# An Abstraction Technique for the Verification of Multi-Agent Systems against ATL Specifications

**Alessio Lomuscio and Jakub Michaliszyn**
Department of Computing, Imperial College London, UK

**Abstract**

We introduce an abstraction methodology for the verification of multi-agent systems against specifications expressed in alternating-time temporal logic (ATL). Inspired by methodologies such as predicate abstraction, we define a three-valued semantics for the interpretation of ATL formulas on concurrent game structures and compare it to the standard two-valued semantics. We define abstract models and establish preservation results on the three-valued semantics between abstract models and their concrete counterparts. We illustrate the methodology on the large state spaces resulting from a card game.

## 1 Introduction

In logic-based approaches to multi-agent systems (MAS) there is a long tradition concerned with the development and use of formalisms aimed at expressing the *strategic abilities* of agents in a system. This interest goes back to earlier work in philosophical logic where formalisms such as STIT have been developed to analyse what agents can bring about in a multi-agent system (Belnap and Perloff 1990). Against this context the framework of Alternating-Time Temporal Logic (ATL) was developed in the late 90s in theoretical computer science to reason about paths representing particular outcomes in games (Alur, Henzinger, and Kupferman 2002). For example, by means of an ATL formula one can express whether two agents in a game can always enforce a certain state of affairs irrespective of the actions of the others.

The semantics of ATL is traditionally given in terms of concurrent game structures, which are transition-based systems whereby all choices of the agents as well as their strategies are explicitly provided. Concurrent game structures are very close to interpreted systems, a popular semantics for reasoning about knowledge in multi-agent systems (Fagin et al. 1995). Combining the two approaches provides a natural and intuitive setting for the practical verification of MAS epistemic and ATL specifications under incomplete information (Lomuscio and Raimondi 2006b).

One key attractiveness of ATL is that key system properties such as controllability and realisability can be formulated as model checking particular ATL formulas. In the original setting, i.e., without fairness constraints, the algorithm for model checking models against CTL formulas can easily be extended to ATL, thereby resulting in the ATL model checking problem being linear against the size of the model and the formula to be considered. The complexity increases for ATL$^*$ and more powerful logics. Indeed in the past ten years a range of logics which extend ATL have been put forward, ranging from ATL$^+$ to ATL$^*$, to combinations with knowledge, and to the very recent family of strategy logic (Chatterjee, Henzinger, and Piterman 2007; Mogavero et al. 2012), and their complexity analysed. Furthermore, different assumptions can be made on the observability of the states in the game and the states of the players, as well as their ability to recall the moves they played, thereby leading to several possible variants. The precise set-up has considerable consequences on the resulting complexity of the model checking problem. Certain cases, e.g., perfect recall under incomplete information, are known to be undecidable. For more details we refer to (Alur, Henzinger, and Kupferman 2002; Schobbens 2004; Jamroga and Ågotnes 2007; Bulling, Dix, and Chesñevar 2008).

Much is known at theoretical level about these variants. However, *practical* model checking of systems against ATL specifications has somewhat lagged behind. JMOCHA (Alur et al. 2001), a revised version of the original MOCHA model checker for reactive modules against ATL specifications, supports advanced features such as compositional and modular verification. MCMAS (Lomuscio and Raimondi 2006b; Lomuscio, Qu, and Raimondi 2009), a model checker for multi-agent systems, supports epistemic and ATL specifications in the context of incomplete information. However, they both lack abstraction features thereby making them prone to be constrained by the state-explosion problem.

The aim of this paper is to lay the foundations for an abstraction methodology for verifying multi-agent systems with incomplete information against ATL specifications.

**Contribution.** Since our overall aim is to develop under- and over-approximations for multi-agent systems, we begin in Section 2 by developing a novel three-valued semantics for ATL which we show to be a conservative extension of the classic two-valued semantics. In Section 3 we define abstractions of ATL models. Abstraction is conducted modularly both on the local states and the local actions associated with the agents, thereby generating considerably smaller models. We show that the values true and false are preserved in three-valued semantics between abstract and concrete models. We

turn to the model checking problem in Section 4 by exploring its complexity in a three-valued setting and by providing constructive definitions for the initial abstraction and algorithms for successive refinements. We exemplify these constructions in Section 5 by showing how the technique put forward here can reduce the state space considerably. We conclude in Section 6 where we also discuss the future work.

**Related work.** As discussed above, while model checking systems against ATL specifications has been explored in the literature, practical model checking has received considerable less attention. While model checkers such as JMOCHA and MCMAS have been made available, one of the key problems remains the state-space explosion. To the best of our knowledge the only previous paper in the literature tackling abstraction in the context of ATL is (Köster and Lohmann 2012). However, their approach is based on modular interpreted systems where complete information is assumed. By contrast here we deal with the general class of interpreted systems and we work in a more general, three-valued semantics, thereby laying the foundation for predicate abstraction techniques.

# 2 Three-Valued Alternating-Time Temporal Logic

In this section we recall the technical set-up for ATL and provide it with a novel, three-valued semantics.

We assume a set of agents $Ag = \{1, \ldots, m\}$. For simplicity, and without loss of generality, note we do not consider the environment here. We use $\Gamma \subseteq Ag$ to denote subsets of agents. By $\overline{\Gamma}$ we denote the complementation of $\Gamma$, i.e., $Ag \setminus \Gamma$.

We also assume a set of propositional variables $\mathcal{V}$; by $\overline{\mathcal{V}}$ we denote the set of all the literals containing propositional variables from $\mathcal{V}$, i.e., $\overline{\mathcal{V}} = \{q, \neg q \mid q \in \mathcal{V}\}$.

## 2.1 Interpreted systems and associated models

We are interested in the following structures.

**Definition 1** (Interpreted system)**.** *An* interpreted system *is a tuple $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ such that for each agent $i$:*

- *$L_i$ is a set of possible* local states*;*
- *$Act_i$ is a set of possible* local actions*;*
- *$P_i : L_i \to 2^{Act_i} \setminus \{\emptyset\}$ is a* local protocol*;*
- *$t_i \subseteq L_i \times ACT \times 2^{L_i}$ is a* local transition relation *with $ACT = Act_1 \times \cdots \times Act_m$;*
- *$I \subseteq L_1 \times \cdots \times L_m$ is a set of* global initial states*;*
- *$\Pi : L_1 \times \cdots \times L_m \to \overline{\mathcal{V}}$ is a* labelling function *such that for any $q, s$ we either have $q \notin \Pi(s)$ or $\neg q \notin \Pi(s)$.*

Note that the transition relation is normally considered to be a function. We here consider a more liberal definition to allow for a uniform presentation with abstract models (see Section 2.4). For the same reason the definition of a labelling function is more general than usual as it allows a state to be labelled by neither $q$ nor $\neg q$.

For a tuple $t = (t_1, \ldots, t_m)$, by $t.i$ we denote its $i$th element $t_i$, with $i \leq m$. We will use this notation to identify individual local states and local actions.

We associate standard temporal models to interpreted systems.

**Definition 2** (Model)**.** *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$, its* associated model *$M_{IS} = (S, T, I, \Pi)$ is a tuple such that:*

- *$S \subseteq L_1 \times \cdots \times L_m$ is the set of global states reachable via $T$ from the set of initial global states $I \subseteq S$,*
- *$T \subseteq S \times ACT \times S$ is a global transition relation such that $T((l_1, \ldots, l_m), a, (l'_1, \ldots, l'_m))$ iff $t_i(l_i, a, l'_i)$ and $a.i \in P_i(l_i)$ for all $i \in Ag$.*

When $IS$ is clear from the context we write $M$ for $M_{IS}$.

In the following we assume that all locally enabled joint actions can be executed, i.e., that for all global states $s \in S$ and joint actions $a \in ACT$ such that $a.i \in P_i(s.i)$ for all $i \in Ag$, there exists an $s' \in S$ such that $T(s, a, s')$.

## 2.2 Alternating-time temporal logic

Alternating-Time Temporal Logic (ATL) was introduced to reason about agents and their strategies in the context of perfect information (Alur, Henzinger, and Kupferman 2002). In this setting agents' strategies depend on the global state of the system. ATL's semantics has been recast on incomplete information in the context of interpreted systems (see, e.g., (Schobbens 2004)).

