Coalition Alternating-time Temporal Logic: a Logic to Find Good Coalitions to Achieve Strategic Objectives

Davide Catta[1], Angelo Ferrando[2], and Vadim Malvone[1]

[1] LTCI, Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France
{davide.catta,vadim.malvone}@telecom-paris.fr
[2] Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, Genoa, Italy angelo.ferrando@unige.it

Abstract. Alternating-time Temporal Logic (ATL) extends the temporal logic CTL, permitting quantification over coalitions of agents. During the model checking process, the coalitions defined in a given formula are predetermined, operating under the assumption that the user possesses knowledge about the specific coalitions under exam. However, this presumption is not universally applicable. The outcome of this paper is twofold. Initially, we introduce CATL, a modified version of ATL which empowers users to define coalition quantifiers based on two key attributes: the number of agents involved within the coalitions and the methodology for grouping these agents. Subsequently, we show the incorporation of CATL into MCMAS, a widely recognized tool dedicated to ATL model checking. Additionally, we provide details of this extension accompanied by empirical experiments.

Keywords: Logics for Strategic Reasoning · Alternating-time Temporal Logic · Coalition of Agents · Formal Verification

1 Introduction

It is difficult to fully trust software systems. When considering the reliability of a software system, it is therefore crucial to ensure certain guarantees are in place for the end user. Out of these guarantees is demonstrating the correctness of the system under exams, and it poses a particularly challenging task. For this very reason, the formal verification of Multi-Agent Systems (MAS) has become an important research field in both theoretical and applied computer science during the last decades. Reactive, autonomous, and distributed systems have become ubiquitous nowadays, and as a result, the need to verify the correctness of such systems has emerged. One of the main contributions in formal verification of MAS is model checking: to verify if a MAS satisfies a given property of interest, a mathematical model (usually a graph-like entity) of the MAS is defined and the property of interest is expressed as a formula in some logical language (usually temporal logic). To ensure that the MAS satisfies the property, we check whether the mathematical representation of the MAS is a *model* (in the logical sense) of the logical formula expressing the property of interest. Agents of a MAS are often conceived as players of a concurrent game: they act synchronously, they have a goal, and they can pursue the latter either alone or by forming coalitions with other

agents. For this reason, logics that are used to specify the properties of interest are usually strategic logics [1,17]. In particular, one of the most popular logics that is used to specify desired properties of a MAS is Alternating-time Temporal Logic (ATL) [1]. This latter logic is an extension of Computation Tree Logic (CTL) obtained by substituting the path quantifiers "there is a path" and "for all paths" of CTL with two strategic operators, $\langle\langle\Gamma\rangle\rangle$ and $[\![\Gamma]\!]$, whose meaning can be spelled out as "there exists a joint strategy for the coalition of agents Γ" and "for all strategies for the coalition of agents Γ", respectively. In this context, a strategy is a generic conditional plan that at each step of the game prescribes an action. With more detail, there are two main classes of strategies: memoryless and memoryfull. In the former case, agents choose an action by considering only the current state while, in the latter case, agents choose an action by considering the full history of the MAS.

From a practical point of view, the most popular tool for the model checking of Multi-Agent Systems is MCMAS [16]. In this tool, the Multi-Agent System is formally modeled as an interpreted system that is a product of local models, one for each agent involved in the multi-agent system to represent its visibility. MCMAS provides the specification of properties via CTL, ATL [1], Strategy Logic [17], and some of their extensions/fragments. The tool handles the model checking problem by using a Binary Decision Diagram (BDD) representation for models and formulas.

Notice that, in the model checking process, both from a theoretical and practical viewpoint, the coalitions in the strategic operators need to be fixed before the verification process. The latter constraint is not always well-known by the developers/users called to verify the Multi-Agent System. In the following, we analyze the main features related to our extension.

Coalitions as variables. To automatically generate the coalitions in an ATL formula φ, it is first necessary to have a way to uniquely identify each coalition inside the formula. In more detail, given an ATL formula φ, we can annotate each strategic operator with a corresponding variable. Just to make an example. Let us assume the ATL formula φ is as follows: $\langle\langle a,b\rangle\rangle F\langle\langle b,c\rangle\rangle G\langle\langle a,c\rangle\rangle Xp$; with $\{a,b\}$, $\{b,c\}$, and $\{a,c\}$ three coalitions, and p an atomic proposition. The formula becomes $\langle\langle\Gamma_1\rangle\rangle F\langle\langle\Gamma_2\rangle\rangle G\langle\langle\Gamma_3\rangle\rangle Xp$, where Γ_1, Γ_2, and Γ_3 are three variables, which will be replaced by the automatically generated coalitions. Note that, in all strategic operators we may add the same Γ variable. In such a case, we would enforce to use the same coalition in the three strategic operators of φ.

In the following, we report the kind of rules we want to enforce over the coalitions. Such rules guide the coalitions' generation, so that all coalitions proposed by our approach both make φ satisfied in the formal model and respect all the guidelines.

Two types of features could be of interest in our investigation: the number of agents and how to group the agents.

Number of agents in coalition. The first analysis concerns the size of the coalitions to generate. In more detail, we want to enforce the minimum (resp. maximum) number of agents per coalition. This is very important, because it relates to possible real-world limitations. For instance, there might be scenarios where coalitions with less than n or more than m agents are not reasonable, because to create a coalition of less than n or more than m agents is too expensive for its gain. For this reason, a *min* (resp.

max) constraint can be specified to rule out all Γ coalitions such that $min \leq |\Gamma|$ (resp. $|\Gamma| \leq max$).

Agents in the same/different coalition. Another relevant group of features concerns which agents can (or not) be in the same coalition. Again, this finds its motivation in real-world applications, where it is common to have limitations on how some agents can be grouped. For instance, considering that the agents are commonly situated in an environment, it may be possible that some of them are close (or not) to each other. For this reason, there might be interest in not having in the same coalition agents that are far from each other (for technological and practical reasons), while there might be interest in having in the same coalition agents that are local to each other. For this reason, a $[a \rightarrow\leftarrow b]$ (resp. $[a \leftarrow\rightarrow b]$) constraint can be specified to keep all Γ coalitions s.t. $a \in \Gamma \iff b \in \Gamma$ (resp. $a \in \Gamma \implies b \notin \Gamma$ and $b \in \Gamma \implies a \notin \Gamma$); where a and b can be any agent.

In this paper, we propose a methodology to tackle the above mentioned rules. To do so, we first introduce a variant of ATL (which we call Coalition ATL) in which the following specifications can be expressed:

1. there is a coalition Π which includes all the members of Γ and all those of Δ such that the member of Π can realize φ by coordinating their actions;
2. there is a coalition Π which includes all the members of Γ and no member of Δ such that the members of Π can realize φ by coordinating their actions;
3. there is a coalition Π counting at most n agents such that the members of Π can realize φ by coordinating their actions;
4. there is a coalition Π counting at least n agents such that the members of Π can realize φ by coordinating their actions;

where Γ and Δ are two coalitions of agents, n is a natural number, and φ is a formula expressing a desired property of the MAS.

Coalition ATL (or CATL for short) is obtained by considering four new strategic operators whose intuitive semantics is expressed in the four above statements. We study the formal properties of this logic. In particular, we show that CATL and ATL have the same expressive power, but CATL can express some ATL properties in an exponentially more concise manner.

After having introduced CATL, we detail an implementation of it in MCMAS. More precisely, we present an extension of MCMAS in which we give the ability to the end user to characterize the coalitions in the strategy quantifiers with respect to two main features: the *number of agents* involved in the coalitions and *how to group* the agents. That is, we ask the user to give some information on the coalitions involved in each strategic operator by considering them as a variable of the problem. With more detail, the user can input a minimum and maximum number of agents involved in the coalitions and give guidelines with respect to the agents that have to (resp., cannot) stay in the same coalitions. After that, our tool extracts all coalitions of agents that respect the user's guidelines. Then, for each valid coalition, our tool verifies the formal specification over the Multi-Agent System. Finally, the coalitions that make the formal specification satisfied in the Multi-Agent System are returned to the user. We consider our

work as a first stone on the development of a more generalized tool for the verification of multi-agent systems.

Differently from [13], in this work we introduce the formal machinery to verify parametric ATL formulas w.r.t. the generated coalitions. To achieve this, we introduce CATL, a variant of ATL that we prove to be as expressive as ATL, but whose formulas are exponentially more succinct. In addition, we provide additional experimental results on some CATL formulas.

Structure of the work. The paper is structured as follows. In Section 2 we give some related works on formal verification of multi-agent systems. In Section 3 we define CATL, we prove that CATL and ATL have the same expressive power, and we study the model checking for our logic. In Section 4 we recall the definition of interpreted systems, which are the mathematical models used in practice. To exemplify our approach, we present a variant of the Train Gate Controller in Section 5. Then, in Section 6 we provide the algorithms to solve CATL model checking via ATL model checking. Finally, we give the details of our extension for MCMAS in Section 7 and, in Section 8, provide experimental results on a parameterised version of the Train Gate Controller scenario. We conclude in Section 9 by recapping our work.

