible actions $- \circ$ by $Vis_{IA, PV} = Act \backslash Inv_{IA, PV}$.

**Definition 2.7 (Independent actions).** Weak independence $WI \subseteq Act \times Act$ is defined as: $WI = \{(a, b) \in Act \times Act | Agent(a) \cap Agent(b) = \emptyset\}$. Strong independence (or independence) $IA_{PV} \subseteq Act \times Act$ is defined as: $IA_{PV} = WI \setminus (Vis_{IA, PV} \times Vis_{IA, PV})$.

We assume in the rest of the paper that a suitable subset $PV$ is given, and omit the subscript $PV$ whenever clear from the context.

### 3 REASONING ABOUT AGENTS’ ABILITIES

Many important properties in a MAS can be specified in terms of the strategic ability of some agents to achieve a given goal. Such properties can be specified by formulas of the strategic logic ATL. The semantics of ATL is typically defined for models of synchronous systems. In this section, we show how it can be adapted to asynchronous MAS.

#### 3.1 Alternating-Time Temporal Logic: Syntax

*Alternating-time temporal logic* [5, 6] generalizes the branching-time temporal logic CTL [22] by replacing the path quantifiers $E, A$ with strategic modalities $\langle\rangle$. Informally, $\langle\rangle\gamma$ expresses that the group of agents $A$ has a collective strategy to enforce the temporal property $\gamma$. The formulas make use of temporal operators: "$X$" ("next"), "$G$" ("always from now on"), "$F$" ("now or sometime in the future"), $U$ ("strong until"), and $R$ ("release"). The logic comes in several syntactic variants, the most popular of which are ATL* and "vanilla ATL" (the latter often called simply "ATL").

**Definition 3.1 (Syntax of ATL*).** Let $P^V$ be a set of propositional variables and $A$ the set of all agents. The language of ATL* is defined by the following grammar (where $p \in P^V$ and $A \subseteq \mathcal{A}$):

- $\varphi ::= p | \neg \varphi | \varphi \land \varphi | \langle\rangle \gamma$
- $\gamma ::= \gamma | \gamma \land \gamma | \langle\rangle \gamma | \langle\rangle \gamma \varphi | \langle\rangle \langle\rangle \varphi \land \varphi \land \varphi R \varphi$

The other Boolean operators are defined as usual. "Release" can be defined as $y_1 R y_2 \equiv \neg((\neg y_1) U (\neg y_2))$. The "sometime" and "always" operators can be defined as $F \equiv \text{true} U \gamma$ and $G \equiv \neg F \gamma$. Moreover, "for all paths" can be defined as $A \gamma \equiv \langle\rangle \gamma$.

**Definition 3.2 (Syntax of ATL).** In "vanilla ATL", every occurrence of a strategic modality is immediately followed by a temporal operator. In that case, "release" is not definable from "until" anymore [47], and it must be added explicitly to the syntax:

- $\varphi ::= p | \neg \varphi | \varphi \land \varphi | \langle\rangle X \langle\rangle \gamma | \langle\rangle \langle\rangle \gamma \varphi | \langle\rangle \langle\rangle \gamma \varphi \varphi | \langle\rangle \langle\rangle \varphi \land \varphi \land \varphi R \varphi$

In the rest of the paper, we are mainly interested in formulas that do not use the next step operator $X$, and do not contain nested strategic modalities. We denote the corresponding subsets of ATL* and ATL by sATL* ("simple ATL*") and sATL ("simple ATL"). Moreover, $\langle\rangle$ is the fragment of sATL* that admits only formulas consisting of a single strategic modality followed by an LTL formula (i.e., $\langle\rangle \gamma$, where $\gamma \in LTL$), and analogously for $\langle\rangle$.

**Example 3.3.** The following formulas of sATL* specify interesting properties of the TCG system: $\langle\rangle F in_1$ (the controller can let train $t_1$ in), $\langle\rangle G \neg in_1$ (the controller can keep $t_1$ out forever), $\langle\rangle F (in_1 \land F \neg in_1)$ (the controller can let $t_1$ through, $\neg (\langle\rangle t_1, t_2) F (in_1 \lor in_2)$ (neither train can get in without the help of the controller, even if it collaborates with the other train).

We claim that most of practically interesting specifications of strategic ability can be expressed in sATL, possibly extended with epistemic operators. Nested strategic modalities allow to express an agent’s ability to endow another agent with ability (or deprive the other agent of ability), which is seldom of practical interest.

#### 3.2 Strategies and Outcomes

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. Here, we follow Schobbens [66], and adopt his taxonomy of four "canonical" strategy types: $Ir, Ir$, $Ir$, and $Ir$. In the notation, $r$ stands for perfect (resp. imperfect) recall, and $I$ (resp. $i$) refers to perfect (resp. imperfect) information. Note that verification of ATL for agents with perfect recall is in general undecidable [28]. Because of that, we focus on memoryless strategies. Formally:

- A memoryless perfect information strategy for agent $i$ is a function $\sigma_i : St \rightarrow Act_i$ s.t. $\sigma_i(g) \in P_i(g')$ for each $g' \in St$.
- A memoryless imperfect information strategy for $i$ is a function $\sigma_i : L_i \rightarrow Act_i$ s.t. $\sigma_i(l) \in Q_i(l)$ for each $l \in L_i$.