The syntax of ATL is given by the following BNF:
$\varphi ::= q \mid \mathrm{tt} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle\Gamma\rangle\!\rangle X\varphi \mid \langle\!\langle\Gamma\rangle\!\rangle(\varphi U \varphi) \mid \langle\!\langle\Gamma\rangle\!\rangle G\varphi$
where $q$ is a propositional variable and $\Gamma \subseteq Ag$. We abbreviate $\langle\!\langle\Gamma\rangle\!\rangle(\mathrm{tt} U \varphi)$ by $\langle\!\langle\Gamma\rangle\!\rangle F\varphi$.

The formula $\langle\!\langle\Gamma\rangle\!\rangle X\varphi$ is read as "$\langle\!\langle\Gamma\rangle\!\rangle$ has a strategy to enforce $\varphi$ in the next state (irrespective of the actions of the agents in $Ag \setminus \Gamma$)"; $\langle\!\langle\Gamma\rangle\!\rangle G\varphi$ means "$\langle\!\langle\Gamma\rangle\!\rangle$ has a strategy to enforce $\varphi$ forever in the future"; and $\langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2$ represents "$\langle\!\langle\Gamma\rangle\!\rangle$ has a strategy to enforce that $\varphi_2$ holds at some point in the future and can ensure that $\varphi_1$ holds until then."

## 2.3 Two- and three-valued semantics for ATL

Let $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ be an interpreted system. A *strategy* for an agent $i \in \Gamma$ is a function $f_i : L_i^+ \to 2^{Act_i} \setminus \{\emptyset\}$ such that for each sequence of states $l_1 \ldots l_s$, all the actions in $f_i(l_1 \ldots l_s)$ are in $P_i(l_s)$.

A strategy is called *memoryless* if it depends only on the last state, i.e., if $f_i(l_1 \ldots l_s l) = f_i(l)$ for any natural number $s$ and any states $l_1, \ldots, l_s, l$.

Given a set of agents $\Gamma$ and an indexed set of strategies $F_\Gamma = \{f_i \mid i \in \Gamma\}$, we define $out(s, F_\Gamma)$ to be the set of infinite paths $s_0 s_1 \ldots$ such that $s_0 = s$ and for all $j$ there exists an action $a$ such that $T(s_j, a, s_{j+1})$ and for all $i \in \Gamma$, $a.i \in f(s_0.i \ldots s_j.i)$.

Interpreted systems with *imperfect recall* are those in which agents are required to use only memoryless strategies; while agents in interpreted systems with *perfect recall* can use arbitrary strategies.

For a path $p = s_0 s_1 \ldots$, by $p^i$ we denote $s_i$, the $i$-th element of $p$.

The following is the standard semantics for ATL.

**Definition 3** (Two-valued semantics)**.** *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$, its associated*

*model $M$ and a global state $s \in S$, the two-valued satisfaction relation $\models_{\overline{2}}$ is inductively defined as follows.*

$M, s \models_{\overline{2}} q$ *iff* $q \in \Pi(s)$

$M, s \models_{\overline{2}} \neg\varphi$ *iff* $M, s \not\models_{\overline{2}} \varphi$

$M, s \models_{\overline{2}} \varphi_1 \wedge \varphi_2$ *iff* $M, s \models_{\overline{2}} \varphi_1$ *and* $M, s \models_{\overline{2}} \varphi_2$

$M, s \models_{\overline{2}} \langle\langle\Gamma\rangle\rangle X\varphi$ *iff for some* $F_\Gamma$ *and all* $p \in out(s, F_\Gamma)$ *we have* $M, p^1 \models_{\overline{2}} \varphi$

$M, s \models_{\overline{2}} \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$ *iff for some* $F_\Gamma$ *and all* $p \in out(s, F_\Gamma)$, *there is a* $k \geq 0$ *s.t. we have* $M, p^k \models_{\overline{2}} \varphi_2$ *and for all* $0 \leq j < k, M, p^j \models_{\overline{2}} \varphi_1$

$M, s \models_{\overline{2}} \langle\langle\Gamma\rangle\rangle G\varphi$ *iff for some* $F_\Gamma$ *and all* $p \in out(s, F_\Gamma)$ *and for all* $i \geq 0$ *we have* $M, p^i \models_{\overline{2}} \varphi$

We now introduce a novel, three-valued semantics for ATL. Each ATL formula $\varphi$ is associated with three possible values (tt for true, ff for false, uu for undefined) when evaluated on an interpreted system $IS$ at a given state. We will later show that the three-valued semantics agrees with the two-valued one whenever the truth value is not undefined.

**Definition 4** (Three-valued semantics)**.** *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i\in Ag}, I, \Pi)$, its associated model $M$ and a global state $s \in S$, the three-valued satisfaction relation $\models_{\overline{3}}$ is inductively defined as follows.*

$$M, s \models_{\overline{3}} q = \begin{cases} \mathrm{tt} \textit{ iff } q \in \Pi(s), \\ \mathrm{ff} \textit{ iff } \neg q \in \Pi(s), \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \neg\varphi = \begin{cases} \mathrm{tt} \textit{ iff } M, s \models_{\overline{3}} \varphi = \mathrm{ff} \\ \mathrm{ff} \textit{ iff } M, s \models_{\overline{3}} \varphi = \mathrm{tt} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \varphi_1 \wedge \varphi_2 = \begin{cases} \mathrm{tt} \textit{ iff } M, s \models_{\overline{3}} \varphi_1 = \mathrm{tt} \textit{ and } M, s \models_{\overline{3}} \varphi_2 = \mathrm{tt} \\ \mathrm{ff} \textit{ iff } M, s \models_{\overline{3}} \varphi_1 = \mathrm{ff} \textit{ or } M, s \models_{\overline{3}} \varphi_2 = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \langle\langle\Gamma\rangle\rangle X\varphi = \begin{cases} \mathrm{tt} \textit{ iff for some strategy } F_\Gamma \textit{ and all } p \in out(s, F_\Gamma) \textit{ we have } M, p^1 \models_{\overline{3}} \varphi = \mathrm{tt} \\ \mathrm{ff} \textit{ iff for some strategy } F_{\overline{\Gamma}} \textit{ and all } p \in out(s, F_{\overline{\Gamma}}) \textit{ we have } M, p^1 \models_{\overline{3}} \varphi = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

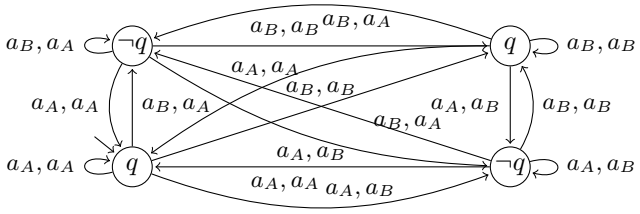
$$M, s \models_{\overline{3}} \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \begin{cases} \mathrm{tt} \textit{ iff for some strategy } F_\Gamma \textit{ and all } p \in out(s, F_\Gamma) \textit{ there is } k \in \mathbb{N} \textit{ s.t. } M, p^k \models_{\overline{3}} \varphi_2 = \mathrm{tt} \textit{ and for all } j < k, M, p^j \models_{\overline{3}} \varphi_1 = \mathrm{tt} \\ \mathrm{ff} \textit{ iff for some strategy } F_{\overline{\Gamma}} \textit{ and all } p \in out(s, F_{\overline{\Gamma}}), k \in \mathbb{N} \textit{ we have } M, p^k \models_{\overline{3}} \varphi_2 = \mathrm{ff} \textit{ or there is } j < k \textit{ s.t. } M, p^j \models_{\overline{3}} \varphi_1 = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \langle\langle\Gamma\rangle\rangle G\varphi = \begin{cases} \mathrm{tt} \textit{ iff for some strategy } F_\Gamma \textit{ and all } p \in out(s, F_\Gamma), i \in \mathbb{N} \textit{ we have } M, p^i \models_{\overline{3}} \varphi = \mathrm{tt} \\ \mathrm{ff} \textit{ iff for some strategy } F_{\overline{\Gamma}} \textit{ and all } p \in out(s, F_{\overline{\Gamma}}), \textit{ there is } i \in \mathbb{N} \textit{ s.t. } M, p^i \models_{\overline{3}} \varphi = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

We write $IS \models_{\overline{2}} \varphi$ iff for all the initial states $s$ we have that $M, s \models_{\overline{2}} \varphi$. Similarly, $IS \models_{\overline{3}} \varphi = \mathrm{tt}$ iff for all the initial states $s$, $M, s \models_{\overline{3}} \varphi = \mathrm{tt}$. We illustrate the differences between the two semantics on an example.

