On Alternating-Time Temporal Logic, Hyperproperties, and Strategy Sharing

Raven Beutner, Bernd Finkbeiner

CISPA Helmholtz Center for Information Security, Germany

Abstract

Alternating-time temporal logic (ATL*) is a well-established framework for formal reasoning about multi-agent systems. However, while ATL* can reason about the strategic ability of agents (e.g., some coalition A can ensure that a goal is reached eventually), we cannot *compare* multiple strategic interactions, nor can we require multiple agents to follow the *same* strategy. For example, we cannot state that coalition A can reach a goal *sooner* (or *more often*) than some other coalition A'. In this paper, we propose HyperATL$^*_S$, an extension of ATL* in which we can **(1)** compare the outcome of multiple strategic interactions w.r.t. a *hyperproperty*, i.e., a property that refers to multiple paths at the same time, and **(2)** enforce that some agents *share* the same strategy. We show that HyperATL$^*_S$ is a rich specification language that captures important AI-related properties that were out of reach of existing logics. We prove that model checking of HyperATL$^*_S$ on concurrent game structures is decidable. We implement our model-checking algorithm in a tool we call `HyMASMC` and evaluate it on a range of benchmarks.

1 Introduction

Logics play a key role in the specification and verification of strategic properties in multi-agent systems (MAS) (Calegari et al. 2021). One of the most influential temporal logics for MASs is alternating-time temporal logic (ATL*), which extends CTL* with (implicit) quantification over strategies (Alur, Henzinger, and Kupferman 2002). As an example, assume we want to formally verify that a set of agents A can ensure that some temporal objective ψ is ultimately fulfilled. We can express this as the ATL* formula $\langle\!\langle A \rangle\!\rangle \mathsf{F}\, \psi$, stating that the agents in A have a joint strategy that ensures that all compatible executions eventually (F) satisfy ψ. Likewise, we can express that coalition A has *no* strategy to ensure that ψ is reached as $[\![A]\!]\, \mathsf{G}\, \neg\psi$, i.e., for every strategy of A, some execution globally (G) satisfies $\neg\psi$.

However, in many situations, we are interested not only in the strategic (in)ability of a coalition but also in comparing the ability of multiple coalitions. For example, we might ask if some coalition A is able to reach some goal ψ strictly sooner (or more often) than some other coalition A'. Indeed, important game-theoretic concepts such as *Shapley values* (Shapley 1953) are inherently based on the relative contribution of individual agents: To compute the Shapley value for some agent i, we need to *compare* the ability of some arbitrary coalitions A with that of $A \cup \{i\}$. Stating such comparison-based properties in ATL* is impossible as ATL* only considers a single path in isolation.

Hyperproperties. In contrast, the formal methods community has extensively studied properties that relate *multiple* system executions and coined them *hyperproperties* (Clarkson and Schneider 2008). In this paper, we bring the powerful concept of hyperproperties to the realm of AI and MASs. We introduce HyperATL$^*_S$ – a temporal logic that combines *strategic reasoning* (as found in ATL*), the ability to compare executions w.r.t. a *hyperproperty* (as, e.g., found in HyperATL*), and the possibility of enforcing agents to *share* strategies. As in HyperATL* (Beutner and Finkbeiner 2021, 2023b), we bind the outcome of a strategic interaction (resulting from an ATL*-like quantification) to a *path variable* and can then refer to atomic propositions on multiple paths. This combination of strategic reasoning and hyperproperties is needed for many AI-related properties; not only the information-flow properties envisioned in (Beutner and Finkbeiner 2021, 2023b). For example, HyperATL$^*_S$ allows us to express that coalition A can reach ψ strictly sooner than coalition A' as follows.

$$\langle\!\langle A \rangle\!\rangle \pi.\, [\![A']\!] \pi'.\, (\neg\psi_{\pi'})\, \mathsf{U} (\neg\psi_{\pi'} \wedge \psi_{\pi}).$$

This formula states that there exist strategies for the agents in A, such that for every path π under those strategies, it holds that: under *every* strategy for the agents in A', there exists some compatible path π', such that π reaches ψ (denoted ψ_π) before π' does (expressed using LTL's *until* operator U). Phrased differently, some strategy A can ensure that ψ is reached *strictly* faster than any strategy for A' could.

Note that this approach is very flexible, as we can compare π and π' w.r.t. to an arbitrary temporal property (e.g., π reaches ψ *more often* than π'). This goes well beyond the capabilities of ATL*, even when extended with quantitative operators (cf. Section 2).

Strategy Sharing and HyperATL$^*_S$. HyperATL$^*_S$ then extends HyperATL* with the ability to force agents to follow the same strategy. A *sharing constraint* ξ is a set of pairs of agents, and the HyperATL$^*_S$ formula $\langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi$ requires

that coalition A can satisfy φ, under the assumption that all agents $(i, j) \in \xi$ play the *same* strategy; similar to what is possible in strategy logic (Mogavero et al. 2014; Chatterjee, Henzinger, and Piterman 2010) in a *non-hyper* setting.

Example 1. *Assume we deal with a MAS modeling a planning task with multiple robots and want to ensure that robots in coalition A can reach some target state. To keep the employment overhead as small as possible, we might ask if the robots can follow some optimal trajectory (i.e., reach the target as fast as possible), despite all using the* same *strategy. We can express this in HyperATL$^*_S$ as follows*

$$\langle\!\langle A \rangle\!\rangle_{\{(i,j)|i,j\in A\}}\, \pi.\, [\![A]\!]\, \pi'.\, (\neg target_{\pi'})\, \mathsf{U}\, target_{\pi}$$

stating that all *robots in A can use a* shared *strategy (on path π) that reaches the target at least as fast as they can without the constraint that they must play the same strategy (path π'). Such shareable strategies are, e.g., key for scalable synthesis (Attie and Emerson 1998).*

We provide further HyperATL$^*_S$ examples (such as determinism and good-enough synthesis) in Section 6.2.

Model Checking. We show that model checking (MC) of HyperATL$^*_S$ on finite-state concurrent game structures (a standard model of MASs) is decidable. As HyperATL$^*_S$ can relate multiple computation paths, we cannot employ the tree-automaton-based MC approach for ATL* (Alur, Henzinger, and Kupferman 2002). Instead, we develop a MC algorithm based on alternating word automata. Our algorithm iteratively simulates path quantification within an automaton, while ensuring that the strategy-sharing constraints between agents are fulfilled.

Implementation. We implement our model-checking algorithm (for *full* HyperATL$^*_S$) in a tool we call `HyMASMC`. Using `HyMASMC`, we can, for the first time, automatically check properties beyond the self-composition fragment of HyperATL* – the largest fragment supported by previous tools (Beutner and Finkbeiner 2021, 2023b) (cf. Section 2). We evaluate `HyMASMC` by verifying a range of properties in MASs from the literature. Our experiments show that our algorithm performs well on *non*-hyper instances that could already be handled using existing solvers (Cermák, Lomuscio, and Murano 2015) and can successfully verify hyperproperties that cannot be expressed in any existing logic, let alone checked with any existing tool.

Supplementary Material. Detailed proofs and additional material can be found in (Beutner and Finkbeiner 2023c).