2 Related Work

In the introduction, we mentioned another important logic for the strategic reasoning called Strategy Logic (SL) [17]. The latter is a powerful formalism for strategic reasoning. As a key aspect, this logic treats strategies as *first-order objects* that can be determined by means of the existential $\exists x$ and universal $\forall x$ quantifiers, which can be respectively read as *"there exists a strategy x"* and *"for all strategies x"*. Therefore, in Strategy Logic, these strategies are not intrinsically glued to a specific agent, but an explicit binding operator (a,x) allows to link an agent a to the strategy associated with a variable x. Unfortunately, the high expressiveness of SL comes at a price. Indeed, it has been proved that the model checking problem for SL becomes non-elementary complete [17] and the satisfiability undecidable [18]. To gain back elementariness, several fragments of SL have been considered. Among the others, Strategy Logic with Simple-Goals [5] considers SL formulas in which strategic operators, bindings operators, and temporal operators are coupled. It has been shown that Strategy Logic with Simple-Goals strictly subsumes ATL and its model checking problem is PTIME-COMPLETE, as it is for ATL [1]. Note that, none of these fragments, nor SL, explicitly allow to parameterize over the coalition of agents.

To conclude this section, we want to focus on the agents' information. Specifically, we distinguish between *perfect* and *imperfect* information [20]. The former corresponds to a basic setting in which every agent has full knowledge about the MAS. However, in real-life scenarios it is common to have situations in which agents have to play without having all relevant information at hand. In computer science these situations occur for example when some variables of a system are internal/private and not visible to an external environment [15,8]. In MAS, the imperfect information is usually modeled by setting an indistinguishability relation over the states of the system [15,20,19]. This

feature deeply impacts on the model checking complexity. For example, ATL becomes undecidable in the context of imperfect information and memoryful strategies [10]. To overcome this problem, some works have either focused on an approximation to perfect information [3,7], developed notions of bounded memory [2,6], or developed hybrid techniques [14,11,12]. Note that, even though in this work we focus on the perfect information scenario, our results hold in the imperfect one as well. We decided to tackle the former case to simplify the presentation and help the reader to fully understand the contribution of the work.

3 Coalition ATL

In this section, we introduce Coalition ATL (CATL for short). Before detailing the syntax and semantics of CATL, let us fix some notation and terminology that we shall use along the paper.

Preliminary notions. If V is a set and $U \subseteq V$, we denote by $\overline{U}$ the complementary $V \setminus U$ of U in V. If v is a (finite or infinite) sequence over U, we denote by $|v|$ its length (which is ω if v is infinite), by v_i its i-th element, by $v_{\leq i}$ the finite prefix $v_1, \ldots, v_i$ of v and by $v_{\geq i}$ the (possibly infinite) suffix of v starting at v_i. If v is a finite sequence, $last(v)$ denotes the last element $v_{|v|}$ of v. We fix once and for all a finite set Ap of atomic propositions or atoms (the letters $p, q, r, \ldots$ will range over this set) and a finite set $\mathsf{Ag} = \{1, \ldots, n\}$ of agents. A subset of Ag will be called a coalition and we will use the Greek Letters $\Gamma, \Delta, \Pi \ldots$ to range over them. If $l = \langle x_1, \ldots, x_n \rangle$ is a tuple, we denote by $l[i]$ the i-th component x_i of the tuple.

3.1 Syntax

We now define the syntax of CATL.

Definition 1. *State φ and path ψ formulas are defined by mutual induction using the following grammar:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle\Gamma\rangle\rangle\psi \mid \langle\langle\Gamma \rightarrow \leftarrow \Delta\rangle\rangle\psi \mid \langle\langle\Gamma \rightarrow \Delta\rangle\rangle\psi \mid \langle\langle\leq_n\rangle\rangle\psi \mid \langle\langle\geq_n\rangle\rangle\psi$$

$$\psi ::= \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{R} \varphi$$

where p is an atom, Γ and Δ are coalitions, and $n \leq |\mathsf{Ag}|$ is a natural number. The boolean connectives $\rightarrow$ and $\wedge$, and the temporal connectives F and G can be defined as usual. We define $\langle\langle\Gamma \leftarrow\rightarrow \Delta\rangle\rangle\psi$ as $(\langle\langle\Gamma \rightarrow \Delta\rangle\rangle\psi) \vee (\langle\langle\Delta \rightarrow \Gamma\rangle\rangle\psi)$. Formulas of CATL are all and only state formulas. Formulas of ATL are CATL formulas in which no occurrences of the operators $\langle\langle\Gamma \rightarrow \leftarrow \Delta\rangle\rangle, \langle\langle\Gamma \rightarrow \Delta\rangle\rangle$, $\langle\langle\leq_n\rangle\rangle$, and $\langle\langle\geq_n\rangle\rangle$ appear for any Γ, Δ, and n.

The size of a formula is the height of its construction tree, the formal definition follows.

Definition 2. *The size* $|\theta|$ *of a formula* θ *is recursively defined as follows:*

- *if* θ *is an atom then* $|\theta| = 0$*;*
- *if* θ *is* $\theta_1 \star \theta_2$ *with* $\star \in \{\wedge, \vee, \mathsf{U}, \mathsf{R}\}$*, then* $|\theta| = max(|\theta_1|, |\theta_2|) + 1$*;*
- *if* θ *is* $\circ\theta_1$ *with* $\circ \in \{\mathsf{X}, \langle\langle\Gamma\rangle\rangle, \langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle, \langle\langle\Gamma \rightarrow \Delta\rangle\rangle, \langle\langle{\leq_n}\rangle\rangle, \langle\langle{\geq_n}\rangle\rangle\}$*, then* $|\theta| = |\theta_1| + 1$.

3.2 Semantics

We specify the meaning of CATL formulas by means of Concurrent Game Structures (CGS for short). Intuitively, a CGS is a labeled directed graph that represents the possible evolution of a given Multi-Agent System with respect to simultaneous choices of actions of a group of (autonomous) agents. Both states and edges are labeled by members of two disjoints alphabets. States are labeled by atomic propositions. These atomic propositions represent the properties that are true at a given state. Each edge is labeled by a tuple, and each member of a given tuple represents an action that is available for a given agent at the source state of the edge. The formal definition follows.

Definition 3. *Given a set* Ap *of atomic proposition, and a set* $\mathsf{Ag} = \{1, \ldots, k\}$ *of agents, A Concurrent Game Structure with Imperfect Information (iCGS for short) constructed over* Ap *and* Ag *is a tuple* $M = \langle S, s_I, \{Act_i\}_{i \in \mathsf{Ag}}, P, T, \{\sim_i\}_{i \in \mathsf{Ag}}, \mathcal{L}\rangle$*, where:*

- S *is a finite set of states and* $s_I \in S$ *is the initial state.*
- Act_i *is a finite non-empty set of actions for any* $i \in \mathsf{Ag}$*; we denote by ACT the product set* $\Pi_{i \in \mathsf{Ag}} Act_i$ *and we call elements of this set* joint actions*.*
- $P : S \times Ag \rightarrow (2^{Act_i} \setminus \emptyset)$ *is the protocol function which assigns a non empty-subset of actions* $P(s,i)$ *of* Act_i *to any agent* i *and state* s*. The set* $P(s,i)$ *represents the set of actions that are available at the state* s *to the agent* i*.*
- $T : S \times ACT \rightarrow S$ *is the (partial) transition function. Such function associates to any state* s *and joint action* $\mathbf{a} = \langle a_1, \ldots, a_k\rangle$ *such that for all* $i \in \mathsf{Ag}$*,* $\mathbf{a}[i] \in P(s,i)$*, a state* $s' = T(s, \mathbf{a})$*.*
- *for each* $i \in \mathsf{Ag}$*,* $\sim_i \subseteq S \times S$ *is an equivalence relation dubbed* indistinguishability relation*.*
- $\mathcal{L} : S \rightarrow 2^{\mathsf{Ap}}$ *is the labeling function, assigning each state* s *to a (possibly empty) subset of* Ap*.*

Given an iCGS M, we say that M is a CGS when the indistinguishability relation $\sim_i$ is the identity for any $i \in \mathsf{Ag}$.

A path ρ is an infinite sequence of states of M, $\rho = s_1, s_2, s_3, \ldots$ respecting the following constraints: for every $i \geq 1$, there is a joint action $\mathbf{a}$ such that $T(s_i, \mathbf{a}) = s_{i+1}$. We denote paths by ρ, τ, and π. An history h is a finite prefix of some path ρ. We use H to denote the set of histories. Given two histories h and h', we say that h and h' are indistinguishable for the agent i (denoted by $h \equiv_i h'$) when they have the same length m and $\langle h_j, h'_j\rangle \in \sim_i$ for all $j \leq m$.

Definition 4. *A uniform strategy (strategy for short) for an agent* i *is a function* $\sigma_i : H \rightarrow Act_i$ *such that:*

1. *for every $h \in H$ we have that $\sigma_i(h) \in P(last(h), i)$.*
2. *For every pair of histories h and h', $h \equiv_i h'$ implies $\sigma_i(h) = \sigma_i(h')$.*

A strategy is memoryless when for every pair of histories h and h' we have that $last(h) = last(h')$ implies $\sigma(h) = \sigma(h')$.

As usual, we can see a memoryless strategy for an agent as a function whose domain is the set of states of the iCGS and whose co-domain is the set of agents actions.

Given a coalition Γ, a uniform strategy for Γ, or simply Γ-strategy, is a collection σ_Γ of uniform strategies comprising one strategy σ_i for each $i \in \Gamma$. Given a path ρ, we say that ρ is σ_Γ-*compatible* iff for every $j \geq 1$, $\rho_{j+1} = T(\rho_j, \mathbf{a})$ for some joint action $\mathbf{a}$ such that for every $i \in \Gamma$, $\mathbf{a}[i] = \sigma_i(\rho_{\leq j})$, and for every $k \in \overline{\Gamma}$, $\mathbf{a}[k] \in P(\rho_j, k)$. We denote with $Out(s, \sigma_\Gamma)$ the set of all σ_Γ-compatible paths whose first state is s.