Thus, a perfect information strategy can assign different actions to any two global states, while under imperfect information the agent’s choices depend only on the local state of the agent. A joint strategy $\sigma$ for a coalition $A \subseteq \mathcal{A}$ is a tuple of strategies, one per agent $i \in A$. We denote the set of $A$’s collective memoryless perfect (resp. imperfect) information strategies by $\Sigma_A^P$ (resp. $\Sigma_A^I$). Additionally, let $\sigma_A = (\sigma_1, \ldots, \sigma_k)$ be a joint strategy for $A = \{i_1, \ldots, i_k\}$. For each $g \in St$, we define $\sigma_A(g) = (\sigma_1(g), \ldots, \sigma_k(g))$.

**Definition 3.4 (Outcome paths).** Let $Y \in \{Ir, Ir\}$. The outcome of strategy $\sigma_A \in \Sigma_A^Y$ in state $g \in St$ is the set $out_M(g, \sigma_A) \subseteq \Pi_M(g)$ such that $\pi = \gamma_0 \gamma_1 \gamma_2 \cdots \in out_M(g, \sigma_A)$ iff $\gamma_0 = g$ and $\forall i \in N \\forall j \in Y \ i \in A$ if $j \in Agent(ai)$, then $a_i \in \sigma_i(\pi(l))$ for $Y = Ir$, and $a_i \in \sigma_i(\pi(l'))$ for $Y = Ir$.

Intuitively, the outcome of a joint strategy $\sigma_A$ in a global state $g$ is the set of all the infinite paths that can occur when in each state of the paths either some agents (an agent) in $A$ execute(s) an action according to $\sigma_A$ or some agents (an agent) in $A$ execute(s) an action following their protocols. Clearly, each action $a$ has to be executed by all agents which have $a$ in their sets of actions. In reasoning about asynchronous systems, one often wants to look only at fair paths, i.e., ones that do not consistently ignore an agent whose action is always enabled. Formally, a path $\pi$ satisfies the *concurrency-fairness condition* (CF) if there is no action enabled in all states of $\pi$ from $\pi[1]$ on, and at the same time weakly independent from all the actions actually executed in $\pi[1], \pi[1+1], \pi[1+2], \ldots$. We denote the set of all such paths starting at $g$ by $\Pi_M^{CF}(g)$.

**Definition 3.5 (CF-outcome).** The concurrency-fair outcome of $\sigma_A \in \Sigma_A^Y$ is defined as $out_M^{CF}(g, \sigma_A) = out_M(g, \sigma_A) \cap \Pi_M^{CF}(g)$.

Note that, in an arbitrary IIS, not every action admitted by agent $i$’s protocol at some local state $l_i$ must be enabled at all the global

---

1. Alternatively, we can require the agent’s choices to be the same for the global states that share the local states.
states \( y \) with \( g' = l_i \). Moreover, it may be the case that the outcome of a strategy is empty, because all the paths consistent with the strategy get “stuck” at some global state. In Sections 5 and 6, we show how to construct reduced models so that this does not create problems. For the moment, we state the following two lemmas.

**Lemma 3.6.** Let \( M' \) be a submodel of \( M \). For each \( l_i \)-joint strategy \( \sigma_A \) we have \( \text{out}_{M'}(l_i, \sigma_A) = \text{out}_{M}(l_i, \sigma_A) \cap \Pi_{M'}(l_i) \), and \( \text{out}_{M'}^C(l_i, \sigma_A) = \text{out}_{M}^C(l_i, \sigma_A) \cap \Pi_{M'}^C(l_i) \).

Proof. Notice that each \( l_i \)-strategy in \( M \) is also a well defined \( l_i \)-strategy in \( M' \) as it is defined on the local states of AMAS which is extended by \( M \) and \( M' \). The lemma follows directly from Definitions 3.4 and 3.5, together with the fact that \( \Pi_{M'}(l_i) \subseteq \Pi_{M}(l_i) \). \( \square \)

**Lemma 3.7.** Let \( M \) be a model, \( \pi, \pi' \in \Pi_{M}(l_i) \), and for some \( i \in \mathcal{A} : \text{Act}(\pi) |_{\text{Act}_i} = \text{Act}(\pi') |_{\text{Act}_i} \). Then, for each \( l_i \)-strategy \( \sigma_i \), we have \( \pi \in \text{out}_{M}(l_i, \sigma_i) \Leftrightarrow \pi' \in \text{out}_{M}(l_i, \sigma_i) \).

Proof sketch. Let \( \text{Act}(\pi) |_{\text{Act}_i} = b_0 b_1 \ldots \) For each \( b_j \) let \( \pi[b_j] \) denote the global state from which \( b_j \) is executed in \( \pi \). By induction we can show that for each \( j \geq 0 \) we have \( \pi[b_j]^i = \pi'[b_j]^i \). For \( j = 0 \) it is easy to notice that \( \pi[b_0]^i = \pi'[b_0]^i = \iota_i \).

Assume that the thesis holds for \( j = k \). The induction step follows from the fact the local evolution \( T_i \) is a function, so if \( \pi[b_k]^i = \pi'[b_k]^i = l \) for some \( l \in L_i \), then \( \pi[b_{k+1}]^i = \pi'[b_{k+1}]^i = T_i(l, b_k) \). This means that for each \( l_i \)-strategy \( \sigma_i \), we have \( \pi \in \text{out}_{M}(l_i, \sigma_i) \Leftrightarrow \pi' \in \text{out}_{M}(l_i, \sigma_i) \), which concludes the proof.

The lemma can be easily generalized to joint strategies \( \sigma_A \in \Sigma_{ir}^A \).

Note that the same property does not hold for perfect information strategies. This is because the current local state \( l_i \) can only change through the execution of an action by agent \( i \), but the current global state can possibly change because of another agent’s transition. Similarly, the analogue of Lemma 3.7 does not hold in synchronous models of MAS, since the local transitions of \( i \) in a synchronous model can be influenced by the actions selected by the other agents.