2 Related Work

Various works have extended ATL* with abilities to reason about probabilistic systems (Chen and Lu 2007), incomplete information (Belardinelli et al. 2017; Berthon, Maubert, and Murano 2017; Belardinelli, Lomuscio, and Malvone 2019), and finite traces (Belardinelli et al. 2018). All of these extension refer to individual paths and cannot express properties that relate *multiple* paths. While resource-aware extensions offer *quantitative* reasoning (Alechina, Demri, and Logan 2020; Bouyer et al. 2019; Henzinger and Prabhu 2006; Jamroga, Konikowska, and Penczek 2016; Chen and Lu 2007), they still cannot state properties that go beyond computing quantities on individual paths. Strategy logic (SL) treats strategies as first-class objects and can naturally express properties where some agents share the same strategy (Mogavero et al. 2014; Chatterjee, Henzinger, and Piterman 2010). While SL can compare the same strategy in different scenarios, it is limited to a *boolean* combination of LTL properties on individual paths, i.e., we cannot compare different paths w.r.t. a *temporal* hyperproperty. All properties we consider in Sections 1 and 6.2 cannot be expressed in SL. Most existing hyperlogics, including HyperLTL and HyperCTL* (Clarkson et al. 2014), reason about paths in a (non-strategic) transition system. HyperATL* was the first temporal logic that combined *strategic* reasoning with the ability to express hyperproperties (Beutner and Finkbeiner 2021, 2023b). This captures strategic *information-flow policies* such as simulation-based non-interference (Mantel and Sabelfeld 2001) and non-deducibility of strategies (Wittbold and Johnson 1990). HyperATL$^*_S$ extends HyperATL* with the ability to force agents to share the same strategy, which is useful for many AI-related properties (cf. Example 1). Moreover, automated verification of HyperATL* was, so far, only possible for the self-composition fragment (Beutner and Finkbeiner 2023b). In this fragment, all quantifiers are grouped together by constructing the self-composition of a MAS (Barthe, D'Argenio, and Rezk 2011), which reduces verification to a parity game. While this fragment suffices for many security-related properties (which are naturally defined in terms of a self-composition), it does not capture any of the properties discussed in Sections 1 and 6. In contrast, our model-checking algorithm (implemented in `HyMASMC`) uses iterative quantifier elimination and is applicable to *all* HyperATL$^*_S$ formulas. In terms of tool support, the `MCMAS` tool family (Lomuscio, Qu, and Raimondi 2009) implements a range of model checkers for strategic properties (e.g., specified in ATL* or SL), often with a strong focus on knowledge (Fagin et al. 1995; van der Hoek and Wooldridge 2003). Generally, knowledge properties *are* hyperproperties; to "know something" means that it should hold on all indistinguishable paths, effectively relating multiple paths in a system (Bozzelli, Maubert, and Pinchinat 2015; Beutner et al. 2023). However, before `HyMASMC`, none of the existing verifiers could check general hyperproperties (beyond knowledge) in MASs.

3 Preliminaries

For two functions $f : X \to Z$ and $f' : Y \to Z$ with $X \cap Y = \emptyset$, we define $f \oplus f' : X \cup Y \to Z$ as the union of both functions. We let AP be a fixed finite set of atomic propositions and let $Agts$ be a fixed finite set of agents. For a set of agent $A \subseteq Agts$, we define $\overline{A} := Agts \setminus A$. Given some set X, we write X^+ (resp. X^ω) for the set of non-empty finite (resp. infinite) sequences over X. For $u \in X^\omega$ and $k \in \mathbb{N}$, we write $x(k)$ for the kth element, $u[k, \infty]$ for the infinite suffix starting at position k, and $u[0, k]$ for the finite prefix up to k. As the underlying model of MASs, we use concurrent game structures (CGS).

Definition 1 (Alur, Henzinger, and Kupferman (2002)). *A concurrent game structure is a tuple $\mathcal{G} = (S, s_0, \mathbb{A}, \kappa, L)$ where S is a finite set of states, $s_0 \in S$ is an initial state, $\mathbb{A}$ is a finite set of actions, $\kappa : S \times (Agts \to \mathbb{A}) \to S$ is a transition function, and $L : S \to 2^{AP}$ is a state labeling.*

An *action vector* is a function $\boldsymbol{a} : Agts \to \mathbb{A}$ assigning an action to each agent. Given a state s and action vector $\boldsymbol{a}$, the transition function κ determines the next state $\kappa(s, \boldsymbol{a})$. A *strategy* in $\mathcal{G}$ is a function $f : S^+ \to \mathbb{A}$, mapping finite paths to actions. We denote the set of all strategies in $\mathcal{G}$ with $Str(\mathcal{G})$. Given a state $s \in S$ and *strategy vector* $\boldsymbol{f} : Agts \to Str(\mathcal{G})$ mapping each agent to a strategy, we can construct the path $Play_{\mathcal{G}}(s, \boldsymbol{f}) \in S^\omega$ that results from each agent acting according to the strategy defined by $\boldsymbol{f}$. Formally, we define $Play_{\mathcal{G}}(s, \boldsymbol{f})$ as the unique infinite path $p \in S^\omega$ such that $p(0) = s$, and for every $k \in \mathbb{N}$ we have $p(k+1) = \kappa\big(p(k), \boldsymbol{a}_k\big)$ where $\boldsymbol{a}_k$ is the action vector defined by $\boldsymbol{a}_k(i) := \boldsymbol{f}(i)(p[0,k])$ for $i \in Agts$. That is, we map each agent i to the action selected by strategy $\boldsymbol{f}(i)$ on the prefix $p[0,k]$, and update the state according to κ.

Note that our CGS definition does not include a protocol function $\varrho : S \times Agts \to (2^{\mathbb{A}} \setminus \{\emptyset\})$ that, in each state, assigns each agent a set of allowed actions. We can simulate the protocol ϱ in the transition function κ by "rerouting" every action that is invalid (according to ϱ) to some allowed action, effectively limiting the available actions of an agent.

ATL*. We briefly recall the syntax and semantics of ATL*. Path and state formulas in ATL* are defined as follows:

$$\psi := a \mid \psi \wedge \psi \mid \neg\psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi \mid \varphi$$
$$\varphi := \langle\!\langle A \rangle\!\rangle\, \psi \mid [\![A]\!]\, \psi$$

where $a \in AP$ and $A \subseteq Agts$. The temporal X refers to the *next* timepoint, and $\psi_1\,\mathsf{U}\,\psi_2$ states that ψ_2 holds at some future timestep and ψ_1 holds at all timesteps *until* then. We use the standard Boolean connectives $\vee, \rightarrow, \leftrightarrow$, and Boolean constants $\top, \bot$, as well as the derived temporal operators *eventually* $\mathsf{F}\,\psi := \top\,\mathsf{U}\,\psi$ and *globally* $\mathsf{G}\,\psi := \neg\,\mathsf{F}\,\neg\psi$. For a path $p \in S^\omega$, we evaluate a path formula as expected:

$$\begin{array}{lll}
p \models_{\mathcal{G}} a & \text{iff} & a \in L\big(p(0)\big) \\
p \models_{\mathcal{G}} \psi_1 \wedge \psi_2 & \text{iff} & p \models_{\mathcal{G}} \psi_1 \text{ and } p \models_{\mathcal{G}} \psi_2 \\
p \models_{\mathcal{G}} \neg\psi & \text{iff} & p \not\models_{\mathcal{G}} \psi \\
p \models_{\mathcal{G}} \mathsf{X}\psi & \text{iff} & p[1,\infty] \models_{\mathcal{G}} \psi \\
p \models_{\mathcal{G}} \psi_1\,\mathsf{U}\,\psi_2 & \text{iff} & \exists k \in \mathbb{N}.\, p[k,\infty] \models_{\mathcal{G}} \psi_2 \text{ and} \\
 & & \quad \forall 0 \le m < k.\, p[m,\infty] \models_{\mathcal{G}} \psi_1 \\
p \models_{\mathcal{G}} \varphi & \text{iff} & p(0) \models_{\mathcal{G}} \varphi
\end{array}$$

For a state $s \in S$, we define:

$$\begin{array}{lll}
s \models_{\mathcal{G}} \langle\!\langle A \rangle\!\rangle \psi & \text{iff} & \exists \boldsymbol{f} : A \to Str(\mathcal{G}). \\
 & & \quad \forall \boldsymbol{f}' : \overline{A} \to Str(\mathcal{G}).\, Play_{\mathcal{G}}\big(s, \boldsymbol{f} \oplus \boldsymbol{f}'\big)] \models_{\mathcal{G}} \psi \\
s \models_{\mathcal{G}} [\![A]\!] \psi & \text{iff} & \forall \boldsymbol{f} : A \to Str(\mathcal{G}). \\
 & & \quad \exists \boldsymbol{f}' : \overline{A} \to Str(\mathcal{G}).\, Play_{\mathcal{G}}\big(s, \boldsymbol{f} \oplus \boldsymbol{f}'\big)] \models_{\mathcal{G}} \psi.
\end{array}$$

That is, $\langle\!\langle A \rangle\!\rangle \psi$ holds in state s if the agents in A can enforce ψ. Formally, this means that there exists a strategy for each agent in A (formalized as function $\boldsymbol{f}$) such that – no matter what strategy the agents in $\overline{A} = Agts \setminus A$ follow (function $\boldsymbol{f}'$) – the resulting path satisfies path formula ψ. Conversely, $[\![A]\!]\psi$ states that coalition A cannot *avoid* ψ, i.e., every strategy for A admits some path that satisfies ψ.

A CGS $\mathcal{G} = (S, s_0, \mathbb{A}, \kappa, L)$ satisfies φ, written $\mathcal{G} \models_{\text{ATL}^*} \varphi$, if $s_0 \models_{\mathcal{G}} \varphi$, i.e., φ holds in the initial state.

