Verifying Multi-Agent Systems by Model Checking Three-valued Abstractions

Alessio Lomuscio
Imperial College London
a.lomuscio@imperial.ac.uk

Jakub Michaliszyn
Imperial College London
j.michaliszyn@imperial.ac.uk

ABSTRACT

We develop the theoretical foundations of a predicate abstraction methodology for the verification of multi-agent systems. We put forward a specification language based on epistemic logic and a weak variant of the logic ATL interpreted on a three-valued semantics. We show that the model checking problem for multi-agent systems in this setting is tractable by giving a provably correct procedure which admits a PTime bound. We give a constructive technique for generating abstract approximations of concrete multi-agent systems models and show that the truth values are preserved between abstract and concrete models. We evaluate the effectiveness of the methodology on a variant of the bit-transmission problem.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model checking

General Terms

Verification

Keywords

Epistemic logic; ATL; model-checking; abstraction

1. INTRODUCTION

With an increasing number of applications of societal importance adopting a multi-agent systems (MAS) approach, there is an increasing need to verify formally that a MAS design is correct before deployment. Over the past ten years a number of approaches have been put forward to verify MAS by means of model checking [13]. These include explicit approaches [22, 4], symbolic techniques [20, 44] and translations into SAT [29, 48]. Most methods that have been introduced support agent-based specifications giving prominence to the mental states of the agents, notably their knowledge [19]. Symbolic approaches are normally considered to be the most powerful in terms of performance as they can comfortably verify systems generating 10^{15} states and beyond [20, 29, 36]. While these are large models, complex MAS inevitably produce state-spaces that are considerably larger when not infinite. To overcome this difficulty abstraction techniques have been put forward [18, 15, 14] with some success.

A considerable limitation of these approaches, however, is that they address the verification of *models* of the system and cannot be extended to the validation of actual *MAS programs*. This hinders the automatic applicability of the methods as any MAS needs first to be modelled before verification can take place. Techniques targeting the verification of MAS code directly have been developed [47, 8, 5, 6]. However, their actual applicability has proven to be very limited as scalability issues have severely affected their effectiveness beyond small scenarios. They have also typically supported only temporal specifications; any agent informational attitude (e.g., beliefs, knowledge, etc.) has so far been treated as a mere predicate and therefore nested statements cannot be expressed.

Verifying code correctness is undecidable in general. One of the leading techniques used in formal verification is predicate abstraction [21]. Predicate abstraction involves automatically generating Boolean predicates in key sections of the code, typically by means of SMT calls [32], which can be used to produce abstract models that under-approximate and over-approximate the concrete models of the actual executions. To reason about specifications on over- and under-approximations temporal specifications are interpreted on three or four truth values. Abstract models can be refined (abstract states can be expanded into finer approximations) if the truth value of a specification is not assessed to true or false by the approximation considered.

In this paper we intend to lay the theoretical foundations to develop a predicate abstraction technique for MAS specified by the strategic-epistemic logic ATLK. This includes epistemic specifications including common knowledge [19] and strategy constructs to represent what coalitions of agents may bring about in the system [2]. As we target the effective verification of MAS, efficiency is a key desideratum in our set-up. For this reason we operate in a setting where ATL operators are interpreted on non-uniform models [37]. This enables us to obtain a PTime verification procedure albeit at a cost of a non-standard interpretation of the ATL operators. In scenarios where the reading of these operators is not suitable, we can simply use them to encode plain temporal-epistemic specifications by means of the logic CTLK, the combination of branching time logic CTL with epistemic logic [40], which is strictly subsumed by ATLK.

Scheme of the paper. In Section 2 we present and discuss the syntax and semantics for the novel three-valued

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

epistemic logic we develop and present an example. In Section 3 we define and give a procedure for the model checking problem in the three-valued setting we investigate. In Section 4 we introduce abstract models to approximate the concrete models and study preservation results in the semantics and between models. In Section 5 we introduce a constructive methodology for producing an initial abstraction and refine the model in case the verification is non-conclusive; we give an example to illustrate the technique in Section 6. We conclude in Section 7 where we also discuss related work.

2. THE THREE-VALUED LOGIC ATLK

In this section we put forward our rationale for the choice of the specification language for MAS that we adopt in this paper. Following these considerations we introduce a novel three-valued logic for knowledge and exemplify its use.

2.1 Desiderata for the Specification Language

Differently from reactive systems whose specifications typically involve temporal properties only, MAS are specified in terms of their informational states, notably the knowledge of the agents in the system [19]. The logics most commonly used [38, 40, 43] combine epistemic modalities with temporal operators. The assumptions on the underlying temporal model range from discrete [38] to continuous [35], from linear [38] to branching [43], from state interpretations to intervals [33]. Irrespective of the different expressive power of the various temporal logics used, the rationale in a MAS setting for including a temporal dimension beside epistemic operators is clear: temporal logic provides a way for reasoning about a changing world and, in combination with epistemic concepts, it can be used to specify the evolution of the agents' knowledge.

The past 10 years have also witnessed a growing attention to the need of specifying strategic properties arising from the interaction within a MAS. The logic ATL [2], originally put forward to reason about the outcome of games, has been widely adopted in the MAS domain and extended to reason about what agents can enforce in an exchange [23, 27, 11]. Since ATL strictly subsumes the branching time temporal logic CTL, it has gradually replaced it in several model checking approaches to the verification of MAS [2, 23, 45, 30, 10]. Recent techniques have featured specification languages even stronger than ATL, including Strategy Logic [39, 25, 24, 12]. The computational complexity of the model checking problem against ATL varies greatly; it ranges from linear complexity in the size of the model to full undecidability, depending on the assumptions made on the observability of the global state and the agents' memory.

While studies on the expressivity and complexity of powerful specification languages make an important contribution to the area and guide us the in the tradeoff between expressivity and efficiency, in this line of research we intend to address the need for the *practical* verification of MAS. Towards this end, we need to base our choice of a specification language not only on expressivity but also, crucially, on considerations of optimal efficiency. Indeed, due to the difficulty of the state explosion problem [13], experimental results show that even a linear complexity upper-bound against the model causes severe difficulties even for state-of-the-art BDD-based model checkers for MAS.

From a computational complexity standpoint, the easiest set up for ATL is complete information, where we to assume that the strategies of the agents depend on the whole state and not on their own private state only. In MAS this is an unrealistic assumption as the agents have access to their local state only. Unfortunately, incomplete information is problematic; for example the model checking problem for incomplete information and perfect recall, i.e., considering local states as local histories, is undecidable [17]. A notable set-up studied in the literature consists in considering incomplete information with local memoryless *uniform strategies* [28, 27]. Here it is assumed that the agents have access to their own local states only, but their non-deterministic strategies are selected consistently along a path. While this seems an appealing framework, its model checking problem is Δ_P^2 complete against an explicit description of the model [26]. Other sophisticated proposals have been made, including richer operators for memory and forgetting [9], but none of these have a PTime bound.

Given the reasons above we here choose to adopt a framework where the agents have incomplete information and do not admit perfect recall, the strategies are local and not uniform. This set up has already been adopted previously in a two valued setting and referred to as "non-uniform" strategies [45] . As we clarify below, the reading of the modalities in this context is different from the usual one in ATL and does not capture the concept of "strict enforcement" typical of stronger versions of ATL. Our choice to adopt this setting is due to the fact that it enjoys good expressivity while its model checking problem is linear in size of both the model and formula. We regard it as the most expressive combination between epistemic and strategy operators which still has a PTime model checking problem. If the reading of the ATL modalities is not suitable to a particular scenario, we can use CTL which is strictly subsumed by the fragment we consider; indeed our technique can more simply be reformulated for the weaker logic CTLK.