### 3.3 Asynchronous Semantics of ATL and \( \text{sATL}^* \)

Our semantics of \( \text{sATL}^* \) for asynchronous interaction, parameterised with the strategy type \( Y \in \{ \text{ir}, \text{ir} \} \), is defined as follows:

- \( M, g \models_Y \psi \Leftrightarrow \text{for } p \in V(g), \text{ for } p \in \mathcal{P} V \); \( \text{if } g \models \psi \); \( \text{if } g \not\models \psi \);
- \( M, g \models_Y \phi_1 \land \phi_2 \Leftrightarrow M, g \models_Y \phi_1 \text{ and } M, g \models_Y \phi_2 \);
- \( M, g \models_Y (\langle A \rangle)Y \Leftrightarrow \text{there is a strategy } \sigma_A \in \Sigma_{ir}^A \text{ such that } \text{out}_{M}(g, \sigma_A) \text{ is complete in the size of the model and the length of the formula. It immediately follows by the size of the model and the length of the formula. It remains NP-complete even for formulas of bounded length.}

#### 4 Model Checking \( \text{iATL}^*_{ir} \) and \( \text{iATL}^*_{irf} \)

We begin by looking at the verification complexity for simplest specifications, consisting of a single strategic modality \( \langle A \rangle \) immediately followed by a single temporal modality.

**Proposition 4.1.** Model checking \( \text{iATL}^*_{ir} \) and \( \text{iATL}^*_{irf} \) is NP-complete in the size of the model and the length of the formula. It remains NP-complete even for formulas of bounded length.
Proof sketch. Analogous to the result in [66] for $\langle \Gamma \rangle$-ATL$_{ir}$.

For the upper bound, observe that model checking of $\langle A \rangle \gamma$ in $M, g$ can be done by (1) guessing an $r$-joint strategy $\sigma_A$ (2) pruning $M$ according to $\sigma_A$, and (3) model checking the CTL formulas: $\neg AGL$ ("the set of paths is nonempty") and $A\gamma$ ("for all paths, $\gamma$") in state $g$ of the resulting model. Since $\sigma_A$ is of at most linear size with respect to $|M|$, and model checking of $A\gamma$ can be done in deterministic polynomial time w.r.t. $|M|$, both with and without fairness assumptions [7], we obtain the bound.

For the lower bound, we use the reduction of [66] of SAT to model checking of the formula $\langle (1) \rangle \forall \gamma$ in a single-agent model (note that single-agent systems can be seen as special cases of both synchronous and asynchronous systems, and the semantics with and without fairness assumptions coincide on such models). Notice that the lower bound does not rely on the length of the formula.

Proposition 4.2. Model checking $\exists$ATL$_{ir}$ and $\exists$ATL$_{irF}$ is PSPACE-complete in the size of the representation (even for formulas of bounded length).

Proof sketch. For the upper bound, observe that model checking of $\langle A \rangle \gamma$ in an AMAS $S$ can be done by: (1) guessing an ir strategy $\sigma_A$ as a deterministic restriction of the protocols $P_i$, $i \in \mathcal{A}$; (2) pruning $M$; (3) model checking, in the resulting representation $S'$, the LTL formulas: (3a) GL and (3b) $\gamma$ (for $\exists$ATL$_{ir}$) or $\gamma \wedge \text{fair}$ (for $\exists$ATL$_{irF}$) where fair is an LTL characterization of fairness. The algorithm returns true iff the output of (3a) was false and that of (3b) was true. Since the size of $\sigma_A$ is linear wrt $|S|$, and model checking LTL is in PSPACE wrt $|S|$ [65], we obtain the bound.

For the lower bound, we adapt the construction from [46]. Given a Turing machine $T$ with space complexity $s(n)$, we construct the concurrent program $P(T)$ as in [46, Theorem 6.1]. According to that theorem, there exists a computation of $T$ on the empty tape which eventually reaches an accepting state iff $P(T) \models \text{EF}$. Now, we observe that $P(T)$ is in fact an asynchronous MAS in the sense of Definition 2.1. Thus, $P(T) \models \text{EF}$ accept iff $IIS(P(T)) \not\models \gamma (\emptyset) \text{G-accept}$. This way we obtain the co-PSPACE-hardness for $\exists$ATL$_{ir}$ (recall that co-PSPACE = PSPACE).

For $\exists$ATL$_{irF}$, we observe that all the paths in $IIS(P(T))$ are fair, so the same construction can be used.

Again, the reduction does not rely on the length of the formula.

\[\square\]

4.2 Model Checking sATL$_{ir}$ and sATL$_{irF}$

The verification complexity for Boolean combinations of formulas from $\exists$ATL is almost the same.

Proposition 4.3. Model checking $\exists$ATL$_{ir}$ and $\exists$ATL$_{irF}$ is NP-hard and in $\Theta^P_2$ in the size of the model and the length of the formula (even for formulas of bounded length).\footnote{Where $\Theta^P_2 = \text{P}^{\text{NP}}$ is the class of problems solvable by a deterministic polynomial-time Turing machine making polynomially many nonadaptive calls to an NP oracle.}

Proof sketch. The lower bounds follow from Proposition 4.1. The following algorithm for checking $\varphi$ in $M, g$ demonstrates the upper bound. First, the non-deterministic algorithm in Proposition 4.1 is used as an oracle that determines the truth value for each subformula $\langle A \gamma \rangle \varphi$ of $\varphi$. Clearly, the oracle is called at most $|\varphi|$ times, and the input in the next call does not depend on the output of the preceding calls. Finally, based on the output of the calls, the value of $\varphi$ is calculated in the standard way.

\[\square\]

Proposition 4.4. Model checking sATL$_{ir}$ and sATL$_{irF}$ is PSPACE-complete in the size of the representation and the length of the formula (even for formulas of bounded length).