4 HyperATL$^*_S$

In ATL*, we can quantify over paths in the system (constructed by some strategy), but with each nested quantification, we create a new path, effectively losing the handle of the path(s) constructed previously. Consequently, formula $\langle\!\langle A \rangle\!\rangle \langle\!\langle A' \rangle\!\rangle \psi$ is equivalent to $\langle\!\langle A' \rangle\!\rangle \psi$. In HyperATL$^*_S$, we want to explicitly state hyperproperties on *multiple* paths. To accomplish this, we extend ATL* with the notation of *path variables* and – whenever we encounter a strategic path quantifier – bind the outcomes of this quantification to such a variable, similar to HyperCTL* (Clarkson et al. 2014) and HyperATL* (Beutner and Finkbeiner 2021, 2023b).

Syntax. Let $\mathcal{V} = \{\pi, \pi', \ldots\}$ be a set of *path variables*. Path and state formulas in HyperATL$^*_S$ are generated by the following grammar.

$$\psi := a_\pi \mid \psi \wedge \psi \mid \neg\psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi \mid \varphi_\pi$$
$$\varphi := \langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi \mid [\![A]\!]_\xi\, \pi.\, \varphi \mid \psi$$

where $a \in AP$, $\pi \in \mathcal{V}$, $A \subseteq Agts$, and $\xi \subseteq Agts \times Agts$ is a *sharing constraint*. We assume that nested state formulas are closed, i.e., for each atomic formula a_π, path variable π is bound by some quantifier.

Similar to ATL*, formula $\langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi$ states that there exists a strategy for coalition A such that all paths under that strategy satisfy φ. However, differently from ATL*, we bind this path to the path variable π. We can then use path variables to refer to multiple paths via indexed atomic propositions. The constraint ξ poses restrictions on the agents' strategies: if $(i,j) \in \xi$, then agents i and j should play the same strategy. We assume that for each quantifier $\langle\!\langle A \rangle\!\rangle_\xi$ and $[\![A]\!]_\xi$, the sharing constraint satisfies $\xi \subseteq (A \times A) \cup (\overline{A} \times \overline{A})$, i.e., ξ can enforce strategy sharing between agents in A and between agents in $\overline{A}$. We omit ξ if $\xi = \emptyset$.

Semantics. We evaluate HyperATL$^*_S$ formulas in the context of a *path assignment*, which is a partial mapping $\Pi : \mathcal{V} \rightharpoonup S^\omega$. We write $\emptyset$ for the path assignment with an empty domain. Given $k \in \mathbb{N}$, we define $\Pi[k, \infty]$ as the assignment defined by $\Pi[k,\infty](\pi) := \Pi(\pi)[k,\infty]$, i.e., the assignment where all paths are (synchronously) shifted by k positions. For a path $p \in S^\omega$, we define $\Pi[\pi \mapsto p]$ as the updated assignment that maps π to p. For path formulas, we define

$$\begin{array}{lll}
\Pi \models_{\mathcal{G}} a_\pi & \text{iff} & a \in L\big(\Pi(\pi)(0)\big) \\
\Pi \models_{\mathcal{G}} \psi_1 \wedge \psi_2 & \text{iff} & \Pi \models_{\mathcal{G}} \psi_1 \text{ and } \Pi \models_{\mathcal{G}} \psi_2 \\
\Pi \models_{\mathcal{G}} \neg\psi & \text{iff} & \Pi \not\models_{\mathcal{G}} \psi \\
\Pi \models_{\mathcal{G}} \mathsf{X}\psi & \text{iff} & \Pi[1,\infty] \models_{\mathcal{G}} \psi \\
\Pi \models_{\mathcal{G}} \psi_1\,\mathsf{U}\,\psi_2 & \text{iff} & \exists k \in \mathbb{N}.\, \Pi[k,\infty] \models_{\mathcal{G}} \psi_2 \text{ and} \\
 & & \quad \forall 0 \le m < k.\, \Pi[m,\infty] \models_{\mathcal{G}} \psi_1 \\
\Pi \models_{\mathcal{G}} \varphi_\pi & \text{iff} & \Pi(\pi)(0), \emptyset \models \varphi.
\end{array}$$

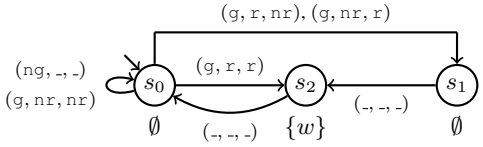

Figure 1: A simple CGS with $Agts = \{sched, W1, W2\}$. Each edge has the form (a_1, a_2, a_3) where a_1, a_2, and a_3 are the actions of $sched$, $W1$, and $W2$, respectively. We write "_" for an arbitrary action.

Whenever we check if a_π currently holds, we check if a holds on the path that is bound to π. A nested state formula φ_π holds iff φ holds in the first state of the path bound to π.

Given a set of agents $A \subseteq Agts$ and sharing constraints ξ, we define $shr_\mathcal{G}(A, \xi) := \{\boldsymbol{f} : A \to Str(\mathcal{G}) \mid \forall i, j \in A.\, (i, j) \in \xi \Rightarrow \boldsymbol{f}(i) = \boldsymbol{f}(j)\}$, i.e., all strategy vectors for A that satisfy the constraints in ξ. HyperATL$^*_S$ state formulas are evaluated in a state s and path assignments Π. For each strategy quantifier, we construct a new path and bind this path to a path variable in Π:

$$
\begin{aligned}
&s, \Pi \models_\mathcal{G} \psi && \text{iff} \quad \Pi \models_\mathcal{G} \psi \\
&s, \Pi \models_\mathcal{G} \langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi && \text{iff} \quad \exists \boldsymbol{f} \in shr_\mathcal{G}(A, \xi). \\
&\qquad \forall \boldsymbol{f}' \in shr_\mathcal{G}(\overline{A}, \xi).\, s, \Pi[\pi \mapsto Play_\mathcal{G}(s, \boldsymbol{f} \oplus \boldsymbol{f}')] \models_\mathcal{G} \varphi \\
&s, \Pi \models_\mathcal{G} [\![A]\!]_\xi\, \pi.\, \varphi && \text{iff} \quad \forall \boldsymbol{f} \in shr_\mathcal{G}(A, \xi). \\
&\qquad \exists \boldsymbol{f}' \in shr_\mathcal{G}(\overline{A}, \xi).\, s, \Pi[\pi \mapsto Play_\mathcal{G}(s, \boldsymbol{f} \oplus \boldsymbol{f}')] \models_\mathcal{G} \varphi
\end{aligned}
$$

Take $\langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi$ as an example. As in ATL*, we existentially quantify over strategies for the agents in A (subject to the condition that they respect the sharing constraints in ξ), followed by universal quantification over strategies for agents in $\overline{A}$ (again, subject to ξ). The resulting strategy vector $\boldsymbol{f} \oplus \boldsymbol{f}'$ then yields a unique path $Play_\mathcal{G}(s, \boldsymbol{f} \oplus \boldsymbol{f}')$, which we bind to path variable π and continue evaluation of φ. Note that in case $\xi = \emptyset$, the quantification behavior is very close to that of ATL* as $shr_\mathcal{G}(A, \emptyset)$ contains all functions $A \to Str(\mathcal{G})$. The important difference to ATL* is that once we have constructed the path $Play_\mathcal{G}(s, \boldsymbol{f} \oplus \boldsymbol{f}')$, we do not immediately evaluate a path formula but rather add the path to our current assignment. Without sharing constraints, HyperATL$^*_S$ corresponds to HyperATL* (Beutner and Finkbeiner 2021, 2023b) and is strictly more expressive than ATL*.

We say that $\mathcal{G}$ satisfies φ, written $\mathcal{G} \models \varphi$, if $s_0, \emptyset \models_\mathcal{G} \varphi$.