**Example 1.** Consider the interpreted system $IS$ defined on $Ag = \{1, 2\}$. Each agent $i \in \{1, 2\}$ is associated with two actions $Act_i = \{a_A, a_B\}$, two local states $L_i = \{l_A, l_B\}$, a protocol function $P_i$ s.t. $P_i(l) = Act_i$ for any $l \in L_i$, and the transition relation $t_i = \{(l_A, a_A, l_A), (l_A, a_B, l_B), (l_B, a_A, l_A), (l_B, a_B, l_B)\}$.

Assume that $I = \{(l_A, l_A)\}$ and $\Pi$ is the labelling function such that $\Pi((l_A, l_A)) = \Pi((l_B, l_B)) = \{q\}$ and $\Pi((l_B, l_A)) = \Pi((l_A, l_B)) = \{\neg q\}$. Then, in the model $M_{IS}$ of $IS$ we have $T = \{((l_a, l_b), (a_c, a_d), (l_c, l_d)) \mid a, b, c, d \in \{A, B\}\}$.




Consider a strategy $f_1$ for agent 1. If $a_A \in f_1((l_A))$, then $out((l_A, l_A), \{f_1\})$ contains a path $p$ such that $p^1 = (l_A, l_B)$. If $a_A \notin f_1((l_A))$, then $a_B \in f_1((l_A, l_A))$, and therefore $out((l_A, l_A), \{f_1\})$ contains a path $p'$ such that $p'^1 = (l_B, l_A)$. Therefore, $IS \not\models_{\overline{2}} \langle\langle 1\rangle\rangle Xq$ and it is not the case that $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{tt}$.

Note that $\overline{\{1\}} = \{2\}$. Consider a strategy $f_2$ for agent 2. If $a_B \in f_2((l_A))$, then there is $p \in out((l_A, l_A), \{f_2\})$ s.t. $p^1 = (l_B, l_A)$. Otherwise, $a_A \in f_2((l_A))$ and there is $p \in out((l_A, l_A), \{f_2\})$ s.t. $p^1 = (l_A, l_B)$. So it is not the case that $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{ff}$, and therefore $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{uu}$.

In other words, $IS \not\models_{\overline{2}} \langle\langle 1\rangle\rangle Xq$ means that agent 1 has no strategy to guarantee $q$ in the next state, while $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{ff}$ represents that agent 2 has a strategy to avoid $q$ (so it is not the case).

## 2.4 Relationship between $\models_{\overline{3}}$ and $\models_{\overline{2}}$

We show that the three-value semantics agrees with the two-valued semantics whenever the three-valued semantics returns tt or ff.

**Theorem 5.** *Consider an interpreted system $IS$, an ATL formula $\varphi$ and a global state $s$. We have the following:*

1. *If $M_{IS}, s \models_{\overline{3}} \varphi = \mathrm{tt}$, then $M_{IS}, s \models_{\overline{2}} \varphi$.*
2. *If $M_{IS}, s \models_{\overline{3}} \varphi = \mathrm{ff}$, then $M_{IS}, s \not\models_{\overline{2}} \varphi$.*

In the proof we use the following lemma.

**Lemma 6.** *For any disjoint groups of agents $\Gamma, \Gamma'$, any state $s$ and any strategies $F_\Gamma, F_{\Gamma'}$ we have $out(s, F_\Gamma \cup F_{\Gamma'}) \subseteq out(s, F_\Gamma)$.*

The proof follows from the fact that $out(s, F_\Gamma \cup F_{\Gamma'})$ represents the paths that result when agents from $\Gamma$ play according to $F_\Gamma$ and agents from $\Gamma'$ play according to $F_{\Gamma'}$, while $out(s, F_\Gamma)$ represents the paths that occur when agents from $\Gamma$ play according to $F_\Gamma$.

*Proof of Theorem 5.* The proof is by induction on $\varphi$.

If $M_{IS}, s \models_3 q = \mathrm{tt}$, then $q \in \Pi(s)$, and so $M_{IS}, s \models_2 q$.

If $M_{IS}, s \models_3 q = \mathrm{ff}$, then $\neg q \in \Pi(s)$. By the definition of the labelling function, $q \notin \Pi(s)$. Therefore, $M_{IS}, s \not\models_2 q$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle X\varphi' = \mathrm{tt}$. Then, there is $F_\Gamma$ such that for all $p \in out(s, F_\Gamma)$ we have $M_{IS}, p^1 \models_3 \varphi' = \mathrm{tt}$. By the inductive hypothesis, we have $M_{IS}, p^1 \models_2 \varphi'$, and therefore $M_{IS}, s \models_2 \langle\langle\Gamma\rangle\rangle X\varphi'$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle X\varphi' = \mathrm{ff}$. So there is a strategy $F_{\overline{\Gamma}}$ s.t. for all $p \in out(s, F_{\overline{\Gamma}})$ we have $M_{IS}, p^1 \models_3 \varphi' = \mathrm{ff}$. Consider any strategy $F_\Gamma$ of $\Gamma$ and a path $p \in out(s, F_\Gamma \cup F_{\overline{\Gamma}})$. By Lemma 6, $p \in out(s, F_{\overline{\Gamma}})$, so $M_{IS}, p^1 \models_3 \varphi = \mathrm{ff}$. By the inductive hypothesis, $M_{IS}, p^1 \not\models_2 \varphi'$. By Lemma 6, $p \in out(s, F_\Gamma)$. So we showed that for any strategy $F_\Gamma$ there is a path $p$ such that $M_{IS}, p^1 \not\models_2 \varphi'$; so $M_{IS}, s \not\models_2 \langle\langle\Gamma\rangle\rangle X\varphi'$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$. Then, there is $F_\Gamma$ such that for all $p \in out(s, F_\Gamma)$ there is $k$ s.t. $M_{IS}, p^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $j < k$ we have $M_{IS}, p^j \models_3 \varphi_1 = \mathrm{tt}$. By the inductive hypothesis we have that $M_{IS}, p^k \models_2 \varphi_2$ and for all $j < k$ we have $M_{IS}, p^j \models_2 \varphi_1$. Therefore $M_{IS}, s \models_2 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \mathrm{ff}$ and $F_{\overline{\Gamma}}$ is such that for all $p \in out(s, F_{\overline{\Gamma}})$ and all $k$ we have $M_{IS}, p^k \models_3 \varphi_2 = \mathrm{ff}$ or there exists $j < k$ s.t. $M_{IS}, p^j \models_3 \varphi_1 = \mathrm{ff}$. Consider any strategy $F_\Gamma$ of $\Gamma$, and let $p \in out(s, F_\Gamma \cup F_{\overline{\Gamma}})$. By Lemma 6, $p \in out(s, F_{\overline{\Gamma}})$ and $p \in out(s, F_\Gamma)$. By the inductive assumption, for all $k$, $M_{IS}, p^k \not\models_2 \varphi_2$ or there exists $j < k$ s.t. $M_{IS}, p^j \not\models_2 \varphi_1$. Since $F_\Gamma$ was an arbitrary strategy, we conclude that $M_{IS}, s \not\models_2 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$.

The remaining inductive cases can be shown in a similar manner. □

Notice that the above proof remains valid if we consider only memoryless strategies.

In the next Section we will explore how the novel, three-valued semantics introduced here can be used as part of an abstraction methodology in order to verify multi-agent systems against ATL specifications.

## 3 Abstraction

In this section we provide an abstraction methodology for constructing abstract interpreted systems. We introduce two techniques: *action abstraction* and *state abstraction*.

Recall that a function is *decomposable* if for any $x_1, \ldots, x_m$ we have that $f((x_1, \ldots, x_m)) = (f_1(x_1), \ldots, f_m(x_m))$, for some $f_i, i = 1, \ldots, m$. By $f.i$ we denote the function $f_i$.

Given an interpreted system $IS$ an *action abstraction function* is a surjective and decomposable function $\alpha : ACT \to ACT^\alpha$, for some set $ACT^\alpha$. The set $ACT$ is the set of joint actions whereas $ACT^\alpha$ constitutes the set of *abstract joint actions* for the corresponding abstract interpreted system. This enables us to define the action abstraction of an interpreted system as follows.