Proof sketch. The lower bounds follow from Proposition 4.2. For the upper bounds, we use the algorithm from Proposition 4.3, but with the algorithm from Proposition 4.2 as the oracle. Since $\text{pPSPACE} = \text{PSPACE}$, we obtain the result.

\[\square\]

4.3 Model Checking sATL$_{ir}$ and sATL$_{irF}$

Finally, we examine the complexity of verification for specifications with arbitrary LTL subformulas.

Proposition 4.5. The following statements hold:

1. Model checking sATL$_{ir}$, sATL$_{irF}$, sATL$_{ir}^*$, sATL$_{irF}^*$ is PSPACE-complete in the size of the model and the formula.

2. For formulas of bounded length, the problem is NP-complete for sATL$_{ir}$ and sATL$_{irF}$, and between NP and $\text{NP} \subseteq \text{PSPACE}$ for sATL$_{ir}^*$ and sATL$_{irF}^*$.

Proof sketch. For (1), the lower bound follows from PSPACE-completeness of LTL model checking [65]. The upper bound for sATL$_{ir}$ and sATL$_{irF}$ can be obtained by guessing the strategy, pruning the model, and verifying the LTL formulas from Propositions 4.2 (note that $\text{NP} \subseteq \text{PSPACE} = \text{PSPACE}$). For sATL$_{ir}^*$ and sATL$_{irF}^*$, we repeat this for each subformula, and compute the Boolean combination.

For (2), the lower bound follows from Proposition 4.1. The inclusion in NP for sATL$_{ir}$ and sATL$_{irF}$ can be obtained by an algorithm similar to that in Proposition 4.1, only an LTL rather than CTL model checker is called. Since LTL model checking is $\text{NLOGSPACE}$-complete for formulas of bounded size [65], and $\text{NLOGSPACE} \subseteq \text{P}$, the upper bound follows.

The upper bound for sATL$_{ir}$ and sATL$_{irF}$ is obtained by an algorithm similar to that in Proposition 4.3, only an LTL rather than CTL model checker is called inside the oracle.

\[\square\]

Proposition 4.6. Model checking sATL$_{ir}^*$, sATL$_{irF}^*$, sATL$_{ir}^{*F}$, and sATL$_{irF}^{*F}$ is PSPACE-complete in the size of the representation and the formula (even for the formulas of bounded length).

Proof sketch. The lower bounds follow from Proposition 4.2. The upper bounds are obtained analogously to Proposition 4.4.

\[\square\]

4.4 Discussion

The above complexity results show that model checking fragments of sATL$_{ir}$ and sATL$_{irF}$ wrt compact representations (i.e., asynchronous MAS) is hard, and the size of the representation is the main factor for this hardness. Moreover, they suggest that there is no general method better than unfolding the representation to an explicit model (i.e., an interleaved interpreted system), and then verifying the IIS. This is because, with PSPACE-complete problems, one should expect exponential running time in practice. Thus, it is essential for the unfolding to produce as small models as possible. In what follows, we recall the idea of partial order reduction, very
5 PARTIAL ORDER REDUCTIONS

Partial order reductions have been defined for various configurations of temporal and temporal-epistemic logics without the "next step" operator $X$ [31, 48, 49, 57, 62]. The idea is to generate reduced models that either preserve some kind of model equivalence, or preserve representatives of Mazurkiewicz traces. The former method was used, for instance, to construct POR for $\text{LTL}_\chi$ and $\text{LTLK}_\chi$ based on stuttering trace equivalence [48, 49], and to obtain reductions for $\text{CTL}^*_\chi$ and $\text{CTLK}_\chi$ based on stuttering bisimulation [31, 48, 49]. The latter method was applied e.g. to prove correctness of reduction for $\text{LTL}_\chi$ formulas under the concurrency-fair semantics [57].

It is essential to notice that the practical value of a reduction scheme depends on how discriminative the underlying notion of equivalence is. Since $\text{CTL}^*_\chi$ equivalences are more discriminative than those for $\text{LTL}_\chi$, partial order reductions preserving $\text{LTL}_\chi$ produce smaller models than these for $\text{CTL}^*_\chi$. $\text{ATL}^*_\chi$ has even more distinguishing power than $\text{CTL}^*_\chi$. Thus, one can expect that equivalences preserving full $\text{ATL}^*_\chi$ ($\text{ATL}$ without the next step operator $X$) would be very discriminative, and result in very inefficient reductions. Aware of this and motivated by practical applications, we look for subsets of $\text{ATL}^*_\chi$ for which the most efficient known partial order reduction methods (i.e., those for $\text{LTL}_\chi$) can be applied.

In what follows, we show that the reductions for $\text{LTL}_\chi$, $\text{CTL}_\chi$, $\text{CTLK}_\chi$, and $\text{CTLK}_\chi$ can be adapted to $\text{sATL}^*_\chi^\mathsf{if}$, both with and without the CF assumption. We begin by introducing the relevant notions of equivalence (Sections 5.1 and 5.2). Then, we propose conditions on reduced models that preserve the equivalences (Sections 5.3 and 5.4). Finally, we present algorithms for POR and show their correctness (Section 6).

Interestingly, it turns out that our approach does not apply to $\text{sATL}^*_\chi^\mathsf{cf}$, cf. Section 6.3. This suggests that $\text{ATL}$ with imperfect information, besides conceptual advantage, can possibly offer some technical benefits over $\text{ATL}$ with perfect information.