Example 2 (Running Example)**.** *Let us consider a very simple CGS between agents $Agts = \{sched, W1, W2\}$, describing a scheduler and two worker agents. The scheduler sched can choose actions $\{g, ng\}$ modeling a **g**rant or **n**o **g**rant, and each of the workers can choose actions $\{r, nr\}$ modeling a **r**equest to work or **n**o **r**equest to work. We model the dynamics of the CGS in Figure 1. If the scheduler chooses ng or both of the workers do not request to work, we remain in idle state s_0. If the scheduler grants work and* both *workers request to work, we directly transition to the working state s_2 where proposition $w \in AP$ holds. If only one of the workers requests work, we also transition to s_2 but pass through s_1, i.e., the work is delayed by one step.*

Let us assume we want to verify that coalition $\{sched, W1, W2\}$ can reach the work state s_2 (strictly) faster than $\{sched, W1\}$. As argued in the introduction, we can express this using the following HyperATL$^*_S$ formula

$$\langle\!\langle sched, W1, W2 \rangle\!\rangle\, \pi.\, [\![sched, W1]\!]\, \pi'.\, (\neg w_{\pi'})\, \mathsf{U}(\neg w_{\pi'} \wedge w_\pi).$$

This formula holds in the above CGS: $\{sched, W1, W2\}$ can construct a path π where w holds in the second step, whereas $\{sched, W1\}$ can, on their own, only ensure that w holds in the third step on π' (at the earliest).

HyperATL$^*_S$ and ATL*. HyperATL$^*_S$ subsumes ATL*:

Proposition 1. *For every ATL* formula φ, there exists an effectively computable HyperATL$^*_S$ formula φ' such that for every CGS $\mathcal{G}$, $\mathcal{G} \models_{ATL^*} \varphi$ iff $\mathcal{G} \models \varphi'$.*

5 Model Checking of HyperATL$^*_S$

While the extension of ATL* to reason about hyperproperties required only minor modifications to its syntax, the subtle changes bring major complications in terms of model checking. In particular, the model-checking algorithm for ATL* proposed by Alur, Henzinger, and Kupferman (2002) is no longer applicable: In ATL*, checking if $\langle\!\langle A \rangle\!\rangle \psi$ holds in some state s can be reduced to the non-emptiness of the intersection of two tree automata. One accepts all trees that represent possible strategies by the agents in A, and one accepts all trees whose paths satisfy the path formula ψ. In HyperATL$^*_S$, this is not possible: In a formula $\langle\!\langle A \rangle\!\rangle_\xi\, \pi.\varphi$, the satisfaction of φ does not only depend on π but also on path variables that are quantified before (outside).

5.1 Alternating Automata

Instead, our model-checking algorithm uses automata to "summarize" path assignments that satisfy subformulas, similar to previous hyperlogics such as HyperLTL (Finkbeiner, Rabe, and Sánchez 2015; Beutner and Finkbeiner 2023a), and HyperATL* (Beutner and Finkbeiner 2021, 2023b). To handle the strategic interaction found in MASs, we rely on *alternating automata*, i.e., automata that alternate between existential (non-deterministic) and universal transitions.

Definition 2. *An alternating parity automaton (APA) over alphabet Σ is a tuple $\mathcal{A} = (Q, q_0, \delta, c)$ where Q is a finite set of states, $q_0 \in Q$ is an initial state, $c : Q \to \mathbb{N}$ is a state coloring, and $\delta : Q \times \Sigma \to \mathbb{B}^+(Q)$ is a transition function that maps pairs of state and letter to a positive boolean formula over Q (denoted with $\mathbb{B}^+(Q)$).*

Formally, we model the alternation in APAs by viewing the transitions as positive boolean formulas over states (i.e., formulas formed using only conjunctions and disjunctions). For example, if $\delta(q, l) = q_1 \vee (q_2 \wedge q_3)$, we can – from state $q \in Q$ and upon reading letter $l \in \Sigma$ – either move to state q_1 or move to *both* q_2 and q_3 (i.e., spawn two copies of our automaton, one starting in state q_2 and one in q_3). We write $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$ for the set of all infinite words that are accepted by $\mathcal{A}$, i.e., all infinite words where we can construct a run tree such that for all paths, the *minimal* color that occurs infinity many times (as given by c) is *even*. See (Vardi 1995; Beutner and Finkbeiner 2023c) for details.

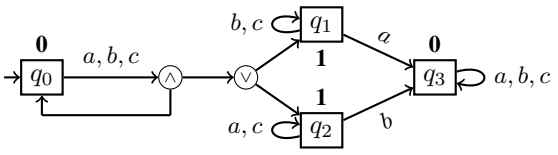

Figure 2: Example APA over alphabet $\Sigma = \{a, b, c\}$.

Example 3. *Consider the APA in Figure 2. We display the color of each state and visualize transition formulas using $\wedge$ and $\vee$ nodes. For example, $\delta(q_0, a) = \delta(q_0, b) = \delta(q_0, c) = q_0 \wedge (q_1 \vee q_2)$, i.e., whenever reading letter a, b, or c in q_0 we start a fresh run from q_0 and* at the same time *start a run from* either q_1 *or* q_2*. To derive the language of the APA, we first observe that state q_1 (resp. q_2) accepts all words that contain at least one a (resp. b) (note that the color of q_1, q_2 is odd). In the initial state q_0, we restart a run from q_0* and *transition to either q_1 or q_2. The language thus contains exactly those words that contain a or b infinitely often.*

Deterministic Automata. Our model-checking algorithm relies on the fact that we can *determinize* APAs. We say $\mathcal{A}$ is a *deterministic* parity automaton (DPA) if we can view δ as a function $Q \times \Sigma \to Q$ that assigns a *unique* successor state to each state, letter pair.

Proposition 2 (Miyano and Hayashi (1984))**.** *For any APA $\mathcal{A}$ with n states, we can effectively compute a DPA $\mathcal{A}'$ with at most $2^{2^{\mathcal{O}(n)}}$ states such $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

5.2 Model Checking Algorithm

We are now in a position to outline our model-checking algorithm. A high-level description is given in Algorithm 1. Here, we write $\langle\!\langle A \rangle\!\rangle$ as a shorthand for either $\langle\!\langle A \rangle\!\rangle$ or $[\![A]\!]$.

Nested State Formulas. Initially, our algorithm recursively checks nested state formulas and replaces them with fresh atomic propositions (Emerson and Halpern 1986). Concretely, given a closed state formula $\varphi = \langle\!\langle A_1 \rangle\!\rangle_{\xi_1} \pi_1 \dots \langle\!\langle A_n \rangle\!\rangle_{\xi_n} \pi_n . \psi$, we first extract all state formulas that are *nested* in the path formula ψ (line 2). For each nested state formula φ', we **(1)** compute all states in which φ' holds (using a recursive call to *modelCheck*); **(2)** mark all those states with a fresh atomic proposition $p_{\varphi'}$ by modifying the labeling function L of $\mathcal{G}$ (line 4); and **(3)** replace all occurrences of φ'_π within ψ with $(p_{\varphi'})_\pi$ (line 5).

Eliminating Path Quantification. Afterward, ψ contains no nested state formulas, and we can tackle the strategic quantifiers. For each state $\dot{s} \in S$, we check if $\dot{s}, \emptyset \models_{\mathcal{G}} \varphi$, and – if it does – add it to the solution set *Sol* (line 12). Our main idea to check $\dot{s}, \emptyset \models_{\mathcal{G}} \varphi$ is to iteratively eliminate paths $\pi_1, \dots, \pi_n$ by *simulating $\mathcal{G}$ using the alternation available in APAs* while summarizing path assignments that satisfy the formula from the fixed state $\dot{s} \in S$. To enable automata-based reasoning about path assignments, i.e., mappings $\Pi : V \to S^\omega$ for some $V \subseteq \mathcal{V}$, we *zip* such an assignment into an infinite word: Given $\Pi : V \to S^\omega$ we define $zip(\Pi) \in (V \to S)^\omega$ as the infinite word over functions $V \to S$, defined by $zip(\Pi)(k)(\pi) := \Pi(\pi)(k)$ for $k \in \mathbb{N}$.