**Definition 7** (Action abstraction)**.** *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ and an action abstraction function $\alpha$, the* action abstraction *of $IS$ w.r.t. $\alpha$ is an interpreted system $IS^\alpha = (\{L_i^\alpha, Act_i^\alpha, P_i^\alpha, t_i^\alpha\}_{i \in Ag}, I^\alpha, \Pi^\alpha)$ such that $I^\alpha = I$, $\Pi^\alpha = \Pi$, and for each agent $i$:*

- $L_i^\alpha = L_i$,
- $Act_i^\alpha = \alpha.i(Act_i)$,
- $P_i^\alpha(l) = \alpha.i(P_i(l))$ *for all states* $l \in L_i$,
- *for all $l, l' \in l_i$ and $a^\alpha \in ACT^\alpha$, $t_i^\alpha(l, a^\alpha, l')$ iff for some $a \in ACT$ s.t. $\alpha.i(a) = a^\alpha$ we have $t_i(l, a, l')$.*

In other words the application of an action abstraction function results in the reduction of the number of actions in an interpreted system, thereby generating an abstract one.

We now explore a similar concept related to states.

Given an interpreted system $IS$, a *state abstraction function* is a surjective and decomposable function $\sigma : S \to S^\sigma$, such that for each agent $i$ and any two local states $l, l' \in L_i$, if $\sigma.i(l) = \sigma.i(l')$, then $P_i(l) = P_i(l')$, for some set of states $S^\sigma$. The set $S$ is the set of global states of the associated model $M_{IS}$ and $S^\sigma$ constitutes the set of *abstract global states* for the corresponding abstract interpreted system.

**Definition 8** (State abstraction)**.** *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ and a state abstraction function $\sigma$, the* state abstraction *of $IS$ w.r.t. $\sigma$ is the interpreted system $IS^\sigma = (\{L_i^\sigma, Act_i^\sigma, P_i^\sigma, t_i^\sigma\}_{i \in Ag}, I^\sigma, \Pi^\sigma)$, where $I^\sigma = \sigma(I)$, $\Pi^\sigma(s^\sigma) = \bigcap_{s \in \sigma^{-1}(\{s^\sigma\})} \Pi(s)$ and for each agent $i$*

- $L_i^\sigma = \sigma.i(L_i)$,
- $Act_i^\sigma = Act_i$,
- $P_i^\sigma(l^\sigma) = \bigcup_{l \in \sigma.i^{-1}(\{l^\sigma\})} P_i(l)$,
- $t_i^\sigma(l^\sigma, a, l'^\sigma)$ *iff for some $l, l'$ such that $\sigma.i(l) = l^\sigma$ and $\sigma.i(l') = l'^\sigma$ we have $t_i(l, a, l')$.*

**Example 2.** Assume that we have two states: $s$ labelled by $\{p, \neg q\}$ and $s'$ labelled by $\{p, q\}$ such that $\sigma(s) = \sigma(s')$. The resulting state will be labelled by $\Pi^\sigma(\sigma(s)) = \{p\}$.

Similarly to the case above the application of a state abstraction function results in the reduction of the number of states in an interpreted system, thereby generating an abstract one.

Given an interpreted system $IS$, an *abstraction function* is a pair $\delta = (\alpha, \sigma)$, where $\alpha$ is an action abstraction function of $IS$ and $\sigma$ is an state abstraction function of $IS^\alpha$.

**Definition 9** (Abstraction)**.** *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ and an abstraction function $\delta = (\alpha, \sigma)$, the* abstraction *of $IS$ w.r.t. $\delta$ is the interpreted system $IS^\delta$ that is the state abstraction of $IS^\alpha$ w.r.t. $\sigma$.*

### 3.1 Preservation Theorem

If $\sigma$ is a state abstraction function and $p$ is a path in an interpreted system $IS$, then by $\sigma(p)$ we denote the path $\sigma(p^0)\sigma(p^1)\ldots$, i.e., the corresponding path in the abstract interpreted system $IS^\sigma$.

The following lemma illustrates the relationship between paths in an interpreted system and its state abstraction.

**Lemma 10.** *Given an interpreted system $IS$, let $\sigma$ be a state abstraction function of $IS$, $s \in S$ be a state and $F^\sigma_\Gamma$ be a strategy for a group of agents $\Gamma \subseteq Ag$. Then there is a strategy $F_\Gamma$ for the agents in $\Gamma$ such that $\sigma(out(s, F_\Gamma)) \subseteq out(\sigma(s), F^\sigma_\Gamma)$.*

*Proof.* Consider $IS^\sigma$ and a strategy $F^\sigma_\Gamma = \{f^\sigma_i \mid i \in \Gamma\}$. Let $F_\Gamma = \{f_i \mid i \in \Gamma\}$ where $f_i(l^0_i \ldots l^k_i) = \sigma.i^{-1}(f^\sigma_i(\sigma.i(l^0_i)) \ldots \sigma.i(l^k_i)))$. It can be checked that $F_\Gamma$ is a strategy on $IS$.

Let $p \in out(s, F_\Gamma)$. So for each state $p^j$ there is a joint action $a$ such that $T(p^j, a, p^{j+1})$. By definition we have that $T^\sigma(\sigma(p^j), a, \sigma(p^{j+1}))$ on $IS^\sigma$, and therefore $\sigma(p) \in out(\sigma(s), F^\sigma_\Gamma)$. □

**Lemma 11.** *Given an interpreted system $IS$, let $\alpha$ be an action abstraction function of $IS$, $s \in S$ be a state and $F^\alpha_\Gamma$ be a strategy for a group of agents $\Gamma \subseteq Ag$. Then there is a strategy $F_\Gamma$ for $\Gamma$ such that $out(s, F_\Gamma) \subseteq out(s, F^\alpha_\Gamma)$.*

*Proof.* Let $\alpha^{-1} \circ F^\alpha_\Gamma$ denote the set $\{\alpha^{-1} \circ f^\alpha_i \mid i \in \Gamma\}$, where $\alpha^{-1}$ is the counter-image of the surjective function $\alpha$.

It can be checked from the definition of action abstraction that $out(s, \alpha^{-1} \circ F^\alpha_\Gamma) \subseteq out(s, F^\alpha_\Gamma)$. □

We prove the following theorem.

**Theorem 12.** *Let $\delta = (\alpha, \sigma)$ be an abstraction function for an interpreted system $IS$. Then, for any state $s \in S$*

1. *If $M^\delta_{IS}, \sigma(s) \models_3 \varphi = \mathrm{tt}$, then $M_{IS}, s \models_3 \varphi = \mathrm{tt}$.*
2. *If $M^\delta_{IS}, \sigma(s) \models_3 \varphi = \mathrm{ff}$, then $M_{IS}, s \models_3 \varphi = \mathrm{ff}$.*

*Proof.* The proof is by induction on $\varphi$.

Consider a state $s$ and the case $\varphi = q$. If $M^\delta_{IS}, \sigma(s) \models_3 q = \mathrm{tt}$, then $q \in \Pi^\delta(s)$. It follows by definition that $q \in \bigcap_{s' \in \sigma^{-1}(\sigma(s))} \Pi^\alpha(\sigma(s'))$, and so $q \in \Pi^\alpha(s)$. Since $\Pi^\alpha = \Pi$, $q \in \Pi(s)$, we conclude that $M_{IS}, s \models_3 q = \mathrm{tt}$.

If $M^\delta_{IS}, \sigma(s) \models_3 q = \mathrm{ff}$, then $\neg q \in \Pi^\delta(s)$. By similar considerations we can conclude that $\neg q \in \bigcap_{s' \in \sigma^{-1}(s)} \Pi^\alpha(s')$, so $\neg q \in \Pi^\alpha(s) = \Pi(s)$ and $M_{IS}, s \models_3 q = \mathrm{ff}$.

Assume that $\varphi = \neg\varphi'$. If $M^\delta_{IS}, \sigma(s) \models_3 \neg\varphi' = \mathrm{tt}$, then $M^\delta_{IS}, \sigma(s) \models_3 \varphi' = \mathrm{ff}$. By the inductive assumption $M_{IS}, s \models_3 \varphi' = \mathrm{ff}$, and therefore $M_{IS}, s \models_3 \neg\varphi' = \mathrm{tt}$. Similarly, if $M^\delta_{IS}, \sigma(s) \models_3 \neg\varphi' = \mathrm{ff}$, then $M_{IS}, s \models_3 \neg\varphi' = \mathrm{ff}$.

The case for $\varphi = \varphi_1 \wedge \varphi_2$ is straightforward through by the inductive hypothesis.

The other inductive cases can be shown by using Lemma 10 and 11. We only report the case of $M^\delta_{IS}, \sigma(s) \models_3 \langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$; the others can be proven similarly.