5.1 Stuttering Equivalences

Let $M$ be a model, $M' \subseteq M$, and $PV \subseteq PV$ a subset of propositions. Stuttering equivalence says that two paths can be divided into corresponding finite segments, each satisfying exactly the same propositions. Stuttering path equivalence requires two models to always have stuttering-equivalent paths.

**Definition 5.2 (Stuttering equivalence [23]).** Models $M$ and $M'$ are stuttering equivalent, denoted $M \equiv_M M'$ if for each path $\pi \in \Pi_M(i)$, there is a path $\pi' \in \Pi_M'(i)$ such that $\pi \equiv_M \pi'$.

**Theorem 5.3 ([23]).** If $M \equiv_M M'$, then we have $M, i \models \phi$ iff $M', i' \models \phi$, for any $\text{LTL}_\chi$ formula $\phi$ over PV.

5.2 Independence-Based Equivalences

Partial order reductions for concurrency-fair $\text{LTL}_\chi$ are based on Mazurkiewicz traces [53–55]. Consider two finite sequences of actions $w, w' \in \text{Act}^*$. We say that $w \sim 1 w'$ iff $w = w_1abw_2$ and $w' = w_1baw_2$, for some $w_1, w_2 \in \text{Act}^*$ and $(a, b) \in I_0$. Let $\equiv$ be the reflexive and transitive closure of $\sim$. By (finite) traces we mean the equivalence classes of $\equiv$, denoted by $[w]_\equiv$.

Let $\nu, \nu' \in \text{Act}^\omega$, and let $\text{Pref}([\nu])$ denote the set of the finite prefixes of $\nu$. Now, $\nu \leq_{1} \nu'$ iff $\forall u \in \text{Pref}([\nu]) \exists u' \in \text{Pref}([\nu']) \exists u'' \in \text{Pref}([\nu') \cap [\nu]_\equiv \models \nu' \leq_{1} \nu$. That is, each finite prefix of $\nu$ can be extended to a permutation (under commuting adjacent independent actions) of some prefix of $\nu'$. Moreover, let $\nu \equiv^\omega \nu'$ iff $\nu \leq_{1} \nu'$ and $\nu' \leq_{1} \nu$. Infinite traces are defined as equivalence classes of $\equiv^\omega$, denoted by $[\nu]_\equiv^\omega$.

**Theorem 5.4 ([61]).** Let $M$ be a model. If $\pi, \pi' \in \Pi_M(i)$ such that $\text{Act}(\pi) \equiv^\omega_M \text{Act}(\pi')$, then $\pi \equiv \pi'$.

Thus, paths over representatives of the same infinite trace cannot be distinguished by any $\text{LTL}_\chi$ formula over $PV$. Note that Mazurkiewicz traces preserve $\text{CF}$, i.e., if $\pi \in \Pi_M^\omega(i)$, then for each $\pi'$ such that $\text{Act}(\pi) \equiv^\omega_M \text{Act}(\pi')$, we have $\pi' \in \Pi_M^\omega(i)$.

5.3 Preserving Traces for $\text{sATL}^*_\chi^\mathsf{if}$

Rather than generating the full model $M = \Pi\Pi\Pi(S)$, one can generate a reduced model $M'$ satisfying the following property:

**AE-CF:** $(\forall \pi \in \Pi_M^\omega(i)) (\exists \pi' \in \Pi_{M'}^\omega(i)) \text{Act}(\pi) \equiv^\omega_M \text{Act}(\pi')$.

Then, $M'$ preserves the $\text{LTL}_\chi$ formulas under $\text{CF}$ over $PV$ [61]. We will now prove that this also works for $\text{sATL}^*_\chi^\mathsf{if}$.

We first show that each set $\text{out}_M(g, \sigma_A)$ is trace-complete in the sense that with each path $\pi$ such that $\text{Act}(\pi) = w$, it contains a path over any $w' \in [w]_\equiv^\omega$.

**Lemma 5.5.** Let $\pi \in \Pi_M(i, \sigma_A)$ and $\text{Act}(\pi) = w$. Then, $\forall w' \in [w]_\equiv^\omega \exists \pi' \in \Pi_M(i, \sigma_A)$ such that $\text{Act}(\pi') = w'$.

**Proof.** Let $M'$ be obtained from $M$ by fixing $P_i(l_i) = \{\sigma_i(l_i)\}$ for each $i \in A, l_i \in L_i$, and pruning the transitions accordingly. The set of paths $\Pi_M(i)$ of $M'$ must be trace-complete [61]. But $\Pi_M'(i) = \text{out}_M(i, \sigma_A)$, which ends the proof.

The above lemma implies the following.

**Lemma 5.6.** Let $M$ be a model and $M'$ its submodel satisfying the property $\text{AE-CF}$. Then, for each $\text{ir}$-strategy $\sigma_A$, $\forall \pi \in \Pi_{M'}^\omega(i, \sigma_A) \exists \pi' \in \Pi_{M'}^\omega(i, \sigma_A)$ such that $\text{Act}(\pi') \equiv_{\Pi_{M'}^\omega(i, \sigma_A)}^\omega \text{Act}(\pi')$.

**Proof.** Assume that $\pi \in \Pi_{M'}^\omega(i, \sigma_A)$. Then there is $\pi' \in \Pi_{M'}^\omega(i, \sigma_A)$ such that $\text{Act}(\pi') \equiv_{\Pi_{M'}^\omega(i, \sigma_A)}^\omega \text{Act}(\pi')$ (by $\text{AE-CF}$). Since $M'$ is a submodel

---

3 The property is usually called stuttering trace equivalence. We opt for a slightly different name to avoid confusion with Mazurkiewicz traces, also used in this paper.