Algorithm 1: Model-checking algorithm for HyperATL$^*_S$.

```
1  def modelCheck(𝒢, φ = ⟨⟨A_1⟩⟩_{ξ_1} π_1 ... ⟨⟨A_n⟩⟩_{ξ_n} π_n . ψ):
2    for φ' in nestedStateFormulas(ψ) do
3      S_{φ'} = modelCheck(𝒢, φ')
4      L = λs. { L(s)          if  s ∉ S_{φ'}
               { L(s) ∪ {p_{φ'}}  if  s ∈ S_{φ'}
5      ψ = ψ[φ'_{π_1}/(p_{φ'})_{π_1}] ··· [φ'_{π_n}/(p_{φ'})_{π_n}]
6    Sol = ∅
7    for ṡ ∈ S do
8      𝒜 = LTLtoAPA(ψ)
9      for j from n to 1 do
10       𝒜 = product(𝒢, ṡ, 𝒜, ⟨⟨A_j⟩⟩_{ξ_j} π_j)
11     if zip(∅) ∈ ℒ(𝒜) then
12       Sol = Sol ∪ {ṡ}
13   return Sol
```

Definition 3. *Assume φ is a HyperATL$^*_S$ formula with free path variables $V \subseteq \mathcal{V}$. We say an automaton $\mathcal{A}$ over $V \to S$ is $(\mathcal{G}, \dot{s})$*-equivalent to φ *if for every path assignment $\Pi : V \to S^\omega$ we have $zip(\Pi) \in \mathcal{L}(\mathcal{A})$ if and only if $\dot{s}, \Pi \models_{\mathcal{G}} \varphi$.*

Now assume that $\varphi = \langle\!\langle A_1 \rangle\!\rangle_{\xi_1} \pi_1 \dots \langle\!\langle A_n \rangle\!\rangle_{\xi_n} \pi_n . \psi$ is the state formula we want to check in state $\dot{s}$. If we *could* compute a $(\mathcal{G}, \dot{s})$-equivalent automaton $\mathcal{A}_\varphi$ for φ, we can immediately check whether $\dot{s}, \emptyset \models_{\mathcal{G}} \varphi$ by testing if $zip(\emptyset) \in \mathcal{L}(\mathcal{A}_\varphi)$. Our main theoretical result is that we can construct such an automaton *incrementally*: We begin with a $(\mathcal{G}, \dot{s})$-equivalent automaton $\mathcal{A}_\psi$ for the body ψ; we then use $\mathcal{A}_\psi$ to construct a $(\mathcal{G}, \dot{s})$-equivalent automaton $\mathcal{A}_{\langle\!\langle A_n \rangle\!\rangle_{\xi_n} \pi_n . \psi}$ for $\langle\!\langle A_n \rangle\!\rangle_{\xi_n} \pi_n . \psi$; and so forth, finally yielding the desired automaton $\mathcal{A}_\varphi$ that is $(\mathcal{G}, \dot{s})$-equivalent to φ. In each step, we apply the construction from the following theorem:

Theorem 1. *Assume that $\varphi = \langle\!\langle A \rangle\!\rangle_\xi \pi . \varphi'$ and let $\mathcal{A}_{\varphi'}$ be an APA over alphabet $(V \cup \{\pi\} \to S)$ that is $(\mathcal{G}, \dot{s})$-equivalent to φ'. We can effectively construct an APA $\mathcal{A}_\varphi$ over alphabet $V \to S$ that is $(\mathcal{G}, \dot{s})$-equivalent to φ. The size of $\mathcal{A}_\varphi$ is at most double exponential in the size of $\mathcal{A}_{\varphi'}$.*

Proof. Let $\mathcal{A}^{det}_{\varphi'} = (Q, q_0, \delta, c)$ be a DPA equivalent to $\mathcal{A}_{\varphi'}$ (cf. Proposition 2). We define $\mathcal{A}_\varphi = (Q \times S, (q_0, \dot{s}), \delta', c')$ where $c'(q, s) := c(q)$ and δ' is defined as follows: If $\varphi = \langle\!\langle A \rangle\!\rangle_\xi \pi . \varphi'$ we define $\delta'\big((q, s), l\big)$ for $l : V \to S$ as

$$\bigvee_{\substack{\boldsymbol{a}: A \to \mathbb{A} \\ \forall i,j \in A.(i,j) \in \xi \\ \Rightarrow \boldsymbol{a}(i) = \boldsymbol{a}(j)}} \quad \bigwedge_{\substack{\boldsymbol{a}': \overline{A} \to \mathbb{A} \\ \forall i,j \in \overline{A}.(i,j) \in \xi \\ \Rightarrow \boldsymbol{a}'(i) = \boldsymbol{a}'(j)}} \Big(\delta\big(q, l[\pi \mapsto s]\big), \kappa\big(s, \boldsymbol{a} \oplus \boldsymbol{a}'\big)\Big)$$

Conversely, if $\varphi = [\![A]\!]_\xi \pi . \varphi'$ we define $\delta'\big((q, s), l\big)$ as

$$\bigwedge_{\substack{\boldsymbol{a}: A \to \mathbb{A} \\ \forall i,j \in A.(i,j) \in \xi \\ \Rightarrow \boldsymbol{a}(i) = \boldsymbol{a}(j)}} \quad \bigvee_{\substack{\boldsymbol{a}': \overline{A} \to \mathbb{A} \\ \forall i,j \in \overline{A}.(i,j) \in \xi \\ \Rightarrow \boldsymbol{a}'(i) = \boldsymbol{a}'(j)}} \Big(\delta\big(q, l[\pi \mapsto s]\big), \kappa\big(s, \boldsymbol{a} \oplus \boldsymbol{a}'\big)\Big)$$

The intuition behind our construction is that we can simulate the strategic quantification at the level of states, similar to what is possible in HyperATL* (Beutner and Finkbeiner 2021, 2023b). Let us take $\varphi = \langle\!\langle A \rangle\!\rangle_\xi \pi . \varphi'$ as an example.

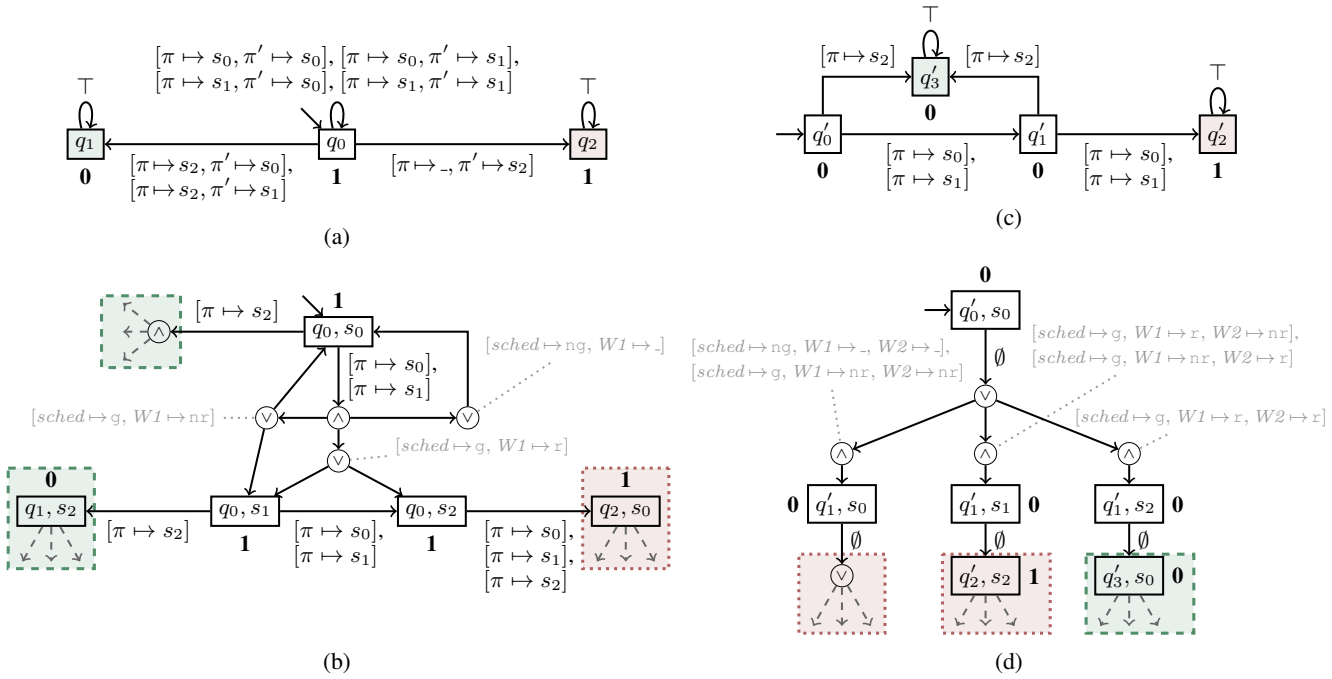

Figure 3: Illustration of our model-checking algorithm on Example 2. In Figure 3a, we depict a DPA over alphabet $\{\pi, \pi'\} \to \{s_0, s_1, s_2\}$ for the body $(\neg w_{\pi'}) \,\mathsf{U}\, (\neg w_{\pi'} \wedge w_\pi)$. In Figure 3b, we sketch the APA over alphabet $\{\pi\} \to \{s_0, s_1, s_2\}$ constructed using Theorem 1 that is $(\mathcal{G}, s_0)$-equivalent to subformula $[\![sched,\, W1]\!]\, \pi'.\, (\neg w_{\pi'}) \,\mathsf{U}\, (\neg w_{\pi'} \wedge w_\pi)$. In Figure 3c, we depict a DPA that is equivalent to the APA in Figure 3b. Lastly, in Figure 3d, we sketch the APA (over singleton alphabet $\emptyset \to \{s_0, s_1, s_2\}$) constructed using Theorem 1 that is $(\mathcal{G}, s_0)$-equivalent to $\langle\!\langle sched,\, W1,\, W2 \rangle\!\rangle\, \pi.\, [\![sched,\, W1]\!]\, \pi'.\, (\neg w_{\pi'}) \,\mathsf{U}\, (\neg w_{\pi'} \wedge w_\pi)$.