Finally, given a natural number $n \leq \mathsf{Ag}$, and two coalitions Γ and Δ we define $\mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta} = \{\Pi \in 2^{\mathsf{Ag}} \mid \Gamma \cup \Delta \subseteq \Pi\}$, $\mathsf{Ag}^{\rightarrow}_{\Gamma,\Delta} = \{\Pi \in 2^{\mathsf{Ag}} \mid \Gamma \subseteq \Pi \wedge \Pi \subseteq \overline{\Delta}\}$, $\mathsf{Ag}^{\leq}_{n} = \{\Gamma \in 2^{\mathsf{Ag}} \mid |\Gamma| \leq n\}$, and $\mathsf{Ag}^{\geq}_{n} = \{\Gamma \in 2^{\mathsf{Ag}} \mid |\Gamma| \geq n\}$.

We now have all the needed ingredients to specify CATL semantics.

Definition 5. *Given a iCGS M, a state s of M, and a state formula φ, the satisfaction relation $M, s \models \varphi$ is defined by structural induction on φ as follows:*

- $M, s \models p$ *iff* $p \in \mathcal{L}(s)$;
- $M, s \models \neg\varphi_1$ *iff it is not the case that* $M, s \models \varphi_1$ *(noted* $M, s \not\models \varphi$*)*;
- $M, s \models \varphi_1 \vee \varphi_2$ *iff* $M, s \models \varphi_1$ *or* $M, s \models \varphi_2$;
- $M, s \models \langle\langle\Gamma\rangle\rangle\psi$ *iff there is a Γ-strategy σ_Γ such that for all $\rho \in Out(s, \sigma_\Gamma)$ we have that $M, \rho \models \psi$;*
- $M, s \models \langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\psi$ *iff there is a coalition $\Pi \in \mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}$ and there is a Π-strategy σ_Π such that for all $\rho \in Out(s, \sigma_\Pi)$ we have that $M, \rho \models \psi$;*
- $M, s \models \langle\langle\Gamma \rightarrow \Delta\rangle\rangle\psi$ *iff there is a coalition $\Pi \in \mathsf{Ag}^{\rightarrow}_{\Gamma,\Delta}$ and there is a Π-strategy σ_Π such that for all $\rho \in Out(s, \sigma_\Pi)$, $M, \rho \models \psi$;*
- $M, s \models \langle\langle{\leq_n}\rangle\rangle\psi$ *iff there is a coalition $\Gamma \in \mathsf{Ag}^{\leq}_{n}$ and there is a Γ-strategy σ_Γ such that for all $\rho \in Out(s, \sigma_\Gamma)$, $M, \rho \models \psi$;*
- $M, s \models \langle\langle{\geq_n}\rangle\rangle$ *iff there is a coalition $\Gamma \in \mathsf{Ag}^{\geq}_{n}$ and there is a Γ-strategy σ_Γ such that for all $\rho \in Out(s, \sigma_\Gamma)$ we have that $M, \rho \models \psi$.*

Given a iCGS M, a path ρ of M, and a path formula ψ, the satisfaction relation is defined as follows:

- $M, \rho \models \mathsf{X}\varphi$ *iff* $M, \rho_2 \models \varphi$
- $M, \rho \models \varphi_1 \,\mathsf{U}\, \varphi_2$ *iff there is an $i \geq 1$ such that $M, \rho_i \models \varphi_2$ and $M, \rho_j \models \varphi_1$ for all $1 \leq j < i$;*
- $M, \rho \models \varphi_1 \,\mathsf{R}\, \varphi_2$ *iff either $M, \rho_i \models \varphi_2$ for all $i \geq 1$ or there is an $i \geq 1$ such that $M, \rho_i \models \varphi_1$ and $M, \rho_j \models \varphi_2$ for all $1 \leq j \leq i$.*

The memoryless satisfaction relation $\models_r$ is obtained by substituting "memoryless strategy" to "strategy" in the clauses for the strategic operators. For a CATL formula φ we write $M \models \varphi$ and we say that M is a model of φ whenever $M, s_I \models \varphi$.

We can now define the model checking problem for CATL.

Definition 6. *Given a iCGS M and a CATL formula φ, the model checking problem is solved by determining whether $M \models \varphi$.*

Remark 1. Since each ATL formula is a CATL formula and since the model-checking problem for ATL over iCGS with uniform strategies is undecidable, we have the same result for CATL.

In what follows, we show that CATL is a variant of ATL, that is: every CATL formula φ can be expressed as an ATL formula φ' and these two formulas are semantically equivalent.

3.3 From CATL to ATL

There is an intuitive translation from CATL formulas to ATL formulas. For instance, suppose that $\langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\psi$ is a CATL formula such that ψ does not contain any occurrence of one of the new strategic operators that we introduced. Given a state s of a model M, we have that $M,s \models \langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\psi$ if and only if $M,s \models \langle\langle\Pi_1\rangle\rangle\psi \vee \cdots \vee \langle\langle\Pi_n\rangle\rangle\psi$ where the $\Pi_1,\ldots,\Pi_n$ are all the coalitions in Ag that contains all elements of both Γ and Δ. Following this intuition, we can define $(-)^\bullet$ as follows:

$$
\begin{array}{rcl}
(\top)^\bullet & = & \top \\
(p)^\bullet & = & p \\
(\neg\varphi)^\bullet & = & \neg(\varphi)^\bullet \\
(\varphi_1 \vee \varphi_2)^\bullet & = & (\varphi_1)^\bullet \vee (\varphi_2)^\bullet \\
(\langle\langle\Gamma\rangle\rangle\psi)^\bullet & = & \langle\langle\Gamma\rangle\rangle(\psi)^\bullet \\
(\langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\psi)^\bullet & = & \bigvee_{\Pi\in\mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}} \langle\langle\Pi\rangle\rangle(\psi)^\bullet \\
(\langle\langle\Gamma\rightarrow\Delta\rangle\rangle\psi)^\bullet & = & \bigvee_{\Pi\in\mathsf{Ag}^{\rightarrow}_{\Gamma,\Delta}} \langle\langle\Pi\rangle\rangle(\psi)^\bullet \\
(\langle\langle{\leq}n\rangle\rangle\psi)^\bullet & = & \bigvee_{\Gamma\in\mathsf{Ag}^{\leq}_{n}} \langle\langle\Gamma\rangle\rangle(\psi)^\bullet \\
(\langle\langle{\geq}n\rangle\rangle\psi)^\bullet & = & \bigvee_{\Gamma\in\mathsf{Ag}^{\geq}_{n}} \langle\langle\Gamma\rangle\rangle(\psi)^\bullet \\
(\mathsf{X}\varphi)^\bullet & = & \mathsf{X}(\varphi)^\bullet \\
(\varphi_1 \,\mathsf{U}\, \varphi_2)^\bullet & = & (\varphi_1)^\bullet \,\mathsf{U}\, (\varphi_2)^\bullet \\
(\varphi_1 \,\mathsf{R}\, \varphi_2)^\bullet & = & (\varphi_1)^\bullet \,\mathsf{R}\, (\varphi_2)^\bullet
\end{array}
$$

From the above, we can notice that, given the translation $(-)^\bullet$ and a CATL formula φ, we can obtain an exponentially bigger representation of φ in ATL (in the worst case).

By using our translation we can obtain the following result.

Proposition 1. *For every CATL formula φ, for every iCGS M, and state s of M, we have that:*

$$M,s \models \varphi \textit{ if and only if } M,s \models (\varphi)^\bullet$$

Proof. The proof is by induction on the size of φ. When φ is an atom, the result is clear. When the main connective of φ is boolean, the result follows directly by induction

hypothesis. All the cases for the strategic operators follow exactly the same pattern, therefore we detail only some of them.
If φ is $\langle\langle\Gamma\rightarrow\leftarrow\Delta\rangle\rangle\psi$ and ψ is $\mathsf{X}\varphi_1$:

For the $(\Rightarrow)$-direction, suppose that $M,s \models \langle\langle\Gamma\rightarrow\leftarrow\Delta\rangle\rangle\mathsf{X}\varphi_1$. This means that there is a coalition $\Pi \in \mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}$ and a Π-strategy σ_Π such that for all $\rho \in \mathit{Out}(s,\sigma_\Pi)$ we have that $M,\rho_2 \models \varphi_1$. By induction hypothesis, we obtain that $M,\rho_2 \models (\varphi_1)^\bullet$, and this for every $\rho \in \mathit{Out}(s,\sigma_\Pi)$. We thus conclude that $M,s \models \langle\langle\Pi\rangle\rangle\mathsf{X}(\varphi_1)^\bullet$, and by the semantics of $\vee$ we deduce that $M,s \models \bigvee_{\Pi\in\mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}} \langle\langle\Delta\rangle\rangle\mathsf{X}(\varphi)^\bullet$ which is exactly $(\varphi)^\bullet$.
For the $(\Leftarrow)$-direction. Suppose that $M,s \models (\varphi)^\bullet$. By the definition of $(-)^\bullet$, this means that $M,s \models \bigvee_{\Pi\in\mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}} \langle\langle\Pi\rangle\rangle\mathsf{X}(\varphi_1)^\bullet$ which is the same as $M,s \models \langle\langle\Pi\rangle\rangle\mathsf{X}(\varphi_1)^\bullet$ for some $\Pi \in \mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}$. We thus deduce that there is a Π-strategy σ_Π such that $M,\rho_2 \models (\varphi_1)^\bullet$ for all $\rho \in \mathit{Out}(s,\sigma_\Pi)$. By induction hypothesis $M,\rho_2 \models \varphi_1$ for any $\rho \in \mathit{Out}(s,\sigma_\Pi)$. Since σ_Π is a Π-strategy and $\Pi \in \mathsf{Ag}^{\rightarrow\leftarrow}_{\Gamma,\Delta}$, we can conclude that $M,s \models \langle\langle\Gamma\rightarrow\leftarrow\Delta\rangle\rangle\mathsf{X}\varphi_1$ as we wanted.