Our intention is to lay the foundation for a predicate abstraction methodologies where abstractions and their refinements are computed during verification. To do this we adopt a three-valued semantics where formulas may be true, false, or undefined at a given state.

2.2 Syntax and Semantics

We put forward a three-valued version of the logic ATLK, ATLK^{3v} for short, combining ATL modalities with epistemic modalities in a three-valued setting. The agents are assumed to have incomplete information and their strategies are based on their local states. We do not assume perfect recall; these systems are often called "memoryless". While as described in Section 7 our set-up is different, we follow [34] for some of the basic definitions.

Let $Ag = \{1, \ldots, m\}$ be a set of agents and $\mathcal{V}$ be a set of propositional variables. We use the letter Γ to denote subsets of agents, e.g., $\Gamma \subseteq Ag$; by $\overline{\Gamma}$ we denote the complementation of Γ, i.e., $Ag \setminus \Gamma$. We use $\overline{\mathcal{V}}$ to denote the set of all the literals containing propositional variables from $\mathcal{V}$, i.e., $\overline{\mathcal{V}} = \{q, \neg q \mid q \in \mathcal{V}\}$. We begin by introducing the models we will be using in the rest of the paper.

Definition 1 (Interpreted systems). *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*;*
- $\Pi : L_1 \times \cdots \times L_m \to 2^{\overline{\mathcal{V}}}$ *is a* labelling function *such that for any variable* q *and a state* $s = (l_1, \ldots, l_m)$ *we either have* $q \notin \Pi(s)$ *or* $\neg q \notin \Pi(s)$*.*

Observe that our definition of interpreted systems extends the standard one [19] by admitting that a propositional variable nor its negation may hold at a state. This will form the basis for giving a three-valued interpretation on interpreted systems.

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.

For convenience we define models which are defined on the set of global states reachable from I via T.

Definition 2 (Models). *Given an interpreted system* $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$*, its* associated model *is a tuple* $M_{IS} = (S,\ T,\ \{\sim_i\}_{i \in Ag},\ I,\ \Pi)$ *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 for all* $i \in Ag$ *we have* $t_i(l_i, a, l'_i)$ *and* $a.i \in P_i(l_i)$*,*
- $\sim_i \subseteq S^2$ *is such that* $s \sim_i s'$ *iff* $s.i = s'.i$*, for all* $i \in Ag$*.*

We omit IS in the subscript if it is clear from the context, i.e., we write M for M_{IS}. The intended meaning of $s \sim_i s'$ is that the global states s, s' are epistemically indistinguishable for the agent i [19]. We extend the notion to groups by defining, for each group Γ, the relation $\sim_\Gamma = (\bigcup_{i \in \Gamma} \sim_i)^+$, where $^+$ denotes the transitive closure operator. For convenience, we write $a \equiv_\Gamma a'$ iff for all $i \in \Gamma$ we have $a.i = a'.i$.

We only consider models such 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 $T(s, a, s')$.

We now introduce the common syntax for the logics ATLK and ATLK^{3v}.

Definition 3 (ATLK specifications). *The set of formulas for the logics ATLK and ATLK* ^{3v} *is defined from* $\mathcal{V}$ *by the following BNF expression:*
$\varphi ::= q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle\Gamma\rangle\rangle X\varphi \mid \langle\langle\Gamma\rangle\rangle(\varphi U \varphi) \mid \langle\langle\Gamma\rangle\rangle G\varphi \mid K_i\varphi \mid C_{\Gamma'}\varphi$
where $i \in Ag$*,* $\Gamma, \Gamma' \subseteq Ag$*,* $\Gamma' \neq \emptyset$ *and* $q \in \mathcal{V}$*.*

We use the standard abbreviations to define $\langle\langle\Gamma\rangle\rangle F\varphi, E_\Gamma\varphi$ and the Boolean connectives.

The formulas $K_i\varphi, C_{\Gamma'}\varphi$ are read as "the agent i knows that φ" and "in the group Γ' it is commonly known that φ", respectively. Their reading is standard [19]. The formula $\langle\langle\Gamma\rangle\rangle G\varphi$ stands for "the agents in Γ *may be able* to ensure that φ holds forever"; the meaning of the "until" modality U is analogous. This reading is the one given in [45] for ATL on non-uniform models and differs from some of the literature on ATL which assumes a definition of strategies on global histories. While our notion of the ATL modalities does not generally represent enforcement, this is retained for the next state operator "X" which can be read as "the agents in Γ can ensure that φ holds at the next state irrespective of the actions of the agents in $Ag \setminus \Gamma$". For computational complexity considerations as well as our previously discussed need to ground strategies on local states, our requirements are considerably weaker as we explain below.

Assume an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$. A (*local memoryless*) *strategy* for an agent $i \in \Gamma$, or simply a strategy, is a function $f_i : s.i \to 2^{Act_i} \setminus \{\emptyset\}$ such that for each local state $s.i \in L_i$ we have $f_i(s) \subseteq P_i(s.i)$. Notice that we do not assume perfect recall, i.e., all the strategies depend on the current local state only.

Given a path $p = s_0 s_1 \ldots$, by p^i we denote s_i, the i-th element of p. Assume a set of agents Γ and an indexed set of strategies $F_\Gamma = \{f_i \mid i \in \Gamma\}$. We say that a set of (infinite) paths X is F_Γ*-compatible* if it is a minimal non-empty set of paths such that for each path $p \in X$ and each position $j \geq 0$ there is a joint action a such that $T(p^j, a, p^{j+1})$, for all $i \in \Gamma$, $a.i \in f_i(s_j.i)$, and for all joint actions $a' \equiv_\Gamma a$ and all states s' such that $T(p^j, a', s')$, there exists a path $p' \in X$ starting with $p^0 \ldots p^j s'$. Let $out(s, F_\Gamma)$ be the family of all F_Γ-compatible sets of paths starting with s.

We do not assume uniformity [45, 41] in the definition of paths. In other words we do not require that the action choice of any agent $i \in \Gamma$ in a path is always the same, as long as it conforms to the strategy function f_i, hence to agent i's protocol P_i.

Two-valued semantics.

Having fixed the above we can now provide the definition for two-valued satisfaction.

Definition 4 (Two-valued Satisfaction). *Assume an interpreted system* $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$*, its associated model* $M = (S,\ T,\ \{\sim_i\}_{i \in Ag},\ I,\ \Pi)$ *and a global state* $s \in S$*. We inductively define the two-valued satisfaction relation* $\models^2$ *as follows.*

$M, s \models^2 q$	*iff* $q \in \Pi(s)$
$M, s \models^2 \neg\varphi$	*iff* $M, s \not\models^2 \varphi$
$M, s \models^2 \varphi_1 \wedge \varphi_2$	*iff* $M, s \models^2 \varphi_1$ *and* $M, s \models^2 \varphi_2$
$M, s \models^2 \langle\langle\Gamma\rangle\rangle X\varphi$	*iff for some strategy* F_Γ*, some* $X \in out(s, F_\Gamma)$ *and all* $p \in X$ *we have* $M, p^1 \models^2 \varphi$
$M, s \models^2 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$	*iff for some strategy* F_Γ*, some* $X \in out(s, F_\Gamma)$ *and all* $p \in X$*, there is a* $k \geq 0$ *s.t. we have* $M, p^k \models^2 \varphi_2$ *and for all* $0 \leq j < k, M, p^j \models^2 \varphi_1$
$M, s \models^2 \langle\langle\Gamma\rangle\rangle G\varphi$	*iff for some strategy* F_Γ*, some* $X \in out(s, F_\Gamma)$ *and all* $p \in X$*,* $i \geq 0$ *we have* $M, p^i \models^2 \varphi$
$M, s \models^2 K_i\varphi$	*iff for all* $s' \sim_i s$ *we have* $M, s' \models^2 \varphi$
$M, s \models^2 C_\Gamma\varphi$	*iff for all* $s' \sim_\Gamma s$ *we have* $M, s' \models^2 \varphi$

An interpreted system IS satisfies a property φ, written as $IS \models^2 \varphi$, iff for all the initial states s we have $M, s \models^2 \varphi$.