The desired automaton $\mathcal{A}_\varphi$ should accept a word $u \in (V \to S)^\omega$ iff there exists a strategy vector $\boldsymbol{f} : A \to \mathit{Str}(\mathcal{G})$ that respects ξ and for all paths π compatible with $\boldsymbol{f}$, the extended zipped path assignment (a word in $(V \cup \{\pi\} \to S)^\omega$) is accepted by $\mathcal{A}^{det}_{\varphi'}$. In our constructions, we track the current state q of $\mathcal{A}^{det}_{\varphi'}$ and simulate $\mathcal{G}$ by keeping track of the current state s. When in state (q, s), we update the automaton state according to the transition function of $\mathcal{A}^{det}_{\varphi'}$ using the current state s for path variable π. To update the state of $\mathcal{G}$, we simulate the strategic behavior: **(1)** we disjunctive fix actions for each agent in A via a function $\boldsymbol{a}$ and ensure that all sharing constraints hold; **(2)** we conjunctively choose actions for $\overline{A}$ as a function $\boldsymbol{a}'$ (subject to the sharing constraints); and **(3)** we update the system state to $\kappa(s, \boldsymbol{a} \oplus \boldsymbol{a}')$.

Arguing that $\mathcal{A}_\varphi$ is $(\mathcal{G}, \dot{s})$-equivalent to φ is based on the determinacy of the underlying game. As we work in a setting of complete information (where all agents observe the state of the CGS), we can replace the existential quantification over strategies for A (as in the semantics of $\langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi'$) with existential quantification over actions for A in each step (as used in the disjunctive choice in the definition of $\mathcal{A}_\varphi$). We give a formal proof in (Beutner and Finkbeiner 2023c).

The size of $\mathcal{A}_\varphi$ is linear in the size of $\mathcal{G}$ and $\mathcal{A}^{det}_{\varphi'}$ (which itself is doubly exponential in $\mathcal{A}_{\varphi'}$, cf. Proposition 2). □

For a formula $\varphi = \langle\!\langle A \rangle\!\rangle_\xi\, \pi.\, \varphi'$ and APA $\mathcal{A}$ that is $(\mathcal{G}, \dot{s})$-equivalent to φ', let *product*$(\mathcal{G}, \dot{s}, \mathcal{A}, \langle\!\langle A \rangle\!\rangle_\xi\, \pi)$ be the APA that is $(\mathcal{G}, \dot{s})$-equivalent to φ constructed using Theorem 1. In Algorithm 1, we start with an APA that is equivalent to ψ (line 8), and apply *product* to iteratively compute $(\mathcal{G}, \dot{s})$-equivalent automata for subformulas $\langle\!\langle A_j \rangle\!\rangle_{\xi_j} \pi_j \ldots \langle\!\langle A_n \rangle\!\rangle_{\xi_n} \pi_n.\, \psi$ for j from n to 1 (line 10). After the loop, we are left with an APA $\mathcal{A}$ over singleton alphabet $(\emptyset \to S)$ that is $(\mathcal{G}, \dot{s})$-equivalent to φ; we can thus decide if $\dot{s}, \emptyset \models_\mathcal{G} \varphi$ by checking if $\mathit{zip}(\emptyset) \in \mathcal{L}(\mathcal{A})$ (line 11).

Proposition 3. *For every CGS $\mathcal{G} = (S, s_0, \mathbb{A}, \kappa, L)$ and closed HyperATL$^*_S$ formula φ, we have*

$$\mathit{modelCheck}(\mathcal{G}, \varphi) = \big\{ s \in S \mid s, \emptyset \models_\mathcal{G} \varphi \big\}.$$

Complexity. Each application of *product* increases the size of $\mathcal{A}$ by (in the worst case) two exponents. Checking a HyperATL$^*_S$ formula with n nested quantifiers is thus in $2n$-EXPTIME, and MC for general formulas is non-elementary. As HyperATL$^*_S$ subsumes HyperATL*, we get a matching non-elementary *hardness* (Beutner and Finkbeiner 2023b). HyperATL$^*_S$ is thus more expressive (and also much harder to model-check) than ATL*. We stress that the non-elementary complexity of HyperATL$^*_S$ stems from its ability to quantify over *arbitrarily* many paths. In most properties of interest, we quantify over few paths (cf. Section 6), which results in a much lower (elementary) complexity. In particular, if we apply Algorithm 1 to an ATL*-equivalent formula (cf. Proposition 1), we deal with a *single* nested quantifier ($n = 1$) and thus match the 2-EXPTIME MC complexity known for ATL* (Alur, Henzinger, and Kupferman 2002).

5.3 Model Checking the Running Example

We illustrate our MC construction on the formula from Example 2. In a first step, we translate the body $(\neg w_{\pi'}) \mathsf{U} (\neg w_{\pi'} \wedge w_\pi)$ to a DPA over alphabet $(\{\pi, \pi'\} \to \{s_0, s_1, s_2\})$, depicted in Figure 3a. Afterward, we can follow the construction from Theorem 1 to obtain an APA over $(\{\pi\} \to \{s_0, s_1, s_2\})$ that is $(\mathcal{G}, s_0)$-equivalent to subformula $[\![sched, W1]\!]\pi'. (\neg w_{\pi'}) \mathsf{U} (\neg w_{\pi'} \wedge w_\pi)$. We depict a sketch in Figure 3b. We start in state (q_0, s_0). When reading letter $[\pi \mapsto s_2]$, we update the automaton state to q_1, so – as q_1 is an accepting sink – every run from such states is accepting. To aid readability, we stop exploration as soon as the automaton state equals q_1 or q_2 and mark them with a green (dashed border) and red (dotted border) box to represent acceptance and rejection, respectively. When reading letters $[\pi \mapsto s_0]$ or $[\pi \mapsto s_1]$, we remain in automaton state q_0. However, to update the state of the CGS, we need to simulate the strategic behavior within the CGS. As we quantify *universally* over strategies for $\{sched, W1\}$, we conjunctively consider all possible action vectors $\{sched, W1\} \to \mathbb{A}$. For each such action vector, we can then disjunctively choose an action for $W2$. In our visualization in Figure 3b, we use decision nodes (as in Example 3); for the reader's convenience, we label each conjunctive choice with the corresponding partial action vector. For example, if we conjunctively pick the (partial) action vector $[sched \mapsto \mathtt{g}, W1 \mapsto \mathtt{r}]$, agent $W2$ can (disjunctively) move to either (q_0, s_1) or (q_0, s_2).

To better understand of the APA we have just constructed, we can translate it to some equivalent DPA, depicted in Figure 3c. For this DPA, we can see that a path assignment is accepted iff s_2 (i.e., the unique state where AP w holds) occurs within the first two steps on π. This exactly matches the intuition of $(\mathcal{G}, s_0)$-equivalence: A path assignment $\Pi : \{\pi\} \to \{s_0, s_1, s_2\}^\omega$ satisfies $s_0, \Pi \models_{\mathcal{G}} [\![sched, W1]\!]\, \pi'. (\neg w_{\pi'}) \mathsf{U} (\neg w_{\pi'} \wedge w_\pi)$ iff s_2 occurs within the first two steps on π. If s_2 does *not* hold on in the first two steps, $\{sched, W1\}$ can ensure that s_2 holds in the third step on π' and thus violate the property.

We can use the DPA in Figure 3c and, again, apply Theorem 1 to the outermost quantifier $\langle\!\langle sched, W1, W2 \rangle\!\rangle \pi$, resulting in the APA over singleton alphabet $(\emptyset \to \{s_0, s_1, s_2\})$ sketched in Figure 3d. Here, we *disjunctively* pick an action vector $\{sched, W1, W2\} \to \mathbb{A}$ (annotated at each decision node). As there are no agents in $\overline{\{sched, W1, W2\}}$, each action vector yields a unique successor state. It is easy to see that this APA accepts $zip(\emptyset) = \emptyset^\omega$, proving $\mathcal{G} \models \varphi$.