If φ is $\langle\langle\leq_n\rangle\rangle\psi$ and ψ is $\varphi_1\,\mathsf{R}\,\varphi_2$:

For the $(\Rightarrow)$ direction, suppose that $M,s \models \langle\langle\leq_n\rangle\rangle\varphi_1\,\mathsf{R}\,\varphi_2$, then there is a coalition $\Gamma \in \mathsf{Ag}^{\leq}_n$ and a Γ-strategy σ_Γ such that for all $\rho \in \mathit{Out}(s,\sigma_\Gamma)$ either $M,\rho_i \models \varphi_2$ for all $i \geq 1$ or there is a $j \geq 1$ such that $M,\rho_j \models \varphi_1$ and $M,\rho_i \models \varphi_2$ for all $1 \leq i \leq j$. We use the induction hypothesis, and we conclude that for all $\rho \in \mathit{Out}(s,\sigma_\Gamma)$ we either have that $M,\rho_i \models (\varphi_2)^\bullet$ for all $i \geq 1$, or $M,\rho_j \models (\varphi_1)^\bullet$ for some $j \geq 1$, and $M,\rho_i \models (\varphi_2)^\bullet$ for all $1 \leq i \leq j$. We thus conclude that $M,s \models \langle\langle\Gamma\rangle\rangle(\varphi)^\bullet_1\,\mathsf{R}\,(\varphi_2)^\bullet$ and by the semantics of $\vee$ we deduce that $M,s \models \bigvee_{\Gamma\in\mathsf{Ag}^{\leq}_n} \langle\langle\Gamma\rangle\rangle(\varphi)^\bullet_1\,\mathsf{R}\,(\varphi_2)^\bullet$ which is exactly $(\langle\langle\leq_n\rangle\rangle\varphi_1\,\mathsf{R}\,\varphi_2)^\bullet$.
For the $(\Leftarrow)$ direction, suppose that $M,s \models (\varphi)^\bullet$ this means that for some $\Gamma \in \mathsf{Ag}^{\leq}_n$ $M,s \models \langle\langle\Gamma\rangle\rangle(\varphi_1)^\bullet\,\mathsf{R}\,(\varphi_2)^\bullet$, thus there is a Γ-strategy σ_Γ such that for all $\rho \in \mathit{Out}(s,\sigma_\Gamma)$ we either have that $M,\rho_i \models (\varphi_2)^\bullet$ for all $i \geq 1$, or there is a $j \geq 1$ such that $M,\rho_j \models (\varphi_1)^\bullet$, and $M,\rho_i \models (\varphi_1)^\bullet$ for all $1 \leq i \leq j$. We use the induction hypothesis, and we conclude that for all $\rho \in \mathit{Out}(s,\sigma_\Gamma)$ we either have that $M,\rho_i \models \varphi_2$ or there is a $j \geq 1$ such that $M,\rho_j \models \varphi_1$ and $M,\rho_i \models \varphi_1$ for all $1 \leq i \leq j$. Since $\Gamma \in \mathsf{Ag}^{\leq}_n$ we conclude that $M,s \models \langle\langle\leq_n\rangle\rangle\varphi_1\,\mathsf{R}\,\varphi_2$ as we wanted.

If φ is $\langle\langle\geq_n\rangle\rangle\psi$ and ψ is $\varphi_1\,\mathsf{U}\,\varphi_2$:

For the $(\Rightarrow)$ direction, suppose that $M,s \models \langle\langle\geq_n\rangle\rangle\varphi_1\,\mathsf{U}\,\varphi_2$ thus there is a coalition $\Gamma \in \mathsf{Ag}^{\geq}_n$ and a Γ-strategy σ_Γ such that for all $\rho \in \mathit{Out}(s,\sigma)$ there is a $j \geq 1$ such that $M,\rho_j \models \varphi_2$ and $M,\rho_i \models \varphi_1$ for all $1 \leq i < j$. We use induction hypothesis and conclude that for all $\rho \in \mathit{Out}(s,\sigma_\Gamma)$ there is a $j \geq 1$ such that $M,\rho_j \models (\varphi_2)^\bullet$ and $M,\rho_i \models (\varphi_1)^\bullet$ for all $1 \leq i < j$. Thus, we have that $M,s \models \langle\langle\Gamma\rangle\rangle(\varphi_1)^\bullet\,\mathsf{U}\,(\varphi_2)^\bullet$ and by the semantics of $\vee$ we deduce that $M,s \models \bigvee_{\Gamma\in\mathsf{Ag}^{\geq}_n} \langle\langle\Gamma\rangle\rangle(\varphi)^\bullet_1\,\mathsf{U}\,(\varphi_2)^\bullet$ which is exactly $(\langle\langle\geq_n\rangle\rangle\varphi_1\,\mathsf{U}\,\varphi_2)^\bullet$.
For the $(\Leftarrow)$ direction, suppose that $M,s \models (\varphi)^\bullet$, thus $M,s \models \langle\langle\Gamma\rangle\rangle(\varphi_1)^\bullet\,\mathsf{U}\,(\varphi_2)^\bullet$ for some $\Gamma \in \mathsf{Ag}^{\geq}_n$. This means that there is a Γ-strategy σ_Γ and for all $\rho \in \mathit{Out}(s,\sigma_\Gamma)$

we have that there is a $j \geq 1$ such that $M, \rho_j \models (\varphi_2)^\bullet$ and $M, \rho_i \models (\varphi_1)^\bullet$ for all $1 \leq i < j$. We use induction hypothesis and we conclude that for all $\rho \in Out(s, \sigma_\Gamma)$ we have that there is a $j \geq 1$ such that $M, \rho_j \models \varphi_2$ and $M, \rho_i \models \varphi_1$ for all $1 \leq i < j$. Since $\Gamma \in \mathsf{Ag}_n^{\geq}$ we have that $M, s \models \langle\langle \geq_n \rangle\rangle \varphi_1 \mathsf{U} \varphi_2$ as we wanted.

□

Given the above result, we can conclude that CATL and ATL have the same expressive power. So, we may derive the following result.

Corollary 1. *The satisfaction relation $\models$ and the memoryless satisfaction relation $\models_r$ coincides over CGSs, that is, for every CATL formula φ, for every CGS M, and state s:*

$$M, s \models \varphi \text{ if and only if } M, s \models_r \varphi$$

We now study the complexity of the model checking problem for CATL with respect to CGSs.

Theorem 1. *Given a formula φ and a CGS M the problem of determining the set $[\![\varphi]\!] = \{s \in S \mid M, s \models \varphi\}$ is in $\Delta_2^P = P^{NP}$ with respect to the size of φ and M.*

Proof. We know that given an ATL formula λ, $[\![\lambda]\!]$ can be computed in polynomial time w.r.t. the size of λ and M. Consider a subformula φ' of φ that contains exactly one strategic operator $\langle\langle \Gamma \rightarrow \leftarrow \Delta \rangle\rangle$, or $\langle\langle \Gamma \rightarrow \Delta \rangle\rangle$, or $\langle\langle \leq_n \rangle\rangle$, or $\langle\langle \geq_n \rangle\rangle$. By Proposition 1 we know that φ' is equivalent to a finite disjunction $\varphi_1 \vee \cdots \vee \varphi_n$ of ATL formulas. Thus given a state s, a certificate consists in an ATL formula φ_i i.e., for the considered φ' checking whether $M, s \models \varphi'$ is in the NP class. We then use the classic bottom-up approach to evaluate each subformula of φ on M: we order the subformulas of φ by their size (in ascending order). For each subformula φ_1 having exactly one of the four above-mentioned strategic operators, we create a new atom p_{φ_1}, we substitute p_{φ_1} to φ_1 in each subformula φ_2 that contains φ_1, and we add p_{φ_1} to the set of satisfied atoms of each state s such that $M, s \models \varphi_1$. This means that we use an NP oracle over a polynomial procedure for each strategic operator in φ and each state s of M. Summing up, the total complexity of determining $[\![\varphi]\!]$ is P^{NP}. □

We now study the complexity of the model checking problem for CATL with respect to iCGSs in which agents use memoryless-strategies.

Theorem 2. *Given a formula φ and an iCGS M the problem of determining the set $[\![\varphi]\!] = \{s \in S \mid M, s \models_r \varphi\}$ is in Δ_3^P with respect to the size of φ and M.*

Proof. We know that the same problem with respect to ATL formulas is in Δ_2^P [21]. We simply remark that, to solve our problem, we can use another NP oracle $|S|$ times to guess the good coalitions needed to satisfy a CATL formula as we do in Theorem 1. From the above the result follows.

3.4 Fragments

In this subsection, we present a fragment of CATL where we have the same complexity of ATL model checking. The intuition behind the resulting approach is to generalize the coalitions involved in the CATL strategic operators. Specifically, we achieve this by replacing $\langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle$ and $\langle\langle{\geq_n}\rangle\rangle$ with $\langle\langle\mathsf{Ag}\rangle\rangle$.

Before doing this, let us recall a classic result that holds in ATL and that will be fundamental in our translation.