4 Typically, the definition contains also the symmetric condition which in our case always holds for $M$ and its submodel $M'$, as $\Pi_M(i) \subseteq \Pi_M'(i)$.
of $M$, we have that $π’ ∈ \text{out}_{M}^{CF}(i)$. This implies that $π’ ∈ \text{out}_{M}^{CF}(i, σ_A)$ by Lemma 5.5. Since $π’ ∈ \text{out}_{M}^{CF}(i, σ_A)$, by Definition 3.4, we obtain that $π’ ∈ \text{out}_{M}^{CF}(i, σ_A)$, which together with the fact that $\text{Act}(π) ≡_{I}^{\phi} \text{Act}(π’)$ completes the proof. ■

Theorem 5.7. Let $M$ be a model and $M’$ its submodel satisfying $\text{AE-CF}$. For each $\text{sATL}_{ir}$ formula $φ$ over $PV$ we have:

$$M, i \models_{\text{wfs}} φ \iff M’, i’ \models_{\text{wfs}} φ.$$  

Proof. Proof by induction on the structure of $φ$. We show the case $φ = ⟨⟨A⟩⟩y$. The cases for $\neg, \land$ are straightforward.

($⇒$) Follows from the fact that for each $σ_A$ we have $\text{out}_{M’}^{CF}(i, σ_A) = \text{out}_{M}^{CF}(i, σ_A) \cap \Pi_{M}^{CF}(i)$, so $\text{out}_{M}^{CF}(i, σ_A) \subseteq \text{out}_{M}^{CF}(i, σ_A)$.

($⇐$) Assume that $M’, i \models_{\text{wfs}} ⟨⟨A⟩⟩y$. From the semantics, there is an ir-joint strategy $σ_A$ such that for each $π ∈ \text{out}_{M}^{CF}(i, σ_A)$ we have $M’, π \models_{\text{wfs}} y$. In order to prove the thesis, we show that for each $π’ ∈ \text{out}_{M}^{CF}(i, σ_A)$ there is $π’ ∈ \text{out}_{M}^{CF}(i, σ_A)$ such that $π ≡_{\text{wfs}} π’$. So, $M’, π’ \models_{\text{wfs}} y$ implies that $M, π \models_{\text{wfs}} y$. Thus, we can conclude that $M, i \models_{\text{wfs}} ⟨⟨A⟩⟩y$. ■

5.4 Stuttering Equivalence without $\text{CF}$

The method based on Mazurkiewicz traces works well for $\text{sATL}_{ir}$, and we will present an algorithm generating reduced models that satisfy condition $\text{AE-CF}$ in Section 6. The same cannot be easily applied to the semantics without fairness. In particular, it is unclear how to generate reduced models that satisfy the analogue of $\text{AE-CF}$ in all paths. However, a similar result can be obtained through stuttering equivalence, based on the following property:

$$\text{AE}_{A}(\forall \sigma_A \in \text{config}(M, i) \forall π ∈ \text{out}_{M}(i, σ_A) \exists π’ ∈ \text{out}_{M}(i, σ_A) : π ≡_{\text{weis}} π’.$$  

Theorem 5.8. Let $A ⊆ \mathcal{A}$, and let $M’$ be a submodel of $M$ satisfying $\text{AE}_{A}$. For each $\text{sATL}_{ir}$ formula $φ$ over $PV$, that refers only to coalitions $A ⊆ \mathcal{A}$:

$$M, i \models_{\text{wfs}} φ \iff M’, i’ \models_{\text{wfs}} φ.$$  

Proof. Proof by induction on the structure of $φ$. We show the case $φ = ⟨⟨A⟩⟩y$. The cases for $\neg, \land$ are straightforward.

Notice that $\text{out}_{M}(i, σ_A) \subseteq \text{out}_{M’}(i, σ_A)$, which together with $\text{AE}_{A}$ implies that the sets $\text{out}_{M}(i, σ_A)$ and $\text{out}_{M’}(i, σ_A)$ are stuttering path equivalent. So, the thesis follows from Theorem 5.3. ■

Thus, we have proved that the structural conditions $\text{AE-CF}$ and $\text{AE}_{A}$ are sufficient to obtain correct reductions with and without fairness (Theorems 5.7 and 5.8). We will discuss algorithms that generate such reduced models in Section 6.

6 ALGORITHMS FOR POR

As mentioned above, the idea of model checking with POR is to reduce the size of models while preserving satisfaction for a class of formulas. Traditionally, the reduction algorithm is based either on depth-first-search (DFS, see [31]), or on double-depth-first-search (DDFS [24]). In this paper, we use the former.

6.1 DFS Algorithm

In the following, the stack represents a path $π = g_0a_0g_1a_1 \cdots g_n$ that is currently being visited. For the top element of the stack $g_n$ the following three operations are computed in a loop:

1. Identify the set $\text{en}(g_n) \subseteq \text{Act}$ of enabled actions.

2. Heuristically select a subset $E(g_n) \subseteq \text{en}(g_n)$ of possible actions (see Section 6.2).

3. For any action $a ∈ E(g_n)$, compute the successor state $g’$ such that $g_n \rightarrow_a g’$, and add $g’$ to the stack thereby generating the path $π’ = g_0a_0g_1a_1 \cdots g_n g’$. Recursively proceed to explore the submodel originating at $g’$.