6 Implementation and Experiments

We have implemented our algorithm in a tool we call `HyMASMC`. As input, our tool reads MASs in the form of ISPL models (Lomuscio, Qu, and Raimondi 2009). For automata operations (in particular, the translation from alternating to deterministic automata), we use `spot`; an actively-maintained automata library (Duret-Lutz et al. 2022). To check APAs over the *singleton* alphabet $(\emptyset \to S)$ for emptiness, we use the parity-game solver `oink` (van Dijk 2018). All results were obtained on a 3.60GHz Xeon® CPU (E3-1271) with 32GB of memory running Ubuntu 20.04.

n	$\lvert S \rvert$	$\lvert S_{reach} \rvert$	$t_{\texttt{MCMAS-SL[1G]}}$	$t_{\texttt{HyMASMC}}$
2	72	9	**0.11**	0.41
3	432	21	6.64	**2.06**
4	2592	49	322.7	**24.3**
5	15552	113	TO	**347.1**

Table 1: We compare `HyMASMC` and `MCMAS-SL[1G]`. We give the size of the system ($|S|$), the size of the reachable fragment ($|S_{reach}|$), and the verification times in seconds. The timeout (TO) is set to 1h.

6.1 Model Checking ATL*

In our first experiment, we want to compare the performance of `HyMASMC` against existing tools for strategic properties. This requires us to consider *non-hyper* properties in the form of ATL* specification (as no existing tool can handle hyperproperties). Concretely, we compare with `MCMAS-SL[1G]`, a solver for a fragment of strategy logic (Cermák, Lomuscio, and Murano 2015). We use the same benchmark family used by Cermák, Lomuscio, and Murano (2015), describing a parametric scheduling problem consisting of agents $Agts = \{sched, y_1, \ldots, y_n\}$ for $n \in \mathbb{N}$. We check the following HyperATL$^*_S$ formula

$$\langle\!\langle sched \rangle\!\rangle \pi. \bigwedge_{i=1}^{n} \mathsf{G} \left(\langle wt, i \rangle_\pi \to \mathsf{F} \neg \langle wt, i \rangle_\pi \right)$$

The formula states that whenever agent y_i waits (modeled by AP $\langle wt, i \rangle$), it will eventually not wait anymore, i.e., the scheduler has a strategy that avoids starvation of all agents. This formula is equivalent to the strategy logic specification used by Cermák, Lomuscio, and Murano (2015).

We check the HyperATL$^*_S$ formulas (and the equivalent strategy logic specifications) with `HyMASMC` and `MCMAS-SL[1G]` for varying values of n. We use the "optimized" algorithm in `MCMAS-SL[1G]` that decomposes the formula as much as possible (Cermák, Lomuscio, and Murano 2015, §4). The results are given in Table 1. We observe that `HyMASMC` performs much faster than `MCMAS-SL[1G]`, which we largely accredit to the very efficient backend solvers in `spot` and `oink`.

6.2 Model Checking Hyperproperties

In this section, we challenge `HyMASMC` with interesting hyperproperties. As underlying MAS models, we use the ISPL models used by `MCMAS` (Cermák, Lomuscio, and Murano 2015; Lomuscio, Qu, and Raimondi 2009) and design a range of specification templates that model interesting use cases of HyperATL$^*_S$. We emphasize that not every template models realistic properties in each of the ISPL instances. However, our evaluation **(1)** demonstrates that HyperATL$^*_S$ can express interesting properties, and **(2)** empirically shows that `HyMASMC` can check such properties in existing ISPL models (confirming this via further real-world scenarios is interesting future work). Note that none of the properties falls in the self-composition fragment of HyperATL* (Beutner and Finkbeiner 2021, 2023b), which formed the largest fragment supported by previous tools.

	Optimality I		Optimality II		Optimality III		OD		GE	
ISPL Model	t_{avg}	t_{max}	t_{avg}	t_{max}	t_{avg}	t_{max}	t_{avg}	t_{max}	t_{avg}	t_{max}
BIT-TRANSMISSION	0.38	0.41	0.39	0.42	0.39	0.44	0.39	0.44	0.38	0.42
BOOK-STORE	0.39	0.42	0.40	0.47	0.40	0.44	0.39	0.42	0.39	0.43
CARD-GAME	0.38	0.39	0.38	0.39	0.41	0.48	0.39	0.46	0.36	0.37
DINING CRYPTOGRAPHERS	0.70	0.77	0.68	0.85	0.70	0.77	0.69	0.74	0.69	1.10
MUDDY-CHILDREN	0.36	0.44	0.36	0.40	0.36	0.42	0.36	0.42	0.36	0.42
SIMPLE-CARD-GAME	0.35	0.38	0.35	0.37	0.35	0.38	0.35	0.38	0.34	0.35
SOFTWARE-DEVELOPMENT	-	-	-	-	-	-	-	-	-	-
STRONGLY-CONNECTED	0.35	0.41	0.34	0.37	0.35	0.37	0.37	0.40	0.34	0.38
TIANJI-HORSE-RACING-GAME	0.38	0.45	0.37	0.39	0.37	0.40	0.37	0.40	0.37	0.42
SCHEDULER-2	0.47	0.51	0.46	0.48	0.95	1.35	0.47	0.53	0.48	0.51
SCHEDULER-3	2.33	2.85	2.29	2.70	9.72	20.1	2.12	2.64	2.01	2.15
SCHEDULER-4	29.5	32.7	24.5	24.7	31.2	35.2	28.36	58.7	24.6	25.1

Table 2: For each ISPL model (Lomuscio, Qu, and Raimondi 2009), we display the average time (t_{avg}) and the maximal time (t_{max}) time (in seconds) needed by HyMASMC across the 20 random instances sampled from each template.

Optimality I. As argued in Section 1 and Example 2, a particular strength of HyperATL$^*_S$ is the ability to compare the power of different coalitions. For $A, A' \subseteq Agts$ and $tgt \in AP$ (modeling the target), we check

$$\langle\!\langle A \rangle\!\rangle\, \pi.\, [\![A']\!]\, \pi'.\, (\neg tgt_{\pi'})\, \mathsf{U} (\neg tgt_{\pi'} \wedge tgt_{\pi}).$$

Optimality II. Using the strategy sharing in HyperATL$^*_S$, we can also check if a coalition can achieve the goal equally fast despite using the same strategy for all agents (cf. Example 1). For a group of agents A and $tgt \in AP$, we use HyMASMC to check

$$\langle\!\langle A \rangle\!\rangle_{\{(i,j) \mid i,j \in A\}}\, \pi.\, [\![A]\!]\, \pi'.\, (\neg tgt_{\pi'})\, \mathsf{U}\, tgt_{\pi}.$$

Optimality III. Likewise, we can express that coalition A can reach a target state at strictly more time points than coalition A' as

$$\langle\!\langle A \rangle\!\rangle\, \pi.\, [\![A']\!]\, \pi'.\, \mathsf{G}(tgt_{\pi'} \rightarrow tgt_{\pi}) \wedge \mathsf{F}(\neg tgt_{\pi'} \wedge tgt_{\pi}).$$

Observational Determinism (OD). An important property in the context of security in MASs is *observational determinism* (Zdancewic and Myers 2003). For example, assume we have a system that contains a controller agent cnt and an AP h that models a high-security value of the system. We want to ensure that the value of h is in control of cnt, which we can express in HyperATL$^*_S$ as

$$\langle\!\langle cnt \rangle\!\rangle\, \pi.\, \langle\!\langle cnt \rangle\!\rangle\, \pi'.\, \mathsf{G}(h_{\pi} \leftrightarrow h_{\pi'}).$$

That is, cnt has a strategy to construct π such that in a second execution, cnt can ensure the same sequence of values for h (despite the other agents potentially acting differently).