Assume that $M^\delta_{IS}, \sigma(s) \models_3 \langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$ and let $F^\delta_\Gamma = \{f^\delta_i \mid i \in \Gamma\}$ be a strategy such that for all $p^\delta \in out(\sigma(s), F^\delta_\Gamma)$, there is $k \geq 0$ s.t. $M^\delta_{IS}, p^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$ we have $M^\delta_{IS}, p^j \models_3 \varphi_1 = \mathrm{tt}$.

Let $F^\alpha_\Gamma$ be a strategy obtained by applying Lemma 10 to $F^\delta_\Gamma$, and $F_\Gamma$ be a result of applying Lemma 11 to $F^\alpha_\Gamma$.

Consider any $p \in out(s, F_\Gamma)$. By Lemmas 10 and 11, $\sigma(p) \in out(\sigma(s), F^\delta_\Gamma)$, and therefore for some $k \geq 0$ we have $M^\delta_{IS}, \sigma(p)^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$ we have $M^\delta_{IS}, \sigma(p)^j \models_3 \varphi_1 = \mathrm{tt}$.

By the inductive hypothesis we have that $M_{IS}, p^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$, $M_{IS}, p^j \models_3 \varphi_1 = \mathrm{tt}$. It follows that $M_{IS}, s \models_3 \langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$. □

Observe that the theorem above holds irrespective of any assumption on the strategies being memoryless or memoryfull.

**Remark.** There are an interpreted system $IS$, a state abstraction function $\sigma$, an action abstraction function $\alpha$, and an ATL property $\varphi$ s.t. $IS \models_2 \varphi$ but $IS^\alpha \not\models_2 \varphi$ and $IS^\sigma \not\models_2 \varphi$.

Indeed, consider the $IS$ from Example 1 but with a different labelling function: $\Pi((l_A, l_A)) = \Pi((l_A, l_B)) = \{q\}$ and $\Pi((l_B, l_A)) = \Pi((l_B, l_B)) = \{\neg q\}$. Let $\varphi = \langle\!\langle 1\rangle\!\rangle X q$. Clearly, $IS \models_2 \varphi$. Consider an action abstraction function $\alpha$ such that $\alpha((a_1, a_2)) = (\epsilon, \epsilon)$ for all $a_1, a_2 \in \{a_A, a_B\}$. We have $IS^\alpha \not\models_2 \varphi$ since in the initial state 1 can only play action $\epsilon$ which may lead to the state $(l_B, l_B)$ labelled with $\neg q$. Similarly, consider a state abstraction function $\sigma$ such that $\sigma((s_1, s_2)) = (\epsilon, \epsilon)$ for all $s_1, s_2 \in \{l_A, l_B\}$. The state abstraction $IS^\sigma$ of $IS$ w.r.t. $\sigma$ contains only one state $(\epsilon, \epsilon)$ s.t. $\Pi^\sigma((\epsilon, \epsilon)) = \emptyset$, and therefore $IS^\sigma \not\models_2 \varphi$.

## 4 Model Checking Using Abstraction

In this section we put forward a methodology based on the abstraction results of the previous Section to model check multi-agent systems against ATL specifications. Recall that the model checking problem is defined as follows.

**Definition 13** (Model checking problem)**.** *Given an interpreted system $IS$ and an ATL specification $\varphi$, the* model checking problem *involves establishing whether $IS \models_2 \varphi$.*

The following theorem was stated in (Alur, Henzinger, and Kupferman 2002) and, to the best of our knowledge, showed in (Dima and Tiplea 2011).

**Theorem 14.** *Model checking interpreted systems with perfect recall and imperfect information against ATL specifications is undecidable.*

However, the problem is decidable if memoryless (as opposed to perfect recall) strategies are considered (Schobbens 2004). Recall that $\Delta^P_2 = \mathrm{P}^{\mathrm{NP}}$ is the class of problems that can be solved in polynomial time by a deterministic Turing machine with an NP oracle.

**Theorem 15.** *Model checking interpreted systems with imperfect information and memoryless strategies against ATL specifications is decidable in $\Delta^P_2$.*

The above complexity is given in the size of the model associated with an interpreted system (observe that the complexity on the size of the interpreted system itself is likely to be harder (Lomuscio and Raimondi 2006a)). However, note

that models of interpreted systems may be exponential in size of the interpreted system that generates them. Since model checkers typically work on the generated models, this creates the well-known difficulty of the *state-space explosion*.

To mitigate this difficulty in the following we develop a technique that allows us to avoid the construction of the (concrete) model associated with an interpreted system and consider its abstraction instead. By the results of the previous section, in several cases this will prove sufficient.

More specifically we proceed as follows. Given an interpreted system and a formula to be verified, firstly we build an action abstraction, then a state abstraction from it (see Subsection 4.2) and attempt verification on the result (see Subsection 4.1). If the result is either $\mathrm{tt}$ or $\mathrm{ff}$ we know that (via Theorem 12) this is also the result of the check on the concrete model. Otherwise, we perform a refinement procedure (see Subsection 4.3) to obtain a further abstract model and attempt verification again. We repeat this procedure until we obtain a model in which the specification can be verified.

### 4.1 Three-valued model checking interpreted systems

We extend the classical model checking problem recalled above into the more general formulation below.

**Definition 16** (Three-valued model checking problem)**.** *Given an interpreted system $IS$, an ATL specification $\varphi$ and a truth value $b \in \{\mathrm{ff}, \mathrm{uu}, \mathrm{tt}\}$, the* three-valued model checking problem *involves establishing whether $IS \models_3 \varphi = b$.*

We can show that model checking interpreted systems against ATL specifications under a three-valued semantics and memoryless strategies is also in $\Delta_2^P$.

**Proposition 17.** *Three-valued model checking interpreted systems with imperfect information and memoryless strategies against ATL specifications is decidable in $\Delta_2^P$.*

*Proof.* For each subformula, starting from the propositions, label all the states with the value of the formula in this state, using the values of its subformulas. This is straightforward in all cases except the subformulas of the form $\langle\langle\Gamma\rangle\rangle$; in these cases call a non-deterministic oracle a polynomial number of times to obtain a strategy as required. Since $\Delta_2^P = P^{NP}$, we obtain the result. □

Notice that the above procedure establishes the value of each subformula in each state. We will use this property later to define the refinement procedure.

### 4.2 Initial abstraction

Let $A(\varphi)$ denote the set of agents in $\varphi$ and $\mathcal{V}(\varphi)$ be the set of propositional variables in $\varphi$. For an interpreted system $IS$ and a local state $l_i \in L_i$, the labelling of $l_i$ is $\Pi(l_i) = \bigcap_{s \in S, s.i = l_i} \Pi(s)$.

For a set of agents $\Gamma$, we define $\alpha^I_{IS}(\Gamma)$ to be the action abstraction function $\alpha$ for $IS$ such that for every agent $i \in \Gamma$ we define $\alpha.i(a_i) = a_i$, and for every $i \notin \Gamma$ we let $\alpha.i(a_i) = \epsilon$. So, in other words, $\alpha^I_{IS}(\Gamma)$ returns an interpreted system disregarding all actions not in $\Gamma$.

Given an action abstraction function $\alpha$ and a set of literals $V$, let $\sigma^I_{IS}(\alpha, V)$ denote the state abstraction function $\sigma$ for $IS^\alpha$ such that for each agent $i \in Ag$, $\sigma.i(l) = \sigma.i(l')$ iff $\Pi(l) \cap V = \Pi(l') \cap V$ and $P_i^\alpha(l) = P_i^\alpha(l')$. So $\sigma^I_{IS}(\alpha, V)$ defines an interpreted system in which states agreeing on literals in $V$ are collapsed onto a single abstract state as long as all agents have the same protocol in these states.

**Definition 18** (Initial abstraction function)**.** *Let $IS$ be an interpreted system and $\varphi$ be an ATL specification. The* initial abstraction function *for $IS$ is the pair $\delta^I_{IS}(\varphi) = (\alpha^I_{IS}(A(\varphi)), \sigma^I_{IS}(\alpha^I_{IS}(A(\varphi)), \overline{\mathcal{V}(\varphi)}))$. $IS^{\delta^I_{IS}(\varphi)}$ denotes the* initial abstraction *of the interpreted system $IS$ w.r.t. $\delta^I_{IS}(\varphi)$.*

In summary the initial abstraction function collapses all actions of agents not referred to in the specification; from this interpreted system all states with the same labelling and the same protocol are further collapsed.

We exemplify this definition in Section 5.