Three-valued semantics.

We now introduce the logic ATLK^{3v}. In ATLK^{3v}, a formula at a given state of an interpreted system can be true (tt), false (ff) or undefined (uu).

Assume an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i\in Ag}, I, \Pi)$, its associated model $M = (S, T, \{\sim_i\}_{i\in Ag}, I, \Pi)$ and a global state $s \in S$. The inductive definition of the three-valued satisfaction $\models^3$ is given below.

We assume the Kleene semantics for the standard boolean connectivities.

$$M, s \models^3 q \quad = \begin{cases} \text{tt iff } q \in \Pi(s), \\ \text{ff iff } \neg q \in \Pi(s), \\ \text{uu otherwise} \end{cases}$$

$$M, s \models^3 \neg\varphi \quad = \begin{cases} \text{tt iff } M, s \models^3 \varphi = \text{ff} \\ \text{ff iff } M, s \models^3 \varphi = \text{tt} \\ \text{uu otherwise} \end{cases}$$

$$M, s \models^3 \varphi_1 \wedge \varphi_2 = \begin{cases} \text{tt iff } M, s \models^3 \varphi_i = \text{tt for all } i \in \{1, 2\} \\ \text{ff iff } M, s \models^3 \varphi_i = \text{ff for any } i \in \{1, 2\} \\ \text{uu otherwise} \end{cases}$$

For the ATL modalities, we adjust the semantics of [34].

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

$$M, s \models^3 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \begin{cases} \text{tt iff for some strategy } F_\Gamma, \text{ some } X \in out(s, F_\Gamma) \text{ and all } p \in X \text{ there is } k \geq 0 \text{ s.t. } M, p^k \models^3 \varphi_2 = \text{tt and for all } j < k,\ M, p^j \models^3 \varphi_1 = \text{tt} \\ \text{ff iff for some strategy } F_{\overline{\Gamma}}, \text{ all } X \in out(s, F_{\overline{\Gamma}}) \text{ and all } p \in X,\ k \geq 0 \text{ we have } M, p^k \models^3 \varphi_2 = \text{ff or there is } j < k \text{ s.t. } M, p^j \models^3 \varphi_1 = \text{ff} \\ \text{uu otherwise} \end{cases}$$

$$M, s \models^3 \langle\langle\Gamma\rangle\rangle G\varphi \quad = \begin{cases} \text{tt iff for some strategy } F_\Gamma, \text{ some } X \in out(s, F_\Gamma) \text{ and all } p \in X,\ i \geq 0 \text{ we have } M, p^i \models^3 \varphi = \text{tt} \\ \text{ff iff for some strategy } F_{\overline{\Gamma}}, \text{ all } X \in out(s, F_{\overline{\Gamma}}) \text{ and all } p \in X \text{ there is } i \geq 0 \text{ s.t. } M, p^i \models^3 \varphi = \text{ff} \\ \text{uu otherwise} \end{cases}$$

Notice that in the cases above the requirements for tt are very similar to the conditions in the two-valued semantics. An ATL formula is ff if the complement of the agents in the modality may be able to ensure the formula is ff.

For the knowledge modalities, we propose the following.

$$M, s \models^3 K_i\varphi = \begin{cases} \text{tt iff } M, s' \models^3 \varphi = \text{tt for all } s' \sim_i s \\ \text{ff iff } M, s \models^3 \varphi = \text{ff} \\ \text{uu otherwise} \end{cases}$$

$$M, s \models^3 C_\Gamma\varphi = \begin{cases} \text{tt iff } M, s' \models^3 \varphi = \text{tt for all } s' \sim_\Gamma s \\ \text{ff iff } M, s \models^3 \varphi = \text{ff} \\ \text{uu otherwise} \end{cases}$$

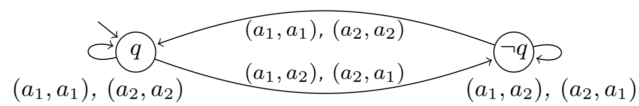
As above, note that the condition for tt is similar to the one for the two-valued case. However, an epistemic formula is ff if the nested formula's valuation at the present state is ff. A weaker alternative, compatible with the two-valued semantics, is to assign $M, s \models^3 K_i\varphi = \text{ff}$ iff there is a state $s' \sim_i s$ such that $M, s' \models^3 \varphi = \text{ff}$. Such definition, however, would not enable us to ensure the transfer of the value ff from the abstract to concrete model (Theorem 11 below), an essential part of our abstraction technique.

We assign $IS \models^3 \varphi = \text{tt}$ iff for all $s \in I$ we have $M, s \models^3 \varphi = \text{tt}$, $IS \models^3 \varphi = \text{ff}$ iff there is $s \in I$ such that $M, s \models^3 \varphi = \text{ff}$, and $IS \models^3 \varphi = \text{uu}$ otherwise.

We now exemplify the above definitions.

Example 1. *Let $Ag = \{1, 2\}$. Consider the interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i\in Ag}, I, \Pi)$ such that*

- $L_1 = \{l_=, l_\neq\}$, $L_2 = \{\epsilon\}$,
- $P_1(l_=) = P_1(l_\neq) = P_2(\epsilon) = Act_1 = Act_2 = \{a_1, a_2\}$,
- $t_1 = \{(l, (a_i, a_j), l_=) \mid l \in L_1 \wedge i = j\} \cup \{(l, (a_i, a_j), l_\neq) \mid l \in L_1 \wedge i \neq j\}$, $t_2 = \{(\epsilon, (a_i, a_j), \epsilon) \mid i, j \in \{1, 2\}\}$,
- $I = \{(l_=, \epsilon)\}$, $\Pi((l_=, \epsilon)) = \{q\}$ *and* $\Pi((l_\neq, \epsilon)) = \{\neg q\}$.



Consider the formula $\langle\langle 1\rangle\rangle Xq$ and the initial state $(l_=, \epsilon)$ indicated in the figure by the incoming arrow. It can be checked that agent 1 cannot enforce q in the next state as agent 2 can use a different action from that of agent 1. Therefore we have $IS \not\models^2 \langle\langle 1\rangle\rangle Xq$ and not $IS \models^3 \langle\langle 1\rangle\rangle Xq = \text{tt}$. Similarly, regardless on the choice of the agent 2, if the agent 1 selects the same action at the initial state, then the next state is $(l_=, \epsilon)$. So it is not the case that $IS \models^3 \langle\langle 1\rangle\rangle Xq = \text{ff}$, and therefore $IS \models^3 \langle\langle 1\rangle\rangle Xq = \text{uu}$.

We have $IS \not\models^2 \langle\langle 1\rangle\rangle Xq$; so it is not the case that agent 1 may ensure q at the next state. We also have $IS \models^3 \langle\langle 1\rangle\rangle Xq = \text{ff}$; so it is not the case that agent 2 has a strategy to avoid q.

Consider the formula K_2q at the initial state. Since both states of the model are indistinguishable for agent 2, we have $IS \not\models^2 K_2q$ and not $IS \models^3 K_2q = \text{tt}$. However, since the initial state $(l_=, \epsilon)$ satisfies q, it is not the case that $IS \models^3 K_2q = \text{ff}$; therefore we have $IS \models^3 K_2q = \text{uu}$.

Note that in the setting we consider the strategies in both two-valued semantics and three-valued semantics can be replaced by agents' protocols.