Proposition 2. *For any pair of coalitions Γ and Δ, for any iCGS M and state s of M, and for any path formula ψ, we have that, if $\Gamma \subseteq \Delta$ then:*

$$M,s \models \langle\langle\Gamma\rangle\rangle\psi \textit{ implies } M,s \models \langle\langle\Delta\rangle\rangle\psi$$

Proof. See [9].

Let CATL$^{\blacktriangledown}$ be the subset of CATL formulas constructed using all CATL connectives but $\langle\langle\Gamma \rightarrow \Delta\rangle\rangle$ and $\langle\langle{\leq_n}\rangle\rangle$. Let $(-)^\circ$ to be a function from CATL$^{\blacktriangledown}$ formulas to ATL formulas, such that:

$$\begin{array}{rcl}
(\top)^\circ & = & \top \\
(p)^\circ & = & p \\
(\neg\varphi)^\circ & = & \neg(\varphi)^\circ \\
(\varphi_1 \vee \varphi_2)^\circ & = & (\varphi_1)^\circ \vee (\varphi_2)^\circ \\
(\langle\langle\Gamma\rangle\rangle\psi)^\circ & = & \langle\langle\Gamma\rangle\rangle(\psi)^\circ \\
(\langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\psi)^\circ & = & \langle\langle\mathsf{Ag}\rangle\rangle(\psi)^\circ \\
(\langle\langle{\geq_n}\rangle\rangle\psi)^\circ & = & \langle\langle\mathsf{Ag}\rangle\rangle(\psi)^\circ \\
(\mathsf{X}\varphi)^\circ & = & \mathsf{X}(\varphi)^\circ \\
(\varphi_1 \,\mathsf{U}\, \varphi_2)^\circ & = & (\varphi_1)^\circ \,\mathsf{U}\, (\varphi_2)^\circ \\
(\varphi_1 \,\mathsf{R}\, \varphi_2)^\circ & = & (\varphi_1)^\circ \,\mathsf{R}\, (\varphi_2)^\circ
\end{array}$$

We can prove the following result.

Proposition 3. *For every CATL$^{\blacktriangledown}$ formula φ, for every iCGS M, and every state s of M, we have that:*

$$M,s \models \varphi \textit{ if and only if } M,s \models (\varphi)^\circ$$

Proof. The proof is by induction on the size of φ. When φ is an atom, the result is clear. When the main connective of φ is boolean, the result follows directly by induction hypothesis. As in Proposition 1, all the cases for the strategic operators follow exactly the same proof pattern. Thus, we only detail the case in which φ is $\langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\psi$ and ψ is $\mathsf{X}\varphi_1$.

For the $(\Rightarrow)$-direction, suppose that $M,s \models \langle\langle\Gamma{\rightarrow}{\leftarrow}\Delta\rangle\rangle\mathsf{X}\varphi_1$. By the CATL semantics, this means that there is a Π such that $\Gamma \cup \Delta \subseteq \Pi$ and a Π-strategy σ such that for all $\rho \in \mathit{Out}(s,\sigma)$ we have that $M,\rho_2 \models \varphi_1$. By induction hypothesis, we conclude that for all $\rho \in \mathit{Out}(s,\sigma)$ we have that $M,\rho_2 \models (\varphi_1)^\circ$, which is the same as $M,s \models \langle\langle\Pi\rangle\rangle\mathsf{X}(\varphi_1)^\circ$. Since $\Pi \subseteq \mathsf{Ag}$, we conclude by Proposition 2 that $M,s \models \langle\langle\mathsf{Ag}\rangle\rangle\mathsf{X}(\varphi_1)^\circ$ which is exactly $(\langle\langle\Gamma \cup \Delta\rangle\rangle\mathsf{X}\varphi_1)^\circ$.

For the ($\Leftarrow$)-direction, suppose that $M,s \models \langle\langle \mathsf{Ag} \rangle\rangle \mathsf{X}\,(\varphi_1)^\circ$, thus for all $\rho \in Out(s,\sigma)$ we have that $M,\rho_2 \models (\varphi_1)^\circ$ for some Ag-strategy σ. By induction hypothesis, we have that $M,\rho_2 \models \varphi_1$ for all $\rho \in Out(s,\sigma)$ and since $\Gamma \cup \Delta \subseteq \mathsf{Ag}$ we can conclude that $M,s \models \langle\langle \Gamma \rightarrow\leftarrow \Delta \rangle\rangle \mathsf{X}\,\varphi_1$.

□

Since the ATL model checking problem is PTIME [1] with respect to CGS, we immediately obtain the following corollary.

Corollary 2. *For any CATL$^{\blacktriangledown}$ formula φ and any CGS M, the problem of determining $[\![\varphi]\!]$ is in PTIME with respect to the size of φ and M.*

Since the ATL model checking problem is Δ_2^P [21] with respect to iCGS with memoryless strategies, we immediately obtain the following corollary.

Corollary 3. *For any CATL$^{\blacktriangledown}$ formula φ and any iCGS M, the problem of determining $[\![\varphi]\!]$ is in Δ_2^P with respect to the size of φ and M.*

In the following sections, we exemplify the formal machinery introduced so far by providing an example.

4 Interpreted Systems

The semantics of a strategic logic, such as ATL or CATL, can be specified, equivalently, by resorting to either CGSs or Interpreted Systems. As the reader has probably noticed, we have chosen to resort to the first alternative by defining the semantics of CATL through CGSs. This choice is due to the fact that CGSs have a simple and intuitive definition. Essentially, they are directed and labeled graphs in which the edge relation is serial and in which edges are labeled by tuples of agent's actions. As intuitive as this definition is, the MCMAS verification tool operates on interpreted systems. So, for the sake of completeness, we now introduce the formal definition of interpreted systems. Then, in the next section, we will present this latter formal model through an example to guide the reader.

An interpreted system, like a CGS, is a formal description of the computations carried out by a set of agents. More specifically, an interpreted system is given by a set of agents. Each of these agents operates on local states, representing the information that they have about the system under exam. The system itself is represented as the product of the local states of the agents: in any of its local states, an agent can perform a fixed set of actions, and the global state of the system evolves with respect to the product of the actions of all the agents. We now state the formal definitions. First, we define agents.

Definition 7 (Agent). *Let $\mathsf{Ag} = \{1,\ldots,k\}$ be a finite set of agent indexes. An* agent *is a tuple $i = \langle L_i, act_i, P_i, t_i \rangle$, where:*

- *L_i is the finite non-empty set of local states.*

- *Act_i is the finite non-empty set of individual actions. We denote by ACT the product set $\Pi_{i\in \mathsf{Ag}} Act_i$ and we call its elements* joint actions.
- *$P_i : L_i \to (2^{Act_i} \setminus \emptyset)$ is the local protocol function associating to any local state a non-empty set of actions representing the actions available to the agent at that state.*
- *$t_i : L_i \times ACT \to L_i$ is the local transition function. Such function takes an agent's state l and a joint action $\mathbf{a} = \langle a_1, \ldots, a_k \rangle$ and outputs an agent's state. The function $t_i(l_i, \mathbf{a})$ is defined if and only if we have that $\mathbf{a}[i] \in P_i(l_i)$. Remark that, the output of the transition function depends on a joint action $\langle a_1, \ldots, a_k \rangle$, but we only require that the i-th component of this joint action belongs to the set of actions that are available for the considered agent at the considered state.*

By the above definition, an agent i is situated in a local state $l \in L_i$ representing the information it has about the system. At any state, the agent can perform the actions in Act_i according to the protocol function P_i. A joint action determines a change in the state of the agent according to the transition function t_i.

If Ag is a set of agents of length k, a **global state** $s \in G$ is a tuple $s = \langle l_1, \ldots, l_k \rangle$ where each l_i is an i agent's state for $i \leq k$. A **history** is a finite sequence $h = s_1, \ldots, s_n$ of global states. We denote by H_G the set of histories of global states. Two global states s and s', are **equivalent** for the agent i whenever $s[i] = s'[i]$. We denote such notion by $s \sim_i s'$. Two histories h and h' are equivalent for the agent i whenever they have the same length m and $h_j \sim_i h'_j$ for any $j \leq m$.

Definition 8. *Given a set of atomic propositions Ap, an interpreted system is a tuple $I = \langle \mathsf{Ag}, s_0, T, \Pi \rangle$ where Ag is a set of agents, $s_0 \in G$ is the (global) initial state, $T : G \times ACT \to G$ is the global transition function such that $T(s, \mathbf{a}) = s'$ iff for every $i \in \mathsf{Ag}$, $t_i(s[i], \mathbf{a}[i]) = s'[i]$. Finally, $\Pi : G \to 2^{\mathsf{Ap}}$ is the labeling function, associating to any global state an (eventually empty) set of atomic propositions.*

A strategy for an agent i, is a function from the set of local histories H_G to the set of actions act_i of the agent i defined exactly as in Definition 4. Joint strategies and paths that are compatible with joint strategies are also defined as the corresponding notions for iCGSs. The semantics of CATL formulas on interpreted systems is defined exactly as in Definition 5, the only difference is that we use an interpreted system I instead of a iCGS M in such a definition.

5 Train Gate Controller Scenario

In this section, we exemplify the formal apparatus introduced in the previous section through an example. We consider a revised version of the Train Gate Controller by [1,3,7,4] in which there are two trains and a controller. The aim of the two trains is to pass a gate. To do this, they need to coordinate with the controller. The trains are initially placed outside the gate and to ask to go in the gate they need to do a request (action *req*). If the controller accepts the request (actions ac_1 and ac_2, respectively), the train has the grant to pass through the gate. Note that, to perform the physical action of passing through the gate, the train has to select the action *in*. Then, it stays in the gate

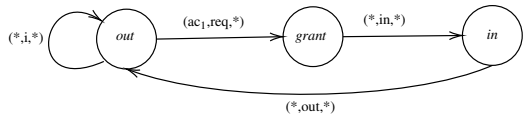

Fig. 1: Local model for train 1 [13].