Good-Enough Synthesis (GE). In many scenarios, there does not exist a strategy that wins in all situations. Instead, we often look for strategies that are *good-enough* (GE), i.e., strategies that win on every possible input sequence for which a winning output sequence exists (Almagor and Kupferman 2020; Aminof, Giacomo, and Rubin 2021; Li et al. 2021). We can express the existence of a GE strategy for $A \subseteq Agts$ in HyperATL$^*_S$ as

$$\langle\!\langle A \rangle\!\rangle\, \pi.\, \langle\!\langle \emptyset \rangle\!\rangle\, \pi'.\, (\mathsf{G}(in_{\pi} \leftrightarrow in_{\pi'}) \wedge \mathsf{F}\, tgt_{\pi'}) \rightarrow \mathsf{F}\, tgt_{\pi}.$$

That is, A has a strategy for π such that if any other (universally quantified) path π' agrees on the input $in \in AP$ with π and wins (e.g., reaches a state where $tgt \in AP$ holds), then π must win as well. Phrased differently, π only needs to win, provided some path with the same inputs *can* win.

Results. For each ISPL model, we sample 20 random HyperATL$^*_S$ formulas from each of the templates and display the verification times in Table 2. We observe that HyMASMC can verify almost all instances within a few seconds. Even on the challenging scheduler instances, verification of complex hyperproperties is only slightly more expensive than checking non-hyper ATL* formulas (cf. Table 1). The only exception is the SOFTWARE-DEVELOPMENT model; this model consists of roughly 15k states, which is too large for any automata-based representation. We stress that already in the non-hyper realm, MCMAS-SL[1G] cannot verify (even simple) ATL* and strategy logic specifications in the SOFTWARE-DEVELOPMENT model and is only applicable to ATL and CTL properties.

7 Conclusion

Starting with the seminal work on ATL*, the past decade has seen immense progress in (temporal) logic-based frameworks that provide rigorous and formal guarantees in MASs. Thus far, most logics focus on a purely path-based view where we reason about the strategic (in)ability of agents. However, many important properties require reasoning about multiple paths at the same time and investigating scenarios where agents share strategies. We have presented HyperATL$^*_S$, a powerful logic that bridges this gap. Our logic: **(1)** can express many important properties such as optimality requirements, OD, and GE; **(2)** admits decidable model checking; and **(3)** can be checked fully automatically using HyMASMC.

For the future, it is interesting to cast even more properties in a *unified* framework using (hyper)logics such as HyperATL$^*_S$ (similar to what has been done for ATL/ATL*) and explore even more scalable verification approaches for HyperATL$^*_S$ using, e.g., symbolic techniques.

Acknowledgments

This work was supported by the European Research Council (ERC) Grant HYPER (101055412), and by the German Research Foundation (DFG) as part of TRR 248 (389792660).

References

Alechina, N.; Demri, S.; and Logan, B. 2020. Parameterised Resource-Bounded ATL. In *Conference on Artificial Intelligence, AAAI 2020*.

Almagor, S.; and Kupferman, O. 2020. Good-Enough Synthesis. In *International Conference on Computer Aided Verification, CAV 2020*.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM*.

Aminof, B.; Giacomo, G. D.; and Rubin, S. 2021. Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up. In *International Joint Conference on Artificial Intelligence, IJCAI 2021*.

Attie, P. C.; and Emerson, E. A. 1998. Synthesis of Concurrent Systems with Many Similar Processes. *ACM Trans. Program. Lang. Syst.*

Barthe, G.; D'Argenio, P. R.; and Rezk, T. 2011. Secure information flow by self-composition. *Math. Struct. Comput. Sci.*

Belardinelli, F.; Lomuscio, A.; and Malvone, V. 2019. An Abstraction-Based Method for Verifying Strategic Properties in Multi-Agent Systems with Imperfect Information. In *Conference on Artificial Intelligence, AAAI 2019*.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2017. Verification of Multi-agent Systems with Imperfect Information and Public Actions. In *International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017*.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2018. Alternating-time Temporal Logic on Finite Traces. In *International Joint Conference on Artificial Intelligence, IJCAI 2018*.

Berthon, R.; Maubert, B.; and Murano, A. 2017. Decidability Results for ATL* with Imperfect Information and Perfect Recall. In *International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017*.

Beutner, R.; and Finkbeiner, B. 2021. A Temporal Logic for Strategic Hyperproperties. In *International Conference on Concurrency Theory, CONCUR 2021*.

Beutner, R.; and Finkbeiner, B. 2023a. AutoHyper: Explicit-State Model Checking for HyperLTL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*.

Beutner, R.; and Finkbeiner, B. 2023b. HyperATL*: A Logic for Hyperproperties in Multi-Agent Systems. *Log. Methods Comput. Sci.*

Beutner, R.; and Finkbeiner, B. 2023c. On Alternating-Time Temporal Logic, Hyperproperties, and Strategy Sharing. *CoRR*, 2312.12403.

Beutner, R.; Finkbeiner, B.; Frenkel, H.; and Metzger, N. 2023. Second-Order Hyperproperties. In *International Conference on Computer Aided Verification, CAV 2023*.

Bouyer, P.; Kupferman, O.; Markey, N.; Maubert, B.; Murano, A.; and Perelli, G. 2019. Reasoning about Quality and Fuzziness of Strategic Behaviours. In *International Joint Conference on Artificial Intelligence, IJCAI 2019*.

Bozzelli, L.; Maubert, B.; and Pinchinat, S. 2015. Unifying Hyper and Epistemic Temporal Logics. In *International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2015*.

Calegari, R.; Ciatto, G.; Mascardi, V.; and Omicini, A. 2021. Logic-based technologies for multi-agent systems: a systematic literature review. *Auton. Agents Multi Agent Syst.*

Cermák, P.; Lomuscio, A.; and Murano, A. 2015. Verifying and Synthesising Multi-Agent Systems against One-Goal Strategy Logic Specifications. In *Conference on Artificial Intelligence, AAAI 2015*.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Inf. Comput.*

Chen, T.; and Lu, J. 2007. Probabilistic Alternating-time Temporal Logic and Model Checking Algorithm. In *International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*.

Clarkson, M. R.; Finkbeiner, B.; Koleini, M.; Micinski, K. K.; Rabe, M. N.; and Sánchez, C. 2014. Temporal Logics for Hyperproperties. In *International Conference on Principles of Security and Trust, POST 2014*.

Clarkson, M. R.; and Schneider, F. B. 2008. Hyperproperties. In *Computer Security Foundations Symposium, CSF 2008*.

Duret-Lutz, A.; Renault, E.; Colange, M.; Renkin, F.; Aisse, A. G.; Schlehuber-Caissier, P.; Medioni, T.; Martin, A.; Dubois, J.; Gillard, C.; and Lauko, H. 2022. From Spot 2.0 to Spot 2.10: What's New? In *International Conference on Computer Aided Verification, CAV 2022*.

Emerson, E. A.; and Halpern, J. Y. 1986. "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. *J. ACM*.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*.

Finkbeiner, B.; Rabe, M. N.; and Sánchez, C. 2015. Algorithms for Model Checking HyperLTL and HyperCTL*. In *International Conference on Computer Aided Verification, CAV 2015*.

Henzinger, T. A.; and Prabhu, V. S. 2006. Timed Alternating-Time Temporal Logic. In *International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2006*.

Jamroga, W.; Konikowska, B.; and Penczek, W. 2016. Multi-Valued Verification of Strategic Ability. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2016*.

Li, Y.; Turrini, A.; Vardi, M. Y.; and Zhang, L. 2021. Synthesizing Good-Enough Strategies for LTLf Specifications.

In *International Joint Conference on Artificial Intelligence, IJCAI 2021*.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In *International Conference on Computer Aided Verification, CAV 2009*.

Mantel, H.; and Sabelfeld, A. 2001. A Generic Approach to the Security of Multi-Threaded Programs. In *Computer Security Foundations Workshop, CSFW 2001*.

Miyano, S.; and Hayashi, T. 1984. Alternating Finite Automata on omega-Words. *Theor. Comput. Sci.*

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Trans. Comput. Log.*

Shapley, L. S. 1953. A Value for n-Person Games.

van der Hoek, W.; and Wooldridge, M. J. 2003. Cooperation, Knowledge, and Time: Alternating-time Temporal Epistemic Logic and its Applications. *Stud Logica*.

van Dijk, T. 2018. Oink: An Implementation and Evaluation of Modern Parity Game Solvers. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2018*.

Vardi, M. Y. 1995. Alternating Automata and Program Verification. In *Computer Science Today: Recent Trends and Developments*.

Wittbold, J. T.; and Johnson, D. M. 1990. Information Flow in Nondeterministic Systems. In *Symposium on Security and Privacy, SP 1990*.

Zdancewic, S.; and Myers, A. C. 2003. Observational Determinism for Concurrent Program Security. In *Computer Security Foundations Workshop, CSFW 2003*.