3. CHECKING MAS AGAINST ATLK^{3V}

We give a definition of the model checking problem against ATLK^{3v} specifications and present a polynomial time algorithm to solve it. We phrase this as a decision procedure.

Definition 5 (Model checking problem). *Given an interpreted system IS, an ATLK specification φ and $b \in \{\text{tt}, \text{ff}, \text{uu}\}$, the* three-valued model checking problem *involves establishing whether $IS \models^3 \varphi = b$.*

We put forward Algorithm 1 to compute the truth value of a formula φ given a state s in an interpreted system IS.

Algorithm 1 The model checking procedure for verifying an IS against ATLK^{3v} specifications.

procedure $\text{VERIFY}(IS, \varphi)$
 $(S_{tt}, S_{ff}) \leftarrow \text{LABEL}(IS, \varphi)$
 if $\forall s \in I.s \in S_{tt}$ **then return** tt
 if $\exists s \in I.s \in S_{ff}$ **then return** ff
 return uu
procedure $\text{LABEL}(IS, \varphi)$
 if $\varphi = q$ **then**
  $S_{tt} \leftarrow \{s \mid q \in \Pi(s)\}$, $S_{ff} \leftarrow \{s \mid \neg q \in \Pi(s)\}$
 else if $\varphi = \varphi_1 \wedge \varphi_2$ **then**
  $(S^1_{tt}, S^1_{ff}) \leftarrow \text{LABEL}(IS, \varphi_1)$
  $(S^2_{tt}, S^2_{ff}) \leftarrow \text{LABEL}(IS, \varphi_2)$
  $S_{tt} \leftarrow S^1_{tt} \cap S^2_{tt}$, $S_{ff} \leftarrow S^1_{ff} \cup S^2_{ff}$
 else if $\varphi = \langle\langle\Gamma\rangle\rangle X\varphi'$ **then**
  $(S'_{tt}, S'_{ff}) \leftarrow \text{LABEL}(IS, \varphi')$
  $S_{tt} \leftarrow \text{PREIMAGE}(IS, \Gamma, S'_{tt})$
  $S_{ff} \leftarrow \text{PREIMAGE}(IS, \overline{\Gamma}, S'_{ff})$
 else if $\varphi = \langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$ **then**
  $(S^1_{tt}, S^1_{ff}) \leftarrow \text{LABEL}(IS, \varphi_1)$
  $(S_{tt}, S_{ff}) \leftarrow \text{LABEL}(IS, \varphi_2)$
  repeat
   $S'_{tt} \leftarrow S_{tt}$
   $S_{tt} \leftarrow \text{PREIMAGE}(IS, \Gamma, S_{tt}) \cap S^1_{tt}$
   $S_{tt} \leftarrow S_{tt} \cup S'_{tt}$
  until $S_{tt} = S'_{tt}$
  repeat
   $S'_{ff} \leftarrow S_{ff}$
   $S_{ff} \leftarrow \text{PREIMAGE}(IS, \Gamma, S_{ff}) \cup S^1_{ff}$
   $S_{ff} \leftarrow S_{ff} \cap S'_{ff}$
  until $S_{ff} = S'_{ff}$
 else if $\varphi = \langle\langle\Gamma\rangle\rangle G\varphi'$ **then**
  $(S_{tt}, S_{ff}) \leftarrow \text{LABEL}(IS, \varphi')$
  repeat
   $S'_{tt} \leftarrow S_{tt}$
   $S_{tt} \leftarrow \text{PREIMAGE}(IS, \Gamma, S_{tt}) \cap S_{tt}$
  until $S_{tt} = S'_{tt}$
  repeat
   $S'_{ff} \leftarrow S_{ff}$
   $S_{ff} \leftarrow \text{PREIMAGE}(IS, \Gamma, S_{ff}) \cup S'_{ff}$
  until $S_{ff} = S'_{ff}$
 else if $\varphi = K_i\varphi'$ **then**
  $(S'_{tt}, S_{ff}) \leftarrow \text{LABEL}(IS, \varphi')$
  $S_{tt} \leftarrow S \setminus \{s \mid \exists s'.s' \sim_i s \wedge s' \notin S'_{tt}\}$
 else if $\varphi = C_\Gamma\varphi'$ **then**
  $(S'_{tt}, S_{ff}) \leftarrow \text{LABEL}(IS, \varphi')$
  $S_{tt} \leftarrow S \setminus \{s \mid \exists s'.s' \sim_\Gamma s \wedge s' \notin S'_{tt}\}$
 return (S_{tt}, S_{ff})

Propositional variables are labelled immediately with true and false. From those and by means of pre-image computations states are labelled with true and false for the relevant ATL and epistemic modalities, whenever possible.

For brevity, Algorithm 1 assumes that we have the relation $\sim_\Gamma$ to compute C_Γ; this can be obtained by means of fix-point computation. The procedure $\text{PREIMAGE}(IS, \Gamma, S')$ used in Algorithm 1 returns a set S_X of states from which Γ can *enforce* S' in one step. Formally, $s \in S_X$ iff there is a joint action $a \in ACT$ and a state s' such that $T(s, a, s')$ and for all $a' \equiv_\Gamma a$ and for all s'' if $T(s, a', s'')$, then $s'' \in S'$.

The algorithm can be shown to be correct.

THEOREM 6. *Consider an interpreted system IS, a state s, an ATLK specification φ and $(S_{tt}, S_{ff}) = \text{VERIFY}(IS, \varphi)$. We have that $IS, s \models^3 \varphi = \text{tt}$ iff $s \in S_{tt}$ and $IS, s \models^3 \varphi = \text{ff}$ iff $s \in S_{ff}$.*

PROOF. The proof is by induction on φ. The basic case and the Boolean cases are immediate. We prove the case for $\varphi = \langle\langle\Gamma\rangle\rangle X\varphi$; the others can be shown similarly.

Assume $\varphi = \langle\langle\Gamma\rangle\rangle X\varphi$ and let $(S_{tt}, S_{ff}) = \text{VERIFY}(IS, \varphi)$. For each state $s \in S_{tt}$ there is $a \in ACT$ and a state $s' \in S$ s.t. $T(s, a, s')$ and for all $a' \equiv_\Gamma a$ and all s'' s. t. $T(s, a', s'')$, we have $IS, s'' \models^3 \varphi = \text{tt}$ by the inductive hypothesis. Consider any strategy $F_\Gamma = \{f_i \mid i \in \Gamma\}$ such that $a.i \in f_i(s)$ for all $i \in \Gamma$. There is a set $X \in out(s, F_\Gamma)$ s.t. $p \in X$ and all each path of X starts with ss'' for some s'' satisfying φ. Therefore $IS, s \models^3 \varphi = \text{tt}$.

If $IS, s \models^3 \varphi = \text{tt}$, then there is a strategy F_Γ and a F_Γ-compatible set $X \in out(s, F_\Gamma)$ such that for all $p \in X$ we have $IS, p^1 \models^3 \varphi = \text{tt}$. As X is F_Γ-compatible, there is a joint action a such that $T(p, a, p^1)$ for some path $p \in X$, and for all joint actions $a' \equiv_\Gamma a$ and states s'' such that $T(s, a', s'')$ we have $IS, s'' \models^3 \varphi = \text{tt}$. Therefore, by the inductive hypothesis, we have $s \in S_{tt}$. □

From the above it follows, correctly, that $IS, s \models^3 \varphi = \text{uu}$ if neither $s \in S_{tt}$ nor $s \in S_{ff}$. Note that all operations in Algorithm 1 are either basic set operations or existential preimage computations. It is known that these can be efficiently implemented on BDD-based symbolic model checking [13].

Algorithm 1 can also be used to provide an upper bound for the verification problem.