### 4.3 Refinement

As mentioned earlier, it may be that $IS^{\delta^I_{IS}(\varphi)} \models_3 \varphi = \mathrm{uu}$, i.e., the initial abstraction is not sufficient to determine the value of $\varphi$. In these cases we need to refine $\delta^I_{IS}(\varphi)$. Unlike the classic technique (Clarke et al. 2000), our refinement procedure focuses on the states for which the value of some subformula of $\varphi$ is $\mathrm{uu}$.

Let $\preceq$ be any linear order on formulas of ATL such that if $\psi$ is a subformula of $\psi'$, then $\psi \preceq \psi'$. Let $Sub(\varphi)$ stand for the set of all the subformulas of $\varphi$ (including $\varphi$ itself). An *evaluation* of $\varphi$ on an interpreted system $IS$ is a function $L : S \times Sub(\varphi) \to \{\mathrm{ff}, \mathrm{uu}, \mathrm{tt}\}$ such that for all $s \in S, \varphi' \in Sub(\varphi)$ we have $L(s, \varphi')$ is the result of the $M_{IS}, s \models_3 \varphi'$. Note that one can easily adjust the model checking algorithm presented in Proposition 17 to return the evaluation.

We now discuss the refinement procedure REFINE (see Algorithm 1) that takes as input an interpreted system $IS$, an abstraction function $\delta$, a specification $\varphi$, an evaluation $L$ of $\varphi$ on $IS^\sigma$ and returns an abstraction function $\delta'$.

Firstly, REFINE computes the set $\Phi$ of subformulas $\varphi'$ of $\varphi$ s.t. for some $s^\delta$ s.t. there are at least two states $s, s' \in S$ s.t. $\sigma(s) = \sigma(s') = s^\delta$ we have $L(s^\delta, \varphi) = \mathrm{uu}$. If $\Phi$ is not empty, we take the smallest element $\psi$ and split all the abstract states $s^\delta$ s.t. $L(s^\delta, \psi) = \mathrm{uu}$ by using the function SPLITS_STATES. If $\Phi$ is empty, we refine the action abstraction function.

The function SPLITS_STATES($IS$, $\sigma$, $S_{\mathrm{uu}}$) defines a state abstraction function $\sigma'$ that agrees with $\sigma$ on all the local states not in $S_{\mathrm{uu}}$. For the remaining states, there are two possibilities. If there are two states $s, s' \in S_{\mathrm{uu}}$ and an agent $i$ such that $\sigma.i(s.i) = \sigma.i(s'.i)$ and $\Pi(s.i) \neq \Pi(s'.i)$, then $\sigma'$ splits the abstracted states into states corresponding to different labelling; otherwise $\sigma'$ separates all the local states in $S_{\mathrm{uu}}$.

The refinement procedure above focuses mostly on the state abstraction function, revising the action abstraction function only if no further updates to the state abstraction function are useful. This is optimised for ATL specifications of the form $\langle\langle\Gamma\rangle\rangle\varphi'$, where $\varphi'$ does not contain a further $\langle\langle\rangle\rangle$ operator. Most concrete ATL specifications follow this pattern. Indeed,

**Algorithm 1** The refinement procedure.

**procedure** REFINE($IS$, $\delta = (\alpha, \sigma)$, $\varphi$, $L$)
  $MS^\delta \leftarrow \{s^\delta \in S^\delta \mid |\sigma^{-1}(\{s^\delta\})| > 1\}$
  $\Phi \leftarrow \{\varphi' \in Sub(\varphi) \mid \exists s^\delta \in MS^\delta . L(s^\delta, \varphi') = \mathrm{uu}\}$
  **if** $\Phi$ is not empty **then**
    $\alpha' \leftarrow \alpha$
    $S_{\mathrm{uu}} \leftarrow \{s \in S \mid L(\sigma(s), \min_{\preceq}(\Phi)) = \mathrm{uu}\}$
    $\sigma' \leftarrow$ SPLIT_STATES($IS$, $\sigma$, $S_{\mathrm{uu}}$)
  **else if** $\alpha = \alpha^I_{IS}(A(\varphi))$ **then**
    $\alpha' \leftarrow \alpha^I_{IS}(Ag \setminus A(\varphi))$
    $\sigma' \leftarrow \sigma^I_{IS}(\alpha', \overline{\mathcal{V}(\varphi)})$
  **else if** $\alpha = \alpha^I_{IS}(Ag \setminus A(\varphi))$ **then**
    $\alpha' \leftarrow id$
    $\sigma' \leftarrow \sigma^I_{IS}(\alpha', \overline{\mathcal{V}(\varphi)})$
  **else** $\sigma' \leftarrow id$, $\alpha' \leftarrow id$
  **return** $(\alpha', \sigma')$
**procedure** SPLIT_STATES($IS$, $\sigma$, $S_{\mathrm{uu}}$)
  $split\_all \leftarrow \forall s, s' \in S_{\mathrm{uu}} . \forall i \in Ag . \sigma.i(s.i) = \sigma.i(s'.i) \Rightarrow \Pi(s.i) = \Pi(s'.i)$
  **for** $i \in Ag, l_i \in L_i$ **do**
    **if** $\exists s \in S_{\mathrm{uu}}\ s.i = l_i$ **then**
      **if** $split\_all$ **then** $\sigma'.i(l_i) \leftarrow l_i$
      **else** $\sigma'.i(l_i) \leftarrow (\sigma.i(l_i), \Pi(l_i))$
    **else** $\sigma'.i(l_i) \leftarrow \sigma.i(l_i))$
  **return** $\sigma'$

when evaluating whether $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle\varphi' = \mathrm{tt}$ the labels of the actions for the agents not in $\Gamma$ are irrelevant. This is why the initial abstraction collapses them. Conversely, when evaluating $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle\varphi' = \mathrm{ff}$ the actions that matter are those of $Ag \setminus \Gamma$; this is reflected in the first refinement of the action abstraction function. If none of the above is effective, we use the identity.

### 4.4 Model Checking with Refinement

In view of the constructions above, we can now give the overall model checking algorithm for verifying an interpreted system $IS$ against an ATL specification $\varphi$ as the procedure VERIFY at Algorithm 2. The procedure calls the function EVALUATE which labels the structure, as discussed in Proposition 17, and returns the appropriate evaluation function.

**Algorithm 2** The verification procedure.

**procedure** VERIFY($IS$, $\varphi$)
  $\delta \leftarrow \delta^I_{IS}(\varphi)$
  **while** $\delta \neq (id, id)$ **do**
    $L \leftarrow$EVALUATE($IS^\delta$, $\varphi$)
    **if** $\exists s \in I . L(\sigma(s), \varphi) = \mathrm{ff}$ **then return** ff
    **else if** $\forall s \in I . L(\sigma(s), \varphi) = \mathrm{tt}$ **then return** tt
    **else** $\delta \leftarrow$REFINE($IS, \delta, L$)
  **return** uu

The algorithm proposed first constructs the initial abstraction $\delta^I_{IS}$. A first check is made for the truth value of $\varphi$ on $\delta^I_{IS}$. If one abstract state corresponding to a concrete initial state does not satisfy $\varphi$, by Theorem 12 we deduce that the specification does not hold on the concrete interpreted system. Similarly if $\varphi$ is satisfied in all abstract states corresponding to all concrete initial states, we deduce that the specification is satisfied on the original model. If neither of these cases holds, we refine the initial abstraction by following the REFINE procedure above and repeat the evaluation tests. Note that the procedure always terminates; in the worst case it will attempt to perform the verification on the original, concrete model.

While in the worst case the procedure is worse than a single initial check on the concrete model, it is possible that either the initial abstraction or one of its refinements will determine the value of the specifications. The reduced models may be much smaller than the original concrete model, which may be too large to be verified even by using symbolic techniques. Even if the concrete model can be handled by concrete techniques, the procedure above offers an attractive methodology for reducing the time and memory footprint of the verification step.

## 5 Verification of Simbridge

In this section we exemplify the methodology above on *Simbridge*, a simple two player trick-taking game. We encode the game as an interpreted system and verify a specification by using the abstraction technique presented above.

The game is played with a deck of 26 cards (13 red and 13 black). Cards are numbered from 1 to 13. The goal of each player is to win the most number of rounds, or *tricks*.

Each player is initially given 13 cards by the croupier. The game is played in 13 rounds. The first player is called the *lead player* of the first round; in all the other rounds the lead player is the player who won the previous trick. In each round the lead player plays any card from his hand; the other player has to respond by playing a card in the same colour. The player who played the card with the higher number wins the trick. If a player cannot play a card in the colour of the card played by the first player, he can respond by playing any card and he loses the trick. If a player has a card in the required colour but plays a card in a different colour, he loses for cheating.