4. Remove $g_n$ from the stack.

The algorithm begins with the stack comprising of the initial state of $M = \text{IIS}(S)$, and terminates when the stack is empty. Notice that the model generated by the algorithm must be a submodel of the $M$. Moreover, it is generated directly from the AMAS, without ever generating the full model $M$. Finally, the size of the reduced model crucially depends on the ratio of $E(g)/\text{en}(g)$. The choice of $E(g)$ is discussed in the next subsection.

6.2 Heuristics for $\text{sATL}_{ir}$ and Subsets of $\text{sATL}_{ir}$

Let $A ⊆ \mathcal{A}$. The conditions $C1 – C3$ below, inspired by [23], define a heuristics for a selection of $E(g) \subseteq \text{en}(g)$ in the algorithm of Sect. 6.1.

C1 Along each path $π$ in $M$ that starts at $g$, each action that is dependent on an action in $E(g)$ cannot be executed in $π$ without an action in $E(g)$ is executed first in $π$. Formally, $\forall π ∈ Π_{M}(g)$ such that $π = g_0 a_0 g_1 a_1 \cdots$ with $g_0 = g$, and $∀ b ∈ \text{Act}$ such that $(b, c) \notin I_A$ for some $c ∈ E(g)$, if $a_i = b$ for some $i ≥ 0$, then $a_j \in E(g)$ for some $j < i$.

C2 If $E(g) \neq \text{en}(g)$, then $E(g) \subseteq \text{In}(g)$.  

C3 For every cycle in $M’$ there is at least one node $g$ in the cycle for which $E(g) = \text{en}(g)$, i.e., for which all the successors of $g$ are expanded.

Theorem 6.1. Let $M = \text{IIS}(S)$, and $M’ ⊆ M$ be the reduced model generated by DFS with the choice of $E(g’)$ for $g’ ∈ St’$ given by conditions $C1, C3$ and the independence relation $I_A$, where $A = \emptyset$. Then, $M’$ satisfies $\text{AE-CF}$.  

Proof. See [61, Theorem 3.3]. ■

Theorem 6.2. Let $A ⊆ \mathcal{A}$, $M = \text{IIS}(S)$, and $M’ ⊆ M$ be the reduced model generated by DFS with the choice of $E(g’)$ for $g’ ∈ St’$ given by conditions $C1, C2, C3$ and the independence relation $I_A$. Then, $M’$ satisfies $\text{AE}_{A}$.  

Proof. Although the setting is slightly different, it can be shown similarly to [23, Theorem 12] that the conditions $C1, C2, C3$ guarantee that the models $M$ and $M’$ are stuttering path equivalent. More precisely, for each path $π = g_0 a_0 g_1 a_1 \cdots$ with $g_0 = i$ in $M$ there is a stuttering equivalent path $π’ = g_0 a_0’ g_1 a_1’ \cdots$ with $a_0’ = i$ in $M’$ such that $\text{Act}(π)\mid_{\text{Vis}_{A}} = \text{Act}(π’)(\mid_{\text{Vis}_{A}}$, i.e., $π$ and $π’$ have the same maximal sequence of visible actions for $A$.

To show that $M’$ satisfies $\text{AE}_{A}$, consider an ir-joint strategy $σ_A$ and $π ∈ \text{out}_{M}(i, σ_A)$. Since $M ≡_{I} M’$, we have that there is $π’ ∈ Π_{M’}(i)$ such that $π ≡_{\text{weis}} π’$ and $\text{Act}(π)\mid_{\text{Vis}_{A}} = \text{Act}(π’)(\mid_{\text{Vis}_{A}}$. Since $\text{Act}_{I} \subseteq \text{Vis}_{A}$ for each $i ∈ A$, the same sequence of actions of each $\text{Act}_{I}$ is executed in $π$ and $π’$. Thus, by the generalization of Lemma 3.7 to ir-joint strategies we get $π’ ∈ \text{out}_{M}(i, σ_A)$. So, by Lemma 3.6 we have $π’ ∈ \text{out}_{M}(i, σ_A)$. ■
Thus, we have obtained a general method of POR for fragments of ATL with imperfect information. The method is in fact a reformulation of the reduction for LTL. This has at least two welcome implications. First, the actual reductions are likely to be substantially much more than one would expect with the expressivity of sATL. Secondly, one can reuse or adapt existing algorithms and tools performing reductions for LTL. Algorithms generating reduced models, in which the choice of $E(g)$ is given by $C_1$, $C_2$, $C_3$ or $C_1$, $C_3$ can be found for instance in [23, 31, 49, 57, 58, 62].

6.3 Bad News for Agents with Perfect Information

Here, we briefly show that the adaptation of LTL-\text{X} reduction does not work for sATL with memoryless perfect information. We begin with a counterexample to Lemma 5.5 which was essential to our formal construction (Example 6.3). Then, we show that the whole method does not preserve formulas of sATL*$_\mu$ (Example 6.4).

Example 6.3. Consider the MAS composed of two agents {1, 2} such that: $L_1 = \{l_1^1, l_1^2\}$, $L_2 = \{l_2^1, l_2^2\}$, $Act_1 = \{e, a\}$, $Act_2 = \{e, b\}$, $P_1(l_1^1) = (a, e)$, $P_1(l_1^2) = (e)$, $P_2(l_2^1) = (b)$, $P_2(l_2^2) = (e)$, and $T_1(l_1^1, a) = l_1^2$, $T_2(l_2^1, b) = l_2^2$.