THEOREM 7. *Three-valued model checking interpreted systems with imperfect information and memoryless local strategies against ATLK specifications is decidable in* PTIME.

PROOF SKETCH. For each subformula φ' of φ, the function $\text{LABEL}(IS, \varphi')$ is run only once in the algorithm. Each of the fixed-point computations in the algorithm terminates after at most $|S|$ steps. The function $\text{PREIMAGE}(IS, \Gamma, S')$ can be computed in polynomial time simply by checking all the states. Therefore, the computing time is polynomial. □

The results in this section give a provably correct polynomial time algorithm for the verification of ATLK^{3v} formulas. Not only is the algorithm attractive from a computational point of view, but it is amenable to symbolic implementation for efficient verification. In the next section we take this one step further and define methodologies to reduce the explicit model to more compact models.

4. ABSTRACTIONS

While the procedure put forward in the previous section is linear in the size of the model, verifying MAS still suffers from the state-space explosion, i.e., the fact that the models grow exponentially in the number of variables and agents in the system. To deal with this we use an abstraction methodology to reduce the size of the models considered. As we show below, this reduction generates an abstract model which depends on the specific characteristics of the interpreted system considered and the specification being checked.

First we recall the definitions below from [34]. A function f 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 functions $f.i$, $i = 1, \ldots, m$.

Definition 8 (State abstraction). *Consider an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i\in Ag}, I, \Pi)$. 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 S^σ of* abstract global states.

Given a state abstraction function σ, the state abstraction *of IS w.r.t. σ is the interpreted system $IS^\sigma = (\{L_i^\sigma, Act_i^\sigma, P_i^\sigma, t_i^\sigma\}_{i\in Ag}, I^\sigma, \Pi^\sigma)$ such that $\Pi^\sigma(s^\sigma) = \bigcap_{s\in\sigma^{-1}(\{s^\sigma\})} \Pi(s)$, $I^\sigma = \sigma(I)$ 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. *Consider an interpreted system with states s_1 labelled by $\{q, r\}$ and s_2 labelled by $\{\neg q, r\}$ such that $\sigma(s_1) = \sigma(s_2) = s^\sigma$. Then, $r \in \Pi^\sigma(s^\sigma)$ as both s_1 and s_2 are labelled by r, but neither $q \in \Pi^\sigma(s^\sigma)$ not $\neg q \in \Pi^\sigma(s^\sigma)$. Therefore, $IS^\sigma, s^\sigma \models^3 r = \mathrm{tt}$ and $IS^\sigma, s^\sigma \models^3 q = \mathrm{uu}$.*

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. One important requirement of the state abstraction function is that it can only merge states with the same protocol. In many applications this may be a significant limitation. To alleviate the impact of this, we provide a further abstraction function, called action abstraction. The goal of the action abstraction function is to reduce the number of actions and therefore the number of possible protocols in the system.

Definition 9 (Action abstraction). *Consider an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i\in Ag}, I, \Pi)$. An* action abstraction function *is a surjective and decomposable function $\alpha : ACT \to ACT^\alpha$, for some set ACT^α of* abstract joint actions.

Given an action abstraction function α, the action abstraction *of IS w.r.t. α 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, all $l, l' \in L_i$ and all $a^\alpha \in ACT^\alpha$, we have $L_i^\alpha = L_i$, $Act_i^\alpha = \alpha.i(Act_i)$, $P_i^\alpha(l) = \alpha.i(P_i(l))$ and $t_i^\alpha(l, a^\alpha, l')$ iff there is $a \in ACT$ s.t. $\alpha.i(a) = a^\alpha$ and $t_i(l, a, l')$.*

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. A *full abstraction* is obtained by combining the action abstraction with the state abstraction.

Definition 10 (Abstractions). *Consider an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i\in Ag}, I, \Pi)$. An* abstraction function *is a pair $\delta = (\alpha, \sigma)$, where α is an action abstraction function of IS and σ is an state abstraction function of IS^α.*

Given an abstraction function $\delta = (\alpha, \sigma)$, the abstraction *of IS w.r.t. δ is the interpreted system IS^δ, i.e., the state abstraction of IS^α w.r.t. σ.*

We can show that the truth values tt and ff are preserved from abstract model to concrete model if the abstract model is obtained from the concrete model by using the abstraction function above.

Theorem 11. *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 Sketch. The proof is by induction on φ. We only present the key cases here; the others can be shown similarly.

Case of $\varphi = q$. If $M^\delta_{IS}, \sigma(s) \models^3 q = \mathrm{tt}$, then $q \in \Pi^\delta(s)$. By definition $q \in \bigcap_{s'\in\sigma^{-1}(\sigma(s))} \Pi^\alpha(\sigma(s'))$, 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 $M_{IS}, s \models^3 q = \mathrm{ff}$.

Case of $\varphi = \neg\varphi'$. 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 $M_{IS}, s \models^3 \neg\varphi' = \mathrm{ff}$.

Case of $\varphi = \langle\!\langle\Gamma\rangle\!\rangle\varphi_1 U \varphi_2$. 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 some $X \in out(\sigma(s), F^\delta_\Gamma)$ and for all $p^\delta \in X$, 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 $F_\Gamma = \{f_i \mid i \in \Gamma\}$ where $f_i(l) = \alpha^{-1}(\sigma.i^{-1}(f^\delta_i(\sigma.i(l))))$ (notice that α^{-1} and $\sigma.i^{-1}$ are the counter-images). One can show using the inductive hypothesis that there exists $X \in out(s, F_\Gamma)$ such that for all $p \in X$ there is $k \geq 0$ s.t. $M_{IS}, p^k \models^3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$, $M_{IS}, p^j \models^3 \varphi_1 = \mathrm{tt}$. It follows that $M_{IS}, s \models^3 \langle\!\langle\Gamma\rangle\!\rangle\varphi_1 U \varphi_2 = \mathrm{tt}$.

Case of $\varphi = K_i\varphi'$. Assume that $M^\delta_{IS}, \sigma(s) \models^3 K_i\varphi' = \mathrm{tt}$. Observe that if $s_1 \sim_i s_2$, then $\sigma(s_1) \sim_i \sigma(s_2)$. Since for all $s^\gamma \sim_i \sigma(s)$ we have that $M^\delta_{IS}, s^\gamma \models^3 \varphi' = \mathrm{tt}$, then clearly for all $s' \sim_i s$ we have $M_{IS}, s' \models^3 \varphi' = \mathrm{tt}$, and therefore $M_{IS}, s \models^3 K_i\varphi' = \mathrm{tt}$.

If $M^\delta_{IS}, \sigma(s) \models^3 K_i\varphi' = \mathrm{ff}$, then $M^\delta_{IS}, \sigma(s) \models^3 \varphi' = \mathrm{ff}$, so $M_{IS}, s \models^3 \varphi' = \mathrm{ff}$ and thus $M_{IS}, s \models^3 K_i\varphi' = \mathrm{ff}$. □

In this section we have shown that constructions previously put forward in [34] for reasoning about ATL specifications with local strategies can be adapted to our present setting where epistemic modalities are also present and strategies are interpreted uniformly on the model.

5. MODEL CHECKING MAS VIA ABSTRACTION REFINEMENT

While in the previous section we presented the underlying principles of three-valued abstraction on interpreted systems, here we give a constructive methodology for generating and refining the abstract models from the concrete model. Our proposed algorithm for checking whether an interpreted system IS satisfies a specification φ is given by Algorithm 2.