### 5.1 Game encoding

We model the game as an interpreted system $IS$ comprising four agents: Croupier, Player 1, Player 2 and Scorekeeper.

**Croupier.** The game begins with Croupier distributing the cards to the players as follows. For each card, starting from 1 Red, Croupier assigns the card to a player by either using the action 1 or 2. If a player has already been given 13 cards, all remaining cards are given, one by one, to the other player. After distributing all the cards, Croupier plays the action $\epsilon$ for the rest of the game.

Croupier's $14^2$ local states are tuples where each local state $(k, l)$ represents the situation where Player 1 was given $k$ cards and Player 2 was given $l$ cards. The Croupier's protocol is defined as follows: in a state $(k, l)$ Croupier can perform action 1 (resp. 2) if $k < 13$ (resp. $l < 13$). If action 1 (resp. 2) is performed, Croupier moves to the state $(k + 1, l)$

| Structure | Actions | States |
|---|---|---|
| Original structure ($IS$) | 2187 | $10^{27}$ |
| Initial abstraction ($IS^{\sigma}$) | 3 | $10^{3}$ |
| Refinement ($IS^{R}$) | 3 | $10^{8}$ |

Table 1: Estimated sizes of the interpreted systems from Section 5.

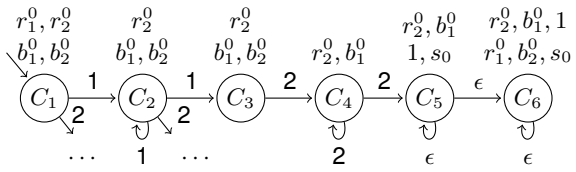



Figure 1: A fragment $M^{\delta}_{IS}$, the model associated with the initial abstraction.

(resp. $(k, l+1)$) regardless on the other agents' actions. Once Croupier is in the state $(13, 13)$ he loops in this state by performing the action $\epsilon$. Note that in this formalisation Croupier does not remember the cards' distribution.

**Players.** The players are symmetrical and follow the same formalisation. We represent the player's local states as $(lp, pc, H)$, where $H$ is the set of cards in the player's hand, $lp \in \{1, 2, C\}$ and $pc \in \{\mathrm{Red}_1, \ldots, \mathrm{Black}_{13}\} \cup \{\emptyset\}$. If $lp = C$ then the game is in the phase when the cards are being distributed and $pc$ represents the previously distributed card. If $lp \neq C$, then tricks are being played, $lp$ represents the current lead player and $pc$ encodes the first card played in the current round. It can be computed that there are approximately $2^{25}$ combinations; so each player has approximately $2^{31}$ possible local states.

The set of local actions contains $\epsilon$, which players use when it is not their turn, and the actions $\mathrm{Red}_1$, ..., $\mathrm{Black}_{13}$ that the players use to play a card when it is their turn. We assume that the protocol only insists on playing cards in the players' hands, i.e., players may not follow the colour played.

The transitions are defined as expected. Players begin with $lp = C$ and update their state while cards are being distributed. Then players start playing cards from their hands and update their state accordingly.

**Scorekeeper.** The scorekeeper's local states contain the states $Cheated_1$, $Cheated_2$ to represent situations where some of the players have cheated during the game, and tuples of the form $(lp, pc, SG)$ where $lp \in \{1, 2, C\}$ represents the lead player, $pc \in \{\mathrm{Red}_1, \ldots, \mathrm{Black}_{13}\} \cup \{\emptyset\}$ represents the first card played in the current round, if $lp \neq C$ and the last card distributed if $lp = C$. $SG$ is a tuple $SG = (S, R1, R2, B1, B2)$ representing the *state of the game* after each round where no player has cheated. $Ri$ represents the number of red cards in the hand of Player $i$; $Bi$ represents the number of black cards in the hand of Player $i$; $S$ represents the number of tricks won by Player 2. Some of the local states of Scorekeeper are not reachable during a game (e.g., states where $R1 + R2 > 13$). By considering this we can estimate that there are approximately $2.5 \times 10^6 \approx 2^{21}$ states possible local states for Scorekeeper.

Scorekeeper starts in the state $(C, \emptyset, (0, 0, 0, 0, 0))$ and then updates his states on the basis of the actions of Croupier and the players.

**Global states and variables.** We can define the set of global states for the system by considering tuples containing the local states for the agents defined above. The initial states for the system are those obtained by considering the initial local states of the agents defined above. We can ascertain there are at least $196 \cdot 2^{31} \cdot 2^{31} \cdot 2^{21} \approx 10^{27}$ global states defined as above. This constitutes a size that would normally not be verifiable even by an advanced, symbolic model checker.

We use the following propositional variables: 1, 2, $s^i$, $r_1^i$, $r_2^i$, $b_1^i$, $b_2^i$ for $i \in \{0, \ldots, 13\}$. Their truth value depends on the Scorekeeper's state. So if in a global state the Scorekeeper is in the local state $(lp, pc, (S, R1, R2, B1, B2))$, then the variables $lp, pc, s^S, r_1^{R1}, r_2^{R2}, b_1^{B1}, b_2^{B2}$ hold at that state. The variable $ch_i$ holds at any global state where Scorekeeper's local state includes $Cheated_i$.

## 5.2 A specification of interest

From the description above it is easy to see that Croupier can ensure that Player 1 wins the game. Croupier can for instance give all the red cards to Player 1; since Player 1 starts the game, Player 2 will never win a trick.

For convenience we use the following Boolean formulas.

$$\begin{aligned} GameOver &= ((1 \vee 2) \wedge r_1^0 \wedge r_2^0 \wedge b_1^0 \wedge b_2^0) \vee ch_1 \vee ch_2 \\ Wins_1 &= s^0 \vee \cdots \vee s^6 \vee ch_2 \end{aligned}$$

$GameOver$ represents that all rounds have been played, i.e., either the cards have been distributed and no player has any card left, or a player has cheated. $Wins_1$ encodes a situation where Player 1 has won the game, i.e., either Player 2 has cheated or he has won fewer than seven tricks.

Consider the following specifications:

$$\Phi_1 = \langle\!\langle Croupier \rangle\!\rangle G(\neg GameOver \vee Wins_1)$$
$$\Phi_2 = \langle\!\langle Croupier \rangle\!\rangle F(GameOver \wedge Wins_1)$$

The first formula states that Croupier has a strategy to ensure that forever in the future either Player 1 wins or the game does not end. The second states that Croupier can force the system to reach a final state where Player 1 has won the game. We now use the abstraction technique to verify the system against both specifications. As we show below, the initial abstraction is sufficient to verify $\Phi_1$, whereas to verify $\Phi_2$ we need to employ the refinement procedure.

## 5.3 Abstraction for $\Phi_1$

Consider the initial abstraction function $\delta = (\alpha, \sigma)$ for $\Phi_1$. Since $A(\Phi_1) = \{Croupier\}$, the initial action abstraction $\alpha$ collapses the actions of all the agents except $Croupier$. This entails that the set of abstract actions in $IS^{\delta}$ is $ACT^{\delta} = \{(\epsilon, \epsilon, \epsilon, \epsilon), (\epsilon, 1, \epsilon, \epsilon), (\epsilon, 2, \epsilon, \epsilon)\}$.

The initial abstraction function is built by considering the variables $\mathcal{V}(\Phi_1) = \{1, 2, r_1^0, r_2^0, b_1^0, b_2^0, ch_1, ch_2, s^0, \ldots, s^6\}$. We do not present the whole abstract model, but only the fraction that is needed to establish the value of $\Phi_1$.

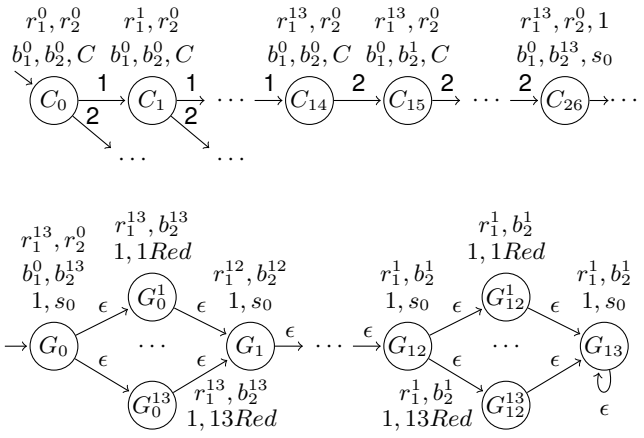



Figure 2: A fragment of the refined structure. For readability, some labels are omited. $G_0$ and $C_{26}$ denote the same state.