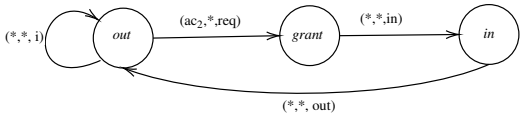

Fig. 2: Local model for train 2 [13].

until it does the action *out*. What we want to show in this game is the fact that the trains need the accordance of the controller to achieve their objectives (*i.e.* to pass the gate).

More formally, this game can be represented as the Interpret System $I = \langle \mathsf{Ag}, s_0, T, \Pi \rangle$, such that:

- $\mathsf{Ag} = \{Train_1, Train_2, Controller\}$;
- $Act_{Train_1} = Act_{Train_2} = \{req, in, out, i\}$, where by action *req* they do a request, by action *in* they go in the gate, by action *out* they go outside the gate, and by action *i* they do nothing. $Act_{Controller} = \{ac_1, ac_2, i\}$, where by action ac_j the *Controller* gives the access to train $j \in \{1,2\}$, and by action *i* it does nothing.

The local model for $Train_1$ is given in Figure 1, the local model for $Train_2$ is given in Figure 2, and the local model of the $Controller$ is given in Figure 3. The global initial state, the transition function, and the labeling function are given in Figure 4. In particular, each global state is represented as a rectangle where the tuple (l_c, l_{t_1}, l_{t_2}) includes the Controller's local state (l_c), the Train 1's local state (l_{t_1}), and the Train 2's local state (l_{t_2}). Furthermore, by the tuple of local states, we can consider as atomic propositions true in each state the names of the local states and, in accordance to them, define the labeling function. Notice that, in the figures, we denote any available action with the symbol $*$.

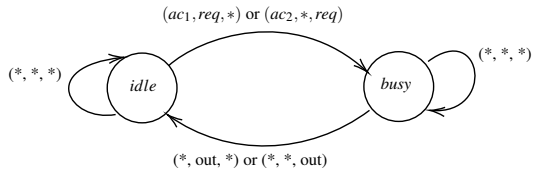

Fig. 3: The local model for the controller [13].

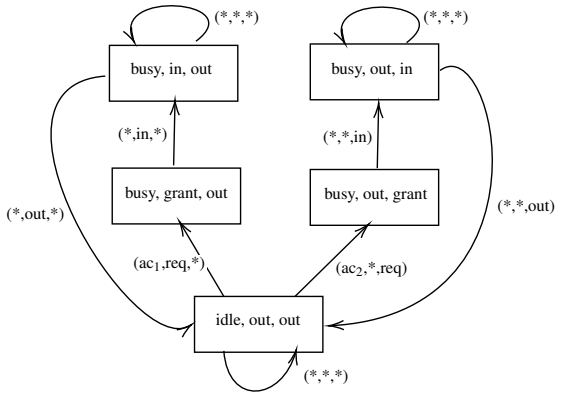

Fig. 4: The interpreted systems IS, where $s_0 = (idle, out, out)$ [13].

The property the Train 1 has a winning strategy to achieve the gate can be represented as follows:

$$\varphi_1 = \langle\langle Train_1 \rangle\rangle \mathsf{F}\, in_1$$

We observe that φ_1 is false since to make the property true the Train 1 needs the agreement of the Controller. By consequence, the property that can be satisfied is the following:

$$\varphi_1 = \langle\langle Train_1, Controller \rangle\rangle \mathsf{F}\, in_1$$

Analysis over coalitions. Considering our example, a possible guideline over the size of the coalitions could be $min : 2$. With such feature, we would enforce the generation of coalitions with at least two agents. This could be guided by the fact that we know that no agent in isolation can achieve its own goals in the train gate controller example. Another possible constraint could be $[Train_1 \rightarrow\leftarrow Controller]$, where we enforce $Train_1$ and $Controller$ to be in coalition. For similar reasons, we may add the constraint $[Train_1 \leftarrow\rightarrow Train_2]$ and enforce the two trains to not be in coalition. Remark that such constraints precisely corresponds to the CATL operators $\langle\langle \Gamma \rightarrow\leftarrow \Delta \rangle\rangle$ and $\langle\langle \Gamma \leftarrow\rightarrow \Delta \rangle\rangle$[3].

6 Verification

In this section, we present the algorithms to solve, in practice, the model checking problem for CATL. To do so, we exploit the MCMAS model checker, and its verification engine for ATL formulas. Note that, as proved previously in the paper (see Proposition 1), CATL and ATL share the same semantics; because of that, the actual verification of CATL formulas can be obtained through the standard verification of ATL ones.

[3] Remark that this operator can be defined in terms of $\langle\langle \Gamma \rightarrow \Delta \rangle\rangle$ see Definition 1.

Once an interpreted system I has been defined, we can unleash our approach. Starting from such model, we can verify for which coalitions of agents an ATL formula φ is verified in I. Specifically, differently from standard ATL, we do not want to explicitly state each Γ coalition in φ; instead, we want to automatically generate such coalitions. Naturally, not all coalitions are always of interest; this of course depends on the domain of use. Thus, even though a coalition makes a formula φ satisfied by a model I, it does not necessarily mean such coalition is a good one (*i.e.*, a usable one).

Now, we move forward and present how our approach uses the pre-processing steps to perform the actual formal verification on MAS. Specifically, this is obtained through two algorithms. Let us explore them in detail.

Algorithm 1 GenCoalitions(Ag, *min*, *max*, T, S)

1: $\Gamma_{valid} = \emptyset$
2: **for** $k \in [min, max]$ **do**
3: **for** $\Gamma \in \Gamma_k^{\mathsf{Ag}}$ **do**
4: **if** $\exists[\Gamma_1 \rightarrow\!\!\leftarrow \Gamma_2] \in T : \{\Gamma_1, \Gamma_2\} \not\subseteq \Gamma \wedge \{\Gamma_1, \Gamma_2\} \cap \Gamma \neq \emptyset$ **then continue**
5: **if** $\exists[\Gamma_1 \longleftrightarrow \Gamma_2] \in S : \{\Gamma_1, \Gamma_2\} \subseteq \Gamma$ **then continue**
6: Add Γ to Γ_{valid}
7: **return** Γ_{valid}

Algorithm 2 MCMAS$_{co}$(I, φ, *min*, *max*, T, S, *num_coalitions*)

1: $\mathsf{Ag} = GetAgents(I)$
2: $\Gamma_{good} = \emptyset$
3: $\Gamma_{valid} = GenCoalitions(\mathsf{Ag}, min, max, T, S)$
4: **for** $\Gamma \in \Gamma_{valid}$ **do**
5: **if** $I \models \varphi^{\Gamma}$ **then**
6: Add Γ to Γ_{good}
7: **if** $|\Gamma_{good}| = num\_coalitions$ **then**
8: **return** Γ_{good}
9: **return** Γ_{good}

Algorithm 1. It reports the steps required to generate a set of valid coalitions, *i.e.*, coalitions that respect the user's guidelines. Algorithm 1 takes in input the set of agents Ag, and the user's guidelines, such as the minimum/maximum number of agents to be in the coalitions, and the set of agents that have to (resp., cannot) stay in the same coalition T (resp., S). At line 1, the set of valid coalitions is initialized to the empty set. Then, at line 2, a value k is selected for any integer value between *min* and *max* (both included). After that, the algorithm loops over all possible values of k (lines 3-6); with k denoting the current size of the considered coalitions. Naturally, there are multiple k

coalitions that can be formed over a set Ag of agents. In more detail, they correspond to all possible combinations of k agents taken from the set Ag; this is expressed by the set Γ_k^{Ag}. For each of these coalitions, the algorithm checks whether the constraints hold or not. First, it checks if all agents that are required to be together in the coalition are as such (line 4). If for at least one couple $[\Gamma_1 \rightarrow\leftarrow \Gamma_2]$, we find only one subset of agents in the coalition, then we skip to the next possible coalition to evaluate. In the same way, the algorithm checks for the agents that are not meant to be together (lines 5). This again is achieved by checking whether for some couple both the subset of agents are in the coalition. If that is the case, then the algorithm moves on to the next coalition to evaluate. At the end of the algorithm, the set Γ_{valid} contains all coalitions respecting the user's guidelines.

Algorithm 2. It performs the actual verification considering all valid agents' coalitions. Algorithm 2 takes in input the model I, the ATL formula to verify φ, and the user's guidelines. At line 1, the set of agents is extracted from I. These are the agents involved in the model. At line 2, the set of good coalitions is initialized to the empty set. By the end of the algorithm, such set will contain the coalitions that respect the user's guidelines and make φ satisfied in I. At line 3, Algorithm 1 is called. In this step, all valid coalitions respecting the user's guidelines are returned. After that, the algorithm loops over such valid coalitions (lines 4-8). For each of them, the model checking is performed (line 5). In here, with φ^{Γ} we denote φ where the coalition has been replaced with the currently selected one (*i.e.*, Γ). If the model checking returns true, *i.e.*, model I satisfies formula φ^{Γ}, then Γ is added to the set of good coalitions Γ_{good} (line 6). After that, if the number of required coalitions has been found (the number of good coalitions in Γ_{good} is equal to *num_coalitions*), then the coalitions are returned (lines 7-8). Otherwise, the algorithm evaluates all the valid coalitions, and returns the good ones at the end (line 9). Note that, in Algorithm 2, we only show the case with one strategic operator in φ, that is, only one Γ coalition is replaced in φ. We decided to do so in order to improve the readability of the procedure. However, in case multiple Γ coalitions are used, the same reasoning is followed, where for each one of them a set of valid coalitions is generated (using Algorithm 1). Then, instead of performing model checking only once (Algorithm 2, line 5), the algorithm would perform the latter for every possible permutation.