The procedure verify-abs first constructs an initial abstraction and performs a finite number of refinement steps until one of two conditions holds: either no further refinement can be performed (the refinement function returns the identity) and the specification φ is undefined on the abstract model, or φ is calculated to be either ff or tt. Observe that

Algorithm 2 The verification procedure.

procedure VERIFY-ABS(IS, φ)
  $\delta \leftarrow \delta^{I}_{IS}(\varphi)$, $result \leftarrow$ uu
  while $\delta \neq (id, id)$ and $result =$ uu **do**
    $result \leftarrow$VERIFY(IS^{δ}, φ)
    $\delta \leftarrow$REFINE(IS, δ, φ)
  return $result$

the procedure VERIFY-ABS uses VERIFY which operates on the three valued semantics.

THEOREM 12 (CORRECTNESS). *Let IS be an interpreted system and φ be an ATLK specification. If* VERIFY-ABS(IS, φ) $=$ tt*, then* $IS \models \varphi$*; if* VERIFY-ABS$(IS, \varphi) =$ ff*, then* $IS \not\models \varphi$.

PROOF SKETCH. Observe from the constructions given below that all refinements are abstractions as per Definition 10; REFINE can only be invoked a finite number of times (see below). The results therefore follow by Theorem 11 and Theorem 6 by observing that if $M_{IS}, s \models^3 \varphi =$ tt, then $M_{IS}, s \models^2 \varphi$ and If $M_{IS}, s \models^3 \varphi =$ ff, then $M_{IS}, s \not\models^2 \varphi$. These two facts can be checked by induction on φ. $\square$

We now constructively define the initial abstraction $\delta^{I}_{IS}(\varphi)$ and the procedure REFINE used above.

Assume an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$. Let α^{Γ}_{IS} be the action abstraction function α for IS that unifies actions of all the agents not in Γ, e.g., for every agent $i \in \Gamma$ we define $\alpha^{\Gamma}_{IS}.i(a_i) = a_i$, and for every $i \notin \Gamma$ we let $\alpha^{\Gamma}_{IS}.i(a_i) = \epsilon$. Consider an action abstraction function α and a set of literals V. Let $\sigma^{\alpha,V}_{IS}$ denote the state abstraction function σ for IS^{α} that unifies all the possible states that agree on the labelling by variables in V, i.e., for each agent $i \in Ag$, $\sigma^{\alpha,V}_{IS}.i(l) = \sigma^{\alpha,V}_{IS}.i(l')$ iff $P^{\alpha}_i(l) = P^{\alpha}_i(l')$ and $\Pi(l) \cap V = \Pi(l') \cap V$, where $\Pi(l) = \bigcap_{s \in S, s.i = l} \Pi(s)$. Finally, define $A(\varphi)$ as the set of agents in φ and $\mathcal{V}(\varphi)$ as the set of propositional variables in φ.

DEFINITION 13 (INITIAL ABSTRACTION). *Let IS be an interpreted system, φ be an ATLK specification. The* initial abstraction function *for IS is the pair $\delta^{I}_{IS}(\varphi) = (\alpha^{I}_{IS}, \gamma^{I}_{IS})$, where $\alpha^{I}_{IS} = \alpha^{A(\varphi)}_{IS}$ and $\gamma^{I}_{IS} = \sigma^{\alpha^{I}_{IS}, \overline{\mathcal{V}(\varphi)}}_{IS}$. The* initial abstraction *of IS is the abstraction of IS w.r.t. $\delta^{I}_{IS}(\varphi)$.*

So the initial abstraction function groups together all the actions of the agents not referred to in the specification φ; all states with the same labelling and the same protocol are then further collapsed. Clearly any initial abstraction is an abstraction as in Definition 10; so Theorem 11 applies.

We conclude by presenting Algorithm 3 giving the refinement procedure REFINE which takes as input an interpreted system IS, an abstraction function δ, a specification φ and returns an abstraction function δ'. Intuitively if $IS^{\delta^{I}_{IS}(\varphi)}$ $\models^3 \varphi =$ uu, then δ' refines δ by separating the states for which the value of some subformula is uu.

In the procedure the set $Sub(\varphi)$ stands for the set of all the subformulas of φ (including φ itself) and $\preceq$ is any linear order on $Sub(\varphi)$ such that if ψ is a subformula of ψ', then $\psi \preceq \psi'$. Let the set $\text{uu}^{IS}(\delta, \varphi)$ be the set of subformulas of φ that in at some abstract state of IS^{δ} corresponding to at least two concrete states have the unknown value.

As shown, REFINE computes the set $\Phi^{IS}((\alpha, \sigma), \varphi)$. If this is not empty, the smallest element ψ is considered and all

Algorithm 3 The refinement procedure.

procedure REFINE(IS, $\delta = (\alpha, \sigma)$, φ)
  if $\Phi^{IS}((\alpha, \sigma), \varphi) \neq \emptyset$ **then**
    $\alpha' \leftarrow \alpha$
    $\Phi_{uu} = min_{\leq}(\text{uu}^{IS}((\alpha, \sigma), \varphi))$
    $(S_{\text{tt}}, S_{\text{ff}}) =$ LABEL(IS^{σ}, Φ_{uu})
    $S_{\text{uu}} \leftarrow \{s \in S \mid \sigma(s) \notin S_{\text{tt}} \cup S_{\text{ff}}\}$
    for $i \in Ag, l_i \in L_i$ **do**
      if $\exists s \in S_{\text{uu}}\ s.i = l_i$ **then**
        if $\exists s, s' \in S_{\text{uu}}.\exists i \in Ag.\sigma.i(s.i) = \sigma.i(s'.i)$ $\wedge\, \Pi(s.i) = \Pi(s'.i)$ **then**
          $\sigma'.i(l_i) \leftarrow (\sigma.i(l_i), \Pi(l_i))$
        else $\sigma'.i(l_i) \leftarrow l_i$
      else $\sigma'.i(l_i) \leftarrow \sigma.i(l_i)$
  else if $\alpha \neq id$ **then**
    $\alpha' \leftarrow id$, $\sigma' \leftarrow \sigma^{\alpha', \overline{\mathcal{V}(\varphi)}}_{IS}$
  else $\alpha' \leftarrow id$, $\sigma' \leftarrow id$
  return (α', σ')

the abstract states s^{δ} in which the value of ψ is uu are split. When this operation can no longer be performed the action abstraction function is refined. Since all sets considered are finite, the procedure terminates.

6. THE BIT TRANSMISSION PROTOCOL

In this section we consider a variation of the bit transmission protocol and show how the technique put forward so far can be used to verify it. We assume the system to be composed of three agents: a sender **S**, whose goal is to send 32 bits to a recipient **R**, who receives the bits and once all of these have been copied sends an acknowledgement to the sender; and the communication channel **CC** that controls whether messages are dropped. We refer to its original formulation and analysis for more details [19, 42].

We model the protocol as an interpreted system IS as follows. We take the communication channel agent as modelled by a single state ($L_{\mathbf{CC}} = \{\epsilon\}$) and four allowed actions, $Act_{\mathbf{CC}} = P_{\mathbf{CC}}(\epsilon) = \{-, \leftarrow, \rightarrow, \leftrightarrow\}$ intuitively corresponding to the direction in which communication is performed. The local transition function is defined as standard $t_{\mathbf{SS}}(\epsilon, \text{a}, \epsilon)$ for all the joint actions a.