We first describe the part of the structure representing the card distribution. Figure 1 depicts five states of this structure. $C_1$ is the initial state of $M_{IS}^{\delta}$ corresponding to exactly one state in $M_{IS}$. The abstract state $C_2$ corresponds to the states in $M_{IS}$ representing situations in which Player 1 received between 1 and 12 red cards and Player 2 did not receive any. State $C_3$ corresponds to exactly the state in $M_{IS}$ in which Player 1 received 13 red cards and Player 2 did not receive any. Note that the only difference between these states is that Croupier's protocol in $C_3$ does not allow for 1 to be played. States $C_4$ and $C_5$ represent configurations in which Player 1 has all the red cards, and Player 2 received either between 1 and 12 ($C_4$), or 13 ($C_5$) black cards.

Consider $F = \{f_{Croupier}\}$ where $f_{Croupier}$ is the following strategy for Croupier: select action 1 in states $C_1$ and $C_2$, play action 2 in states $C_3$ and $C_4$, and $\epsilon$ in all the other states. Then, in $out(C_1, F)$ we have $C_1C_2^{\omega}$, $C_1C_2^+C_3C_4^{\omega}, C_1C_2^+C_3C_4^+C_5^{\omega}, C_1C_2^+C_3C_4^+C_5^+C_6^{\omega}$. It can be easily verified that all the paths satisfy $G\neg(GameOver \wedge Wins_2)$.

Therefore, we have that $IS^{\delta} \models_3 \Phi_1 = \mathrm{tt}$; so by Theorems 5 and 12 we deduce that $IS \models_2 \Phi_1$. The system $IS^{\delta}$ is much smaller than $IS$; its number of states can be bounded by the number of possible valuations of variables from $\mathcal{V}(\Phi_1)$ multiplied by the number of possible protocols; this results in $242 \cdot 4 \approx 10^3$ states. This is a size that can easily be verified on a model checker.

### 5.4 Abstraction for $\Phi_2$

Since $\mathcal{V}(\Phi_2) = \mathcal{V}(\Phi_1)$ the initial abstraction function $\delta$ is the same as the $\delta$ for $\Phi_1$ (see Figure 1). It can be checked that Croupier does not have a strategy to force $GameOver$. For example consider $F = \{f_{Croupier}\}$ where $f_{Croupier}$ is a strategy according to which action 1 is selected in $C_1$ and $C_2$. Then we have that $C_1C_2^{\omega} \in out(C_1, F)$, but $GameOver$ never holds on this path.

Note that the subformulas of $\Phi_2$ are $\langle\!\langle Croupier \rangle\!\rangle F(GameOver \wedge Wins_1)$, $GameOver \wedge Wins_1$, $GameOver$, $Wins_1$, and all the variables in $GameOver$ and $Wins_1$. Of these, the only subformula that is undefined in some of the states of the abstracted structure is $\Phi_2$. $\Phi_2$ is not undefined only where $GameOver$ is true. It follows that we cannot establish whether $\Phi_2$ holds on the system.

We now therefore execute the refinement procedure REFINE. This returns the same action abstraction function $\alpha' = \alpha$ and the state abstraction function $\sigma'$ such that $\sigma'(s) = \sigma(s)$ for $s$ satisfying $GameOver$ and $\sigma'(s) = (\sigma(s), \Pi(s))$ for all the remaining states. Let $IS^R$ be the abstract system obtained by refining the initial abstraction by means of $(\alpha', \sigma')$.

Like $IS^{\delta}$, the system $IS^R$ has only three global actions; its number of states can be bounded by $30079730 \cdot 4 \approx 10^8$. A fragment of this structure is presented in Figure 2. This is a size that can easily be verified by a modern model checker.

Consider now $F = \{f_{Croupier}\}$ where $f_{Croupier}$ is a strategy for Croupier that selects action 1 in states $C_0, \ldots, C_{13}$, action 2 in states $C_{14}, \ldots, C_{25}$ and $\epsilon$ at other points. Figure 2 shows that $out(C_0, F)$ consists of the paths of the form $C_0 \ldots C_{26}G_0G_0' \ldots G_{12}G_{12}'G_{13^{\omega}}$, where $G_i' = (G_i^1 + \cdots + G_i^{13})$. It can be checked that all these paths satisfy $F(GameOver \wedge Wins_1)$. It follows that $\Phi_2$ is satisfied in the refinement and therefore, by Theorem 12, we have that it also holds on the concrete model $IS$.

## 6 Conclusions

Model checking multiagent systems is now a well-established area of research with results ranging from theoretical investigations to sophisticated implementations. While two notable implementations, jMOCHA and MCMAS, support ATL as a specification language, neither provides abstraction functionalities. Yet, it is well known that abstraction is a fundamental technique to apply when verifying concrete systems.

Abstraction, including refinement, has been developed for epistemic specifications (Cohen et al. 2009) and, of course, temporal ones (Clarke, Grumberg, and Long 1994; Clarke et al. 2000). But in the context of ATL specifications, most of the work has so far focused on the theoretical aspects of the model checking problem, including realistic settings with incomplete information. Yet, to use ATL specifications in practice there is a compelling need for powerful abstraction techniques to reduce the state-spaces of concrete systems. In this paper we intended to make a contribution in this direction.

Our long-term research objective is to verify concrete systems via under-/over-approximations. We have laid the foundations of this by defining a novel three-valued semantics for ATL. We have shown this to be a conservative extension of the classical two-valued one. We have then provided a constructive definition to derive a first, coarse, yet potentially very effective abstraction, and a refinement algorithm that splits states potentially resolving uncertainty in the truth value of the specifications being checked. Applying the procedure to a card game of incomplete information. That showed the first abstraction reduced the state-space by a factor of over $10^{24}$ to a well-manageable state-space of approximately $10^3$. The first specification of interest was satisfied on the first abstraction; the second required us to execute the refinement

algorithm which resulted in a model of approximately $10^8$ states. The refined model enabled us to decide on the truth of the second specification; also note that state-spaces in the region of $10^8$ can be verified by a model checker for ATL such as MCMAS.

## References


Alur, R.; de Alfaro, L.; Grosu, R.; Henzinger, T. A.; Kang, M.; Kirsch, C. M.; Majumdar, R.; Mang, F.; and Wang, B.-Y. 2001. jMocha: A model checking tool that exploits design structure. In *Proceedings of the 23rd International Conference on Software Engineering (ICSE01)*, 835–836. IEEE.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Belnap, N., and Perloff, M. 1990. Seeing to it that: A canonical form for agentives. In *Knowledge Representation and Defeasible Reasoning*, volume 5 of *Studies in Cognitive Systems*, 167–190. Springer.

Bulling, N.; Dix, J.; and Chesñevar, C. I. 2008. Modelling coalitions: ATL + argumentation. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS08)*, 681–688. IFAAMAS.

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2007. Strategy logic. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR07)*, volume 4703 of *Lecture Notes in Computer Science*, 59–73. Springer.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV00)*, volume 1855 of *Lecture Notes in Computer Science*, 154–169. Springer.

Clarke, E. M.; Grumberg, O.; and Long, D. 1994. Model checking and abstractions. *ACM Transactions on Programming Languages and Systems* 16(5):1512–1542.

Cohen, M.; Dam, M.; Lomuscio, A.; and Russo, F. 2009. Abstraction in model checking multi-agent systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS09)*, 945–952. IFAAMAS Press.

Dima, C., and Tiplea, F. L. 2011. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *arXiv preprint arXiv:1102.4225*.

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

Jamroga, W., and Ågotnes, T. 2007. Modular interpreted systems. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS07)*, 131–138. IFAAMAS.

Köster, M., and Lohmann, P. 2012. Abstraction for model checking modular interpreted systems over ATL. In *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*. Springer. 95–113.

Lomuscio, A., and Raimondi, F. 2006a. The complexity of model checking concurrent programs against CTLK specifications. In *DALT06*, volume 4327 of *Lecture Notes in Computer Science*, 29–42.

Lomuscio, A., and Raimondi, F. 2006b. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS06)*, 161–168. ACM Press.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of the 21th International Conference on Computer Aided Verification (CAV09)*, volume 5643 of *Lecture Notes in Computer Science*, 682–688. Springer.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2012. What makes ATL* decidable? A decidable fragment of strategy logic. In *Proceedings of the 23th International Conference on Concurrency Theory (CONCUR12)*, volume 7454 of *Lecture Notes in Computer Science*. Springer. 193–208.

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