Remark 2. Given a CATL formula φ and a model I, to solve the model checking problem through our procedure (Algorithm 2), we need to set *num_coalitions* to 1. This is done to enforce the algorithm to terminate as soon as a good coalition is found (in accordance to the CATL strategic operators' semantics). Furthermore, given the CATL strategic operators involved in the formula φ, we need to determine the values for the different parameters: *min*, *max*, T, and S. On one hand, suppose that we have a CATL formula $\varphi = \langle\langle \leq_n \rangle\rangle \mathsf{X}\, p$ (resp., $\varphi = \langle\langle \geq_n \rangle\rangle \mathsf{X}\, p$), then the parameters passed to Algorithm 2 are set as follows: $min = 0$, $max = n$, $T = \emptyset$, and $S = \emptyset$ (resp., $min = n$, $max = |\mathsf{Ag}|$, $T = \emptyset$, and $S = \emptyset$). On the other hand, suppose that we have a CATL formula $\varphi = \langle\langle \Gamma_1 \rightarrow\leftarrow \Gamma_2 \rangle\rangle \mathsf{X}\, p$ (resp., $\varphi = \langle\langle \Gamma_1 \leftarrow\rightarrow \Gamma_2 \rangle\rangle \mathsf{X}\, p$), then the parameters passed to Algorithm 2 are set as follows: $min = 0$, $max = |\mathsf{Ag}|$, $T = \{[\Gamma_1 \rightarrow\leftarrow \Gamma_2]\}$, and $S = \emptyset$ (resp., $min = 0$, $max = |\mathsf{Ag}|$, $T = \emptyset$, and $S = \{[\Gamma_1 \leftarrow\rightarrow \Gamma_2]\}$).

7 Implementation

A prototype of our approach has been implemented in Python[4]. The prototype gets in input an interpreted system I, specified in terms of an ISPL file (the formalism supported by the MCMAS model checker), an ATL formula to verify φ, and generates all coalitions of agents which make $I \models \varphi$. To understand the tool, first, we need to describe its pillar components.

The model checker we use is MCMAS [16], which is the *de facto* standard model checker of strategic properties on MAS. MCMAS expects in input an interpreted system specified as an ISPL file. In such a file, the interpreted system is defined along with the formal property of interest to verify. From the viewpoint of a MCMAS user, our tool can be seen as an extension of MCMAS that allows the user to, not only perform the verification of ATL properties as usual, but to extract which coalitions of agents make such properties verified in the model.

Since MCMAS expects a fully instantiated ATL formula, in order to extract which coalitions of agents are good candidates, our tool performs a pre-processing step. In such step, as described previously in the paper, all coalitions which follow the user's guidelines are generated (Algorithm 1) and tested on MCMAS (Algorithm 2). In each run, MCMAS returns the boolean result corresponding to the satisfaction of the ATL formula over the interpreted system. The coalitions for which MCMAS returns a positive verdict are then presented as output to the user. To help the reader to understand the whole machinery see Figure 5.

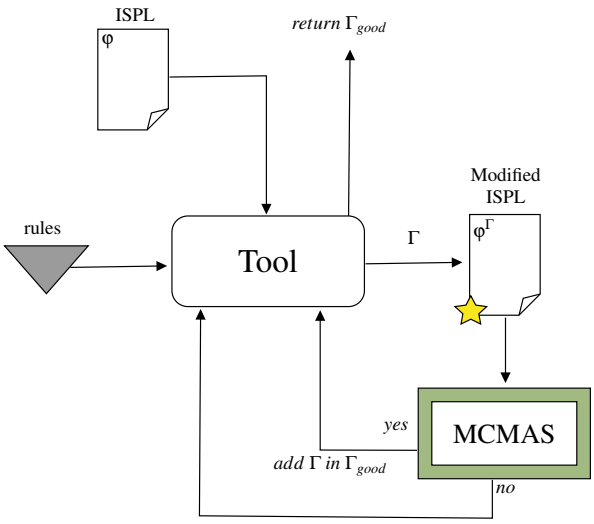

Fig. 5: Overview of the tool.

[4] https://github.com/AngeloFerrando/mcmas-multi-coalitions

The generation of all agents' coalitions has been implemented in Python, as well as its enforcement over the ISPL file. In fact, for each coalition following the user's guidelines, our tool updates the ISPL in the following way. Considering Algorithm 2, this step is implicitly performed in line 5, where the model checking is performed. However, at the implementation level, the actual verification through MCMAS requires to explicitly modify the ISPL file w.r.t. the Γ coalition of interest (*i.e.*, each coalition generated by Algorithm 1). To achieve this technical step, first, the tool searches all occurrences of Γ coalitions in the ISPL file. This can be done by looking for the `groups` keyword (which is the one used in MCMAS to define the agents belonging to each coalition used in the ATL formula). After that, the tool replaces each coalition with a coalition following the user's guidelines. Naturally, in case of multiple Γ coalitions in the ATL formula, all possible permutations of valid coalitions are considered. Once the ISPL file has been properly modified with valid coalitions, MCMAS is called to perform the actual verification.

8 Experiments

We tested our tool over the train gate controller scenario, on a machine with the following specifications: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4 cores 8 threads, 16 GB RAM DDR4. We carried out various experiments on our running example. But, we have not only considered the case with two trains. Instead, we experimented with larger number of trains as well, to better evaluate our tool's performance. Tables 1–3 report the results we obtained.

N	-	≥ 2	≥ 3	≥ 4	≤ 2	≤ 3	≤ 4	$\exists!i.[T_i \rightarrow\leftarrow C]$	$\forall i.[T_i \rightarrow\leftarrow C]$	$\exists!(i,j).[Ti \leftarrow\rightarrow Tj]$	$\forall(i,j).[Ti \leftarrow\rightarrow Tj]$
2	3	3	1	0	2	3	3	2	1	2	2
4	15	15	11	5	4	10	14	8	1	11	4
6	63	63	57	42	6	21	41	32	1	47	6
8	255	255	247	219	8	36	92	128	1	191	8
10	1023	1023	1013	968	10	55	175	512	1	767	10

Table 1: Number of good coalitions generated in our experiments. 1^{st} column reports number of trains. 2^{nd} column, no guidelines are given. 3^{rd} to 5^{th} column minimum number of agents per coalition is required. 6^{th} to 8^{th} column maximum number of agents per coalition is required. 9^{th} to 12^{th} columns guidelines on which agents can stay (or not) in coalition with [13].

Let us start with Table 1. It contains the number of coalitions we found through experiments. In more detail, the table is so structured. The first column reports the number of trains used in the experiments (from 2 to 10 trains). Then, the rest of the columns correspond to the results we get w.r.t. some specific guidelines. Going from left to right. First, we find the case where no guidelines have been passed to the tool. In such case, the tool reports all good coalitions, without any filter. This would correspond to a scenario where we would not have any sort of resource limitation and to group agents. Then, we have three different scenarios where we set the minimum number of

agents per coalition (*i.e.*, we pass the *min* guideline). We do so for *min* equals to 2, 3, and 4. That is, we request only coalitions containing at least 2, 3, and 4 agents, respectively. Here, we can note how with *min* : 2, the number of coalitions does not change w.r.t. the case with no guidelines. This is due to the fact that, as expected, no coalitions with less than 2 agents can satisfy the property of interest; which we remind being $\varphi = \langle\langle\Gamma\rangle\rangle F\, in$. Instead, the other two cases have a fewer number of coalitions. This does not come as a surprise, since we are requesting only larger coalitions (we filter out all coalitions with 2 and 3 agents, respectively). After that, we find similar cases, where instead the *max* guideline is used. First, by enforcing the maximum number of agents in each coalition to be 2, then 3, and finally 4. W.r.t. the previous cases, here we can note how the choice of limiting the maximum number of agents in the coalitions is much more effective in reducing the number of good coalitions proposed. This again is reasonable, because we are filtering out the larger coalitions. Finally, we find the last four columns, which are focused on guidelines on which agents can stay with whom in the coalitions. First, we find the case where we request one single train to be in coalition with the controller. Note that, we do not decide such train *a priori*; it can be any of the available trains. In such case, the number of good coalitions is reduced, but not too much. This is due to the fact that requesting only one train to be in coalition with the controller is not a strong guideline (indeed, other trains can be in coalition as well). Then, the next case consists in requesting all trains to be in coalition with the controller. In this case, we obtain only one good coalition (no matter the number of trains). Since we are requesting all agents to be in coalition, this result is in line with the expectations. In the second to last column, we find a case where we request two trains not to be in coalition. As before, we are not interested in which trains, as long as only two are required not to be in coalition. As expected, this guideline does not affect much the number of good coalitions generated. Indeed, asking to not having just two trains in coalition does not filter out many viable alternatives. Last column presents the same scenario, but where all trains are requested to not be in the same coalition. So, each train cannot collaborate with any other train. This produces a number of coalitions equivalent to the number of trains used in the experiments. This again does not come as a surprise, since the only possible good coalitions are the ones with one train and the controller (no other trains involved).