The sender **S** is modelled through 2^{32} *regular* local states, one for each possible message to be sent, and two *special* states: $\perp$ denoting an internal problem and ✓ representing a state when the message was delivered. The actions of **S** are of the form $\text{b}^{\text{v}}_{\text{i}}$, where $i \in \{0, \ldots, 31\}$ and $v \in \{0, 1\}$. An action $\text{b}^{\text{v}}_{\text{i}}$ corresponds to sending the information that "the value of i-th bit is v". Formally, we consider the local states $L_{\mathbf{S}} = \{0, 1\}^{32} \cup \{\perp, ✓\}$, the local actions $Act_{\mathbf{S}} = \{\text{b}^{\text{v}}_{\text{i}} \mid i \in \{0, \ldots, 31\} \wedge v \in \{0, 1\}\}$, a fully non-deterministic protocol $P_{\mathbf{S}}(s) = Act_{\mathbf{S}}$ for all states S of the model, and a local transition function $t_{\mathbf{S}}(s, (\text{a}_{\mathbf{S}}, \text{a}_{\mathbf{CC}}, \text{a}_{\mathbf{R}}), s')$ iff

- $s = s'$ and $s \in \{\perp, ✓\}$, or
- $\text{a}_{\mathbf{CC}} \in \{\rightarrow, \leftrightarrow\}$, $\text{a}_{\mathbf{R}} = \text{ack}$, $s' = ✓$, or
- $s = b_0 b_1 \ldots b_{31}$, $\text{a}_{\mathbf{S}} = b^{v}_{i}$, $v = b_i$, $s' = s$, and either $\text{a}_{\mathbf{CC}} \notin \{\rightarrow, \leftrightarrow\}$ or $\text{a}_{\mathbf{R}} \neq \text{ack}$, or
- $s = b_0 b_1 \ldots b_{31}$, $\text{a}_{\mathbf{S}} = b^{v}_{i}$, $v \neq b_i$, $s' = \perp$.

The receiver $\mathbf{R}$ is modelled by considering 3^{32} states, $L_{\mathbf{R}} = \{0, 1, ?\}^{32}$, and two actions, $Act_{\mathbf{R}} = \{\epsilon, \text{ack}\}$. The protocol function is such that $P_{\mathbf{R}}(s) = \{\text{ack}\}$ if $s \in \{0,1\}^{32}$ and $P_{\mathbf{R}}(s) = \{\epsilon\}$ otherwise. The transitions are given by $t_{\mathbf{R}}(b_0 b_1 \ldots b_{31}, (\mathrm{b}_i^v, \mathrm{a}_{\mathbf{CC}}, \mathrm{a}_{\mathbf{R}}), b'_0 b'_1 \ldots b'_{31})$ iff

- $\mathrm{a}_{\mathbf{CC}} \in \{\rightarrow, \leftrightarrow\}$, $b'_i = v$ and for all $j \neq i$, $b'_j = b_j$, or
- $\mathrm{a}_{\mathbf{CC}} \notin \{\rightarrow, \leftrightarrow\}$ and for all j, $b'_j = b_j$.

The set of global initial states is given by $I = \{0,1\}^{32} \times \{\epsilon\} \times \{?^{32}\}$, representing the fact that $\mathbf{S}$ starts with any message to be sent and $\mathbf{R}$ has not copied any bit yet.

We use the propositional variable $\mathcal{ACK}$ to label the states where the state of $\mathbf{R}$ is $\checkmark$, *fail* to mark the states where the state of $\mathbf{R}$ is $\bot$, $\mathcal{S}_i^v$ to denote the states of the form $(b_0 b_1 \ldots b_{31}, \epsilon, l_{\mathbf{R}})$, where $b_i = v$, and $\mathcal{R}_i^v$ to indicate the states of the form $(l_{\mathbf{S}}, \epsilon, b_0 b_1 \ldots b_{31})$ where $b_i = v$.

By considering the global transition function from the initial state, we can compute the set of reachable global states S for the model M associated to the interpreted system IS here described. The size of S can be estimated as $4^{32} \approx 10^{19}$. This is beyond what any modern symbolic model checker would normally be able to compute.

We can use the logic ATLK defined earlier to state specifications of the protocol. For instance we may be interested to check whether $\mathbf{S}$ and $\mathbf{CC}$ may be able to ensure that eventually $\mathbf{R}$ knows the value v of the i-th bit, for some $i \in \{0, \ldots, 31\}$ and $v \in \{0, 1\}$:

$$\Phi_1 = (\mathcal{S}_i^v \rightarrow \langle\!\langle \{\mathbf{S}, \mathbf{CC}\} \rangle\!\rangle F K_{\mathbf{R}} \mathcal{R}_i^v)$$

Intuitively we would expect Φ_1 to be satisfied. Naturally, $\mathbf{S}$ is not on his own able to ensure the delivery of the messages. Indeed, we would expect the specification

$$\Phi_2 = \langle\!\langle \{\mathbf{CC}\} \rangle\!\rangle G \neg \mathcal{ACK}$$

to be satisfied as without any fairness assumption $\mathbf{CC}$ could simply block all messages.

Lastly, we may want to check whether $\mathbf{S}$ and $\mathbf{CC}$ may be able to guarantee that the whole message is delivered:

$$\Phi_3 = \langle\!\langle \{\mathbf{S}, \mathbf{CC}\} \rangle\!\rangle F \mathcal{ACK}$$

We illustrate the use of the abstraction technique above on the interpreted system IS and the specifications Φ_1, Φ_2, Φ_3.

The initial action abstraction for Φ_1 collapses all the actions of $\mathbf{R}$. The state abstraction generates an interpreted system with four states only: $S^1 = \{(s_1^{\mathbf{S}}, \epsilon, s_1^{\mathbf{R}}), (s_1^{\mathbf{S}}, \epsilon, s_2^{\mathbf{R}}), (s_2^{\mathbf{S}}, \epsilon, s_1^{\mathbf{R}}), (s_2^{\mathbf{S}}, \epsilon, s_2^{\mathbf{R}})\}$. The initial states I are $(s_1^{\mathbf{S}}, \epsilon, s_1^{\mathbf{R}})$ and $(s_2^{\mathbf{S}}, \epsilon, s_1^{\mathbf{R}})$. It is easy to check that $\textsc{verify}(IS^1, \Phi_1)$ returns tt on this small system. By Theorem 6 we can deduce that $M \models \Phi_1$.

The initial abstraction for Φ_2 results in an interpreted system IS^2 with two global states only $S^2 = \{s, s'\}$ such that s satisfies $\mathcal{ACK}$ and s' satisfies $\neg\mathcal{ACK}$. The state s' is the only initial state; the transitions in the system are: $t(s', \mathrm{a}, s')$ for all the possible joint actions a, and $t(s', (\mathrm{a}_{\mathbf{S}}, \mathrm{a}_{\mathbf{CC}}, \mathrm{a}_{\mathbf{R}}), s)$ for all the actions such that $\mathrm{a}_{\mathbf{CC}} \in \{-, \Rightarrow\}$. Clearly, we have that $\textsc{verify}(IS^2, \Phi_2) = \text{tt}$ which, as above, enables us to conclude that $M \models \Phi_2$.

Verifying Φ_3 leads us to the same initial abstraction $IS^3 = IS^2$. The procedure $\textsc{Label}(IS, \Phi_3)$ returns $(\{s\}, \emptyset)$; so $\textsc{verify}(IS^3, \Phi_3) = \text{uu}$. $\textsc{Verify-Abs}$ therefore calls for the refinement of IS^3 by means of the $\textsc{Refine}$ procedure that splits the abstract states into the concrete states, resulting in an interpreted system IS^4. The procedure determines that the smallest subformula whose value cannot be determined is Φ_3 itself and that its value is unknown at s'. Therefore, the procedure splits the abstract state s' into concrete states leading to the interpreted system IS^4. It can be checked that $\textsc{Verify}(IS^4, \Phi_3) = \text{tt}$. As before, it therefore follows that $M \models \Phi_3$. Even after the refinement takes place on the resulting IS^4 all the states satisfying v are abstracted into one abstract state only; so IS^4 has $2^{32} - 1$ fewer states than the concrete model. It can be shown that no smaller abstraction would allow determining the truth value of Φ_3.