Define an $Ir$-strategy $\sigma_{(1,2)}$ as follows: $\sigma_1(l_1^1, l_1^2) = a$, $\sigma_1(l_1^2, l_1^2) = \sigma_2(l_1^1, l_1^2) = \sigma_2(l_1^2, l_1^2) = \sigma_2(l_1^2, l_1^2) = \epsilon$. It is easy to see that $out((l_1^1, l_1^2), \sigma_{(1,2)})$ is not trace complete. Note that $(a, b) \in I$, but while $out((l_1^1, l_1^2), \sigma_{(1,2)})$ contains the path over $ab\epsilon^\omega$, it does not contain any path over $ba\epsilon^\omega$.

Example 6.4. Consider formula $\langle c \rangle (\langle f \in i_1 \wedge f \in i_2 \rangle)$, interpreted with the $Ir$ semantics. Clearly, the formula holds in the TGC model in Figure 2a, but not in the reduced model in Figure 2b.

7 HOW BIG IS THE GAIN?

The efficiency of our method follows from the efficiency of partial order reductions for LTL-\text{X}, which has been documented in many papers [49, 56, 60]. We refer to those papers for experimental results, and present here only a quick estimation of the savings that are obtained for the Trains and Controller scenario from Section 2. Let $TGC_n$ be the asynchronous MAS consisting of the controller $c$ and $n$ trains $(t_1, \ldots, t_n)$. Take $PV = \{i_1, \ldots, i_n\}$, and let $i_n \in V(g)$ iff $g^i = T$. That is, $i_n$ holds iff train $t_i$ is in the tunnel.

Note that each action of the controller changes one of the $i_n$ variables. Hence, all the actions of $c$ are visible. It is easy to check that both variants of the POR algorithm from Section 6 generate the reduced model $M'_n$ in Figure 2b. For instance, in the global state $(G, A, W)$, two transitions are enabled: $b_1$ and $a_1$. The set $\{a_1\}$ satisfies conditions $C_1$, $C_2$, $C_3$, whereas $b_1$ is a visible transition. Thus, state $(R, A, T)$ is not visited. Similarly, in $(G, W, A)$, transitions $a_1$ and $b_3$ are enabled. The set $\{b_1\}$ satisfies conditions $C_1$, $C_2$, $C_3$, whereas $a_1$ is a visible transition. Therefore, state $(R, T, A)$ is not visited.

By Theorems 6.1 and 5.7, the reduced model $M'_n$ satisfies exactly the same sATL*$_\mu$ formulas over $PV$ as $IIS(TGC_n)$. Moreover, by Theorems 6.2 and 5.8, $M'_n$ and $IIS(TGC_n)$ satisfy the same formulas of sATL*$_\mu$ using only the strategic operators $\langle c \rangle$, $\langle \emptyset \rangle$. So, for example, one can-model check formula $\langle c \rangle G \neg i_1$ in $M'_n$ instead of $TGC_n$, and get the same output.

How big is the gain? Quoting the estimates from [48, 49], the size of the full state space is $|St_{IIS(TGC_n)}| \geq 2^{n+1}$, while the size of the reduced model is $|St_{M'_n}| = 2n + 1$. Thus, the reduced state space is exponentially smaller than the size of the full model. Of course, such optimistic results are by no means guaranteed. For many AMAS, the reduction may remove a smaller fraction of states. Still, it is important to note that the complexity of $ATL_\mu$ model checking is NP-hard in the size of the model (not the size of the representation!), and all the attempts at actual algorithms so far run in exponential time. So, even a linear reduction of the state space is likely to produce an exponential improvement of the performance.

8 CONCLUSIONS AND FUTURE WORK

Many important properties of multi-agent systems are underpinned by the ability of some agents (or groups) to achieve a given goal. In this paper, we propose a general semantics of strategic ability for asynchronous MAS, and study the model checking problem for relevant subsets of alternating-time temporal logic. We concentrate on imperfect information strategies, and consider two semantic variants: one looking at all the infinite executions of strategies, and the other taking into account only the fair execution paths.

The theoretical complexity results follow the same pattern as those for synchronous MAS, though proving them required careful treatment. Consequently, model checking of strategic abilities under imperfect information for asynchronous systems is as hard as in the synchronous case. This makes model reductions essential for practical verification. The most important result of this paper consists in showing that the partial order reduction for LTL-\text{X} can be almost directly applied to $ATL_\mu$ without nested strategic modalities. The importance of the result stems from the fact that LTL-\text{X} has relatively weak distinguishing power, and therefore admits strong reductions, clustering paths into relatively few equivalence classes.

Interestingly, it turns out that the scheme does not work for ATL with perfect information strategies. Until now, virtually all the results have suggested that verification of strategic abilities is significantly easier for agents with perfect information. Thus, we identify an aspect of verification that might be in favor of imperfect information strategies in some contexts.

The ideas presented in this paper open many exciting paths for future research. We will have a closer look at some alternative semantics for $ATL_\mu$ in asynchronous MAS, including the “deadlock-friendly” semantics and the one based on “subjective” ability. We also plan to extend our method to a larger subset of $ATL^*$ specifications, a subset of Strategy Logic [12], and to $ATL^*$ with epistemic operators using possibly techniques reported in [18]. Experimental evaluation of the reductions, on known benchmarks and randomly generated models, is also on the list. Adapting the POR scheme to combinations of strategic and epistemic modalities is another interesting path for future work. Finally, we would like to investigate if our partial order reduction scheme can be combined with the bisimulation-based reduction for $ATL_\mu$, proposed recently in [8].

Acknowledgements. We thank the anonymous reviewers for their insightful comments. W. Jamroga and W. Penczek acknowledge the support of the National Centre for Research and Development Poland (NCBiR) under the PolLux grant VoteVerif (POL-LUX-IV/1/2016).