N	-	≥ 2	≥ 3	≥ 4	≤ 2	≤ 3	≤ 4	$\exists!i.[T_i \rightarrow\leftarrow C]$	$\forall i.[T_i \rightarrow\leftarrow C]$	$\exists!(i,j).[Ti \leftarrow\rightarrow Tj]$	$\forall(i,j).[Ti \leftarrow\rightarrow Tj]$
2	0,06	0,03	0,01	0,00018	0,05	0,05	0,05	0,03	0,01	0,04	0,04
4	0,28	0,23	0,15	0,06	0,13	0,22	0,36	0,13	0,02	0,2	0,09
6	8,87	8,33	6,88	4,62	1,99	4,34	6,9	4,45	0,08	6,68	0,89
8	53,41	51,11	47,4	38,88	4,5	12,49	24,69	25,18	0,1	37,99	1,77
10	382,18	380,66	369,09	340,04	12,03	40,58	99,93	186,9	0,19	304,06	3,94

Table 2: Execution time (in seconds) to generate the set of good coalitions in our experiments. 1^{st} column reports number of trains. 2^{nd} column, no guidelines are given. 3^{rd} to 5^{th} column minimum number of agents per coalition is required. 6^{th} to 8^{th} column maximum number of agents per coalition is required. 9^{th} to 12^{th} columns guidelines on which agents can stay (or not) in coalition with [13].

Moving on with Table 2, we find the same kind of experiments of Table 1. Nonetheless, instead of reporting the number of good coalitions generated, Table 2 reports the execution time required to extract such coalitions. The execution time comprises both the generation of the valid coalitions, and their verification through MCMAS. The columns are the same as in Table 1, but we can observe how much time the tool required to extract the coalitions. Naturally, we can observe that stronger are the guidelines, less is the execution time (since less are the valid coalitions that need to be verified in MCMAS). One important aspect to point out is that our experiments required less than 1 minute when considering the scenarios with at most 8 trains and less than 6 minutes (or so) for 10 trains. This is encouraging, since our approach handles even scenarios where the resulting model is far from being trivial (or small).

N	$\langle\langle\geq_0\rangle\rangle$	$\langle\langle\geq_2\rangle\rangle$	$\langle\langle\geq_3\rangle\rangle$	$\langle\langle\geq_4\rangle\rangle$	$\langle\langle\leq_2\rangle\rangle$	$\langle\langle\leq_3\rangle\rangle$	$\langle\langle\leq_4\rangle\rangle$	$\langle\langle T_1 \rightarrow \leftarrow C\rangle\rangle$	$\langle\langle T \rightarrow \leftarrow C\rangle\rangle$	$\langle\langle T_1 \leftarrow \rightarrow T_2\rangle\rangle$
2	0,06	0,03	0,01	0,00018	0,05	0,05	0,05	0,03	0,01	0,04
4	0,12	0,05	0,05	0,04	0,12	0,12	0,12	0,05	0,02	0,09
6	1,16	0.47	0,41	0,32	0,99	0,99	0,99	0,48	0,08	0,97
8	1,28	0,87	0,76	0,65	1,95	1,95	1,85	0,86	0,1	1,84
10	5,02	2,06	1,75	1,55	4,35	4,08	4,02	2,19	0,19	4,26

Table 3: Execution time (in seconds) to verify CATL formulas in our experiments. 1^{st} column reports number of trains. 2^{nd} column, no guidelines are given. 3^{rd} to 5^{th} column minimum number of agents per coalition is required. 6^{th} to 8^{th} column maximum number of agents per coalition is required. 9^{th} to 11^{th} columns guidelines on which agents can stay (or not) in coalition with. Note that T stays for the whole set of trains.

In Table 3, we provide the results for the CATL formula $\varphi = \langle\langle - \rangle\rangle \mathsf{F}\ in_1$, in which for each column we use the corresponding strategic operator. Notice that, for all the cases our tool returns $\top$, but row 2 column 5. In the latter case, the procedure returns $\bot$ because there are no coalitions with more than 4 agents. In fact, in the above mentioned case, we have only two trains and one controller (*i.e.*, the maximum number of agents is 3). Furthermore, we have removed the 12^{th} column since it is semantically equivalent to column 6^{th}. Another relevant aspect is that our procedure performs better in the case of CATL formulas. This is due to the fact that our algorithm needs to search for only one good coalition (*i.e.*, $num\_coalitions = 1$); while in the experiments carried out in Table 1 and Table 2, all good coalitions are returned (*i.e.*, $num\_coalitions = \infty$).

9 Conclusions

In this paper, we have presented a variant of ATL, called CATL, in which we can reason upon coalitions. We have proved that CATL has the same expressive power of ATL, but it is exponentially more succinct than ATL. We also studied the model checking complexity of CATL in case of both imperfect and perfect information. Furthermore we implemented an extension of MCMAS in which the users can characterize the coalitions in the strategy quantifiers. To do this, we have considered coalitions as variables of

the problem. In particular, we have shown how to give the power to a user to handle two main features: the number of agents involved in the coalitions and how to create coalitions by considering who have to play together and who have to play against. This tool is a first stone to develop a more generalized verification approach for MAS.

References

1. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. Belardinelli, F., Lomuscio, A., Malvone, V.: Approximating perfect recall when model checking strategic abilities. In: KR18 (2018)
3. Belardinelli, F., Lomuscio, A., Malvone, V.: An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In: Proceedings of AAAI (2019)
4. Belardinelli, F., Ferrando, A., Malvone, V.: An abstraction-refinement framework for verifying strategic properties in multi-agent systems with imperfect information. Artif. Intell. **316**, 103847 (2023). https://doi.org/10.1016/j.artint.2022.103847, https://doi.org/10.1016/j.artint.2022.103847
5. Belardinelli, F., Jamroga, W., Kurpiewski, D., Malvone, V., Murano, A.: Strategy logic with simple goals: Tractable reasoning about strategies. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 88–94. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/13, https://doi.org/10.24963/ijcai.2019/13
6. Belardinelli, F., Lomuscio, A., Malvone, V., Yu, E.: Approximating perfect recall when model checking strategic abilities: Theory and applications. J. Artif. Intell. Res. **73**, 897–932 (2022). https://doi.org/10.1613/jair.1.12539, https://doi.org/10.1613/jair.1.12539
7. Belardinelli, F., Malvone, V.: A three-valued approach to strategic abilities under imperfect information. In: Calvanese, D., Erdem, E., Thielscher, M. (eds.) Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020. pp. 89–98 (2020). https://doi.org/10.24963/kr.2020/10, https://doi.org/10.24963/kr.2020/10
8. Bloem, R., Chatterjee, K., Jacobs, S., Könighofer, R.: Assume-guarantee synthesis for concurrent reactive programs with partial information. In: TACAS. pp. 517–532 (2015)
9. Demri, S., Goranko, V., Lange, M.: Temporal Logics in Computer Science: Finite-State Systems. Cambridge University Press, USA, 1st edn. (2016)
10. Dima, C., Tiplea, F.: Model-checking ATL under Imperfect Information and PerfectRecall Semantics is Undecidable. Tech. rep., arXiv (2011)
11. Ferrando, A., Malvone, V.: Strategy RV: A tool to approximate ATL model checking under imperfect information and perfect recall. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 1764–1766. ACM (2021). https://doi.org/10.5555/3463952.3464230, https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1764.pdf
12. Ferrando, A., Malvone, V.: Towards the combination of model checking and runtime verification on multi-agent systems. In: Dignum, F., Mathieu, P., Corchado, J.M., de la Prieta, F. (eds.) Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection - 20th International Conference, PAAMS 2022, L'Aquila, Italy, July 13-15, 2022, Proceedings. Lecture Notes in Computer Science, vol.

13616, pp. 140–152. Springer (2022). https://doi.org/10.1007/978-3-031-18192-4_12, https://doi.org/10.1007/978-3-031-18192-4_12
13. Ferrando, A., Malvone, V.: How to find good coalitions to achieve strategic objectives. In: Rocha, A.P., Steels, L., van den Herik, H.J. (eds.) Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22-24, 2023. pp. 105–113. SCITEPRESS (2023). https://doi.org/10.5220/0011778700003393, https://doi.org/10.5220/0011778700003393
14. Ferrando, A., Malvone, V.: Towards the verification of strategic properties in multi-agent systems with imperfect information. In: Agmon, N., An, B., Ricci, A., Yeoh, W. (eds.) Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023. pp. 793–801. ACM (2023). https://doi.org/10.5555/3545946.3598713, https://dl.acm.org/doi/10.5555/3545946.3598713
15. Kupferman, O., Vardi, M.Y.: Module checking revisited. In: CAV'97. pp. 36–47. Springer (1997)
16. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multi-agent systems. Software Tools for Technology Transfer (2015). https://doi.org/10.1007/s10009-015-0378-x, http://dx.doi.org/10.1007/s10009-015-0378-x
17. Mogavero, F., Murano, A., Perelli, G., Vardi, M.: Reasoning about strategies: On the model-checking problem. ACM Trans. Comp. Log. **15**(4), 34:1–34:47 (2014). https://doi.org/10.1145/2631917, http://doi.acm.org/10.1145/2631917
18. Mogavero, F., Murano, A., Perelli, G., Vardi, M.Y.: Reasoning about strategies: on the satisfiability problem. Log. Methods Comput. Sci. **13**(1) (2017). https://doi.org/10.23638/LMCS-13(1:9)2017, https://doi.org/10.23638/LMCS-13(1:9)2017
19. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: FOCS. pp. 746–757 (1990)
20. Reif, J.H.: The complexity of two-player games of incomplete information. J. Comput. Syst. Sci. **29**(2), 274–301 (1984)
21. Schobbens, P.Y.: Alternating-time logic with imperfect recall. Electronic Notes in Theoretical Computer Science **85**(2), 82–93 (2004)