In summary, the abstraction refinement here put forward enabled us to determine the value of ATLK specifications on models that would be too large to verify by means of any state-of-the-art model checker.

7. CONCLUSIONS AND RELATED WORK

In this paper we have developed the theoretical underpinnings for a predicate abstraction methodology for verifying MAS against epistemic specifications. We put forward a temporal-epistemic specification language which includes some features expressing a weak form of strategic reasoning; we gave a three-valued semantics so that this can be used in an abstraction setting and showed that, differently from other approaches, its model checking problem has PTime complexity. The initial abstraction and the refinement steps are given constructively and can be computed automatically; the scenario we discussed demonstrates potential significant gains of the technique.

Abstractions preserving epistemic properties of MAS have previously been introduced [14, 3, 31, 18]. However, all of these rely on ad-hoc constructions in which human intervention is required to generate the abstract model. Additionally, none of them is intended for three-valued model approximations as we do here. Much closer to our approach is instead [34] which markedly differs from the present investigation by addressing ATL specifications only. In contrast, we here show that these can be extended to the epistemic case. In addition to its increased expressivity, the present setup also offers a very attractive PTime complexity for the model checking problem, whereas this is Δ_2^P in the case of [34]. This is essential for our future work in which we plan to investigate the verification of MAS when these are given by compact representations such as reactive modules [1] and ISPL [36]. In this case our technique is expected to have a PSPACE bound, whereas we would expect the technique from [34] to become doubly-exponential making any concrete application problematic.

Further ahead we intend to apply these ideas to limited classes of agent programs. Verification methods for MAS programs have of course been discussed prominently in the literature [16, 47, 5, 46, 7]. Even if these results have proven useful, the performance of these is limited due to the state-space explosion. One of the most promising techniques to overcome this is predicate abstraction. We see the development of MAS-oriented underpinnings as a necessary step before predicate abstraction methodologies for MAS can be implemented.

ACKNOWLEDGEMENTS

This research was funded by the EPSRC under grant EP/I00520X.

8. REFERENCES

[1] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proceedings of the 10th International Conference on Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 521–525. Springer-Verlag, 1998.

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

[3] O. A. Bataineh and R. V. D. Meyden. Abstraction for epistemic model checking of dining cryptographers-based protocols. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK11)*, pages 247–256. ACM, 2011.

[4] F. Belardinelli, A. V. Jones, and A. Lomuscio. Model checking temporal-epistemic logic using alternating tree automata. *Fundamenta Informaticae*, 112(1):19–37, 2011.

[5] R. H. Bordini, L. A. Dennis, B. Farwer, and M. Fisher. Automated verification of multi-agent programs. In *23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008)*, pages 69–78. IEEE, 2008.

[6] R. H. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proceedings of the 15th International Conference on Computer Aided Verification (CAV03)*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag, 2003.

[7] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[8] R. H. Bordini, R. H. Hubner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason.* John Wiley, 2007.

[9] T. Brihaye, A. D. Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In *Logical Foundations of Computer Science*, pages 92–106. Springer, 2009.

[10] N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In *Specification and Verification of Multi-agent Systems*, pages 125–159. Springer, 2010.

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

[12] P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Computer Aided Verification*, volume 8559 of *Lecture Notes in Computer Science*, pages 525–532. Springer, 2014.

[13] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.

[14] M. Cohen, M. Dam, A. Lomuscio, and H. Qu. A symmetry reduction technique for model checking temporal-epistemic logic. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI09)*, pages 721–726, Pasadena, USA, 2009.

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

[16] L. Dennis, M. Fisher, M. Webster, and R. H. Bordini. Model checking agent programming languages. *Automated Software Engineering*, 19(1):5–63, 2012.

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

[18] C. Enea and C. Dima. Abstractions of multi-agent systems. In *Proceedings of 5th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS07)*, volume 4696 of *Lecture Notes in Computer Science*, pages 11–21. Springer, 2007.

[19] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[20] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

[21] S. Graf and H. Saïdi. Construction of abstract state graphs with pvs. In *Proceedings of the 9th International Conference on Computer Aided Verification (CAV97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1997.

[22] W. Hoek and M. Wooldridge. Model checking knowledge and time. In *SPIN 2002 – Proceedings of the Ninth International SPIN Workshop on Model Checking of Software*, 2002.

[23] W. Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[24] X. Huang and R. van der Meyden. Symbolic model checking epistemic strategy logic. In *AAAI*, pages 1426–1432, 2014.

[25] X. Huang and R. van der Meyden. A temporal logic of strategic knowledge. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR14)*. AAAI Press, 2014.

[26] W. Jamroga and J. Dix. Model checking abilities under incomplete information is indeed δ_p^2-complete. *EUMAS*, pages 14–15, 2006.

[27] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 62:1–35, 2004.

[28] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic. Master's thesis, University of Utrech, The Netherlands, 2003.

[29] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny. Verics 2007 - a model checker for knowledge and real-time. *Fundamenta Informaticae*, 85(1):313–328, 2008.

[30] M. Kacprzak and W. Penczek. Fully symbolic unbounded model checking for alternating-time temporal logic. *Autonomous Agents and Multi-Agent Systems*, 11(1):69–89, 2005.

[31] M. Koleini, E. Ritter, and M. Ryan. Model checking agent knowledge in dynamic access control policies. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2013.

[32] S. Lahiri, R. Nieuwenhuis, and A. Oliveras. Smt techniques for fast predicate abstraction. In *Computer Aided Verification*, pages 424–437. Springer, 2006.

[33] A. Lomuscio and J. Michaliszyn. An epistemic Halpern-Shoham logic. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI13)*, pages 1010–1016. AAAI Press, 2013.

[34] A. Lomuscio and J. Michaliszyn. An abstraction technique for the verification of multi-agent systems against ATL specifications. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR14)*, pages 428–437. AAAI Press, 2014.

[35] A. Lomuscio, W. Penczek, and B. Woźna. Bounded model checking knowledge and real time. *Artificial Intelligence*, 171(16-17):1011–1038, 2007.

[36] A. Lomuscio, H. Qu, and F. Raimondi. 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*, pages 682–688. Springer, 2009.

[37] A. Lomuscio and F. Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of the 5th International Joint Conference on Autonomous agents and Multi-Agent Systems (AAMAS06)*, pages 161–168. ACM Press, 2006.

[38] R. Meyden. Knowledge based programs: On the complexity of perfect recall in finite environments. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, pages 31–50. Morgan Kaufmann Publishers, 1996.

[39] F. Mogavero, A. Murano, and M. Vardi. Reasoning About Strategies. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS10)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 133–144. Schloss Dagstuhl, 2010.

[40] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[41] J. Pilecki, M. Bednarczyk, and W. Jamroga. Synthesis and verification of uniform strategies for multi-agent systems. In *Computational Logic in Multi-Agent Systems*, volume 8624 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2014.

[42] F. Raimondi and A. Lomuscio. Automatic verification of deontic interpreted systems by model checking via OBDDs. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI04)*, pages 53–57. IOS PRESS, 2004.

[43] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: an algorithm and its implementation. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 630–637. ACM, 2004.

[44] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 5(2):235–251, 2005.

[45] F. Raimondi and A. Lomuscio. Model checking knowledge, strategies, and games in multi-agent systems. Technical Report RN/05/01, Department of Computer Science, UCL, London, UK, 2005.

[46] D. T. Trang, Y. Yao, N. Alechina, and B. Logan. Verifying heterogeneous multi-agent programs. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 149–156, 2014.

[47] M. Wooldridge, M. Fisher, M. Huget, and S. Parsons. Model checking multiagent systems with MABLE. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS02)*, pages 952–959, Bologna, Italy, 2002.

[48] B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for Timed Automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.