Agent-based Refinement for Predicate Abstraction of Multi-Agent Systems

Francesco Belardinelli[1] and **Alessio Lomuscio**[1] and **Jakub Michaliszyn**[2]

Abstract. We put forward an agent-based refinement methodology for the verification of infinite-state Multi-Agent Systems by predicate abstraction. We use specifications defined in a three-valued variant of the temporal epistemic logic ATLK. We define "failure states" as candidates for refinement, and provide a sound automatic procedure for their identification. Further, we introduce a methodology based on Craig's interpolants for the refinement of the agent-specific predicates upon which the abstraction is built. We illustrate the refinement technique on an infinite-state auction scenario, and show that specifications of interest, that could not be checked by plain abstraction, can now be verified on the refined models.

1 Introduction

Over the past 15 years, considerable research has taken place in the area of verification of *finite state* Multi-Agent Systems (MAS). This includes symbolic model checking methods [13, 25], SAT-based methods [31], partial-order reductions [23], and symmetry reduction [8]. Considerably less attention has so far been paid to devising techniques for verifying *infinite state* MAS. Since, like standard programs, MAS typically denote infinite models, devising techniques for verifying infinite state MAS remains of considerable interest.

Predicate abstraction [9, 19] is a successful approach to the verification of infinite state programs. In predicate abstraction finite state models, representing under- and over-approximations of the system, are generated automatically by Boolean programs built on predicates derived from the program's and the system's specifications. If the truth value of the specification cannot immediately be determined on the initial Boolean program, the list of predicates is updated automatically and a new Boolean program is generated and checked. While this procedure cannot be complete due to the undecidability of the underlying problem, by checking several refinements in succession it is often possible to determine the truth value of the specification on the infinite-state program. A key aspect of this approach is the actual derivation of the refined abstractions.

While predicate abstraction is an established technique in software verification, considerable challenges need to be overcome before it can be applied to MAS. These include the fact that MAS semantics are modular in the agents and that agent-based specifications are considerably richer than those traditionally used in software engineering. Any predicate abstraction technique for MAS ought to support these aspects.

In this paper we introduce a refinement technique for verifying MAS against specifications defined in the agent-based logic ATLK.

[1] Department of Computing, Imperial College London, UK
[2] Institute of Computer Science, University of Wrocław, Poland

A key aspect of the approach we take is the particular setup we consider when combining ATL [2] and epistemic logic [12, 30] to form the logic ATLK that we use to specify MAS. ATL is most often used in its original variant that assumes systems with perfect recall and complete information. This setup is attractive from a verification perspective as the corresponding model checking problem is PTIME, like CTL and CTLK [11, 26]. In contrast, epistemic logic is defined on the basis of incomplete information. This creates a tension in combinations such as ATEL [15], where ATL and epistemic modalities do not share the same underlying information model for the agents in the system. Proposals have been made to overcome this modelling difficulty. The natural setting involves assuming incomplete information for both the strategic ATL modalities and the epistemic operators [17]. If memoryless, uniform strategies [18] are assumed, this leads to a decidable model checking problem; the resulting complexity is however Δ_2^P-complete [16]. In turn this makes the model checking problem exponential against implicit structures given by modelling languages. Given the difficulty of model checking large state spaces, any practical prospect of verifying MAS is unfeasible under this assumption.

To solve the difficulty above we here work with a variant of ATLK which is defined on incomplete information and memoryless, non-uniform strategies. Under non-uniform strategies, agents do not have to play the same action in the same local state, as long as the action is allowed by their protocol. This set up has been proposed in [27] and used in a number of applications [25]. Under this setting ATLK retains a PTIME model checking problem and verification can be performed via fixed-point characterisations of the ATL operators. The semantics of non-uniform strategies is considerably more convoluted and was formally presented in [20, 21, 22]. In this paper we adopt this framework, which we recall in the next section. However, we refer to these papers for more motivations, discussions of these features, as well as relationship with alternative assumptions, including plain ATEL (see [1]). We stress that the framework here proposed entirely subsumes CTLK, for which no predicate refinement methodology has been proposed yet.

Related Work. Other than the contributions hereafter, we are aware of no work addressing the verification of infinite-state MAS by predicate abstraction. In [32, 3] the authors define a predicate abstraction methodologies supporting CTL and Alternating Modal Logic (AML) specifications. This work differs from the present one in several respects. Firstly, their specification language does not support epistemic modalities. Secondly, the AML semantics assumes complete information and perfect recall; instead we only assume incomplete information and no memory. Thirdly, no method is given for the refinement of predicates. In contrast, we here put forward an algorithm based on Craig's interpolants that is used to generate successive refinements from the agents' models.

Closer to our work are [21, 22] where a three valued logic ATLK is defined and a procedure for an agent-based state and action abstraction is given. However, no solution is proposed there for performing refinement on the abstract models. So if a specification is initially undefined, no conclusion can be drawn. We here follow that approach, but extend it by identifying so called "failure pairs", which we exploit to build a refined model. This enables us to solve the verification problem in several cases of interest where the original technique fails.

Predicate abstraction for the verification of MAS against temporal-epistemic specifications was proposed in [14]. Our contribution differs from that work as we support ATL specifications; the underlying semantics is different; also while [14] addresses the specific case of GSM programs, here we deal with generic MAS; lastly, in common with [22], [14] cannot deal with predicate refinement, which constitutes our main contribution here.

Scheme of the paper. The paper is structured as follows. In Sections 2 we summarise the methodology from [22] for initial abstraction on MAS. In Section 3 we define the concept of *failure pair*; define an algorithm for their identification, and study its properties. We adopt these in Section 4 to derive Craig's interpolants that we use to revise the list of predicates in the initial abstraction. We exemplify the technique in Section 5 and conclude in Section 6 by pointing to future work.

2 Predicate Abstraction for MAS

In this paper we assume that agents have imperfect information and use memoryless (positional) strategies [4, 27]; this is in contrast with previous approaches [32, 3] that assume perfect information. We initially follow the three-valued abstraction methodology introduced by [20, 21], that we summarise hereafter. In the following $Ag = \{1, \dots, m\}$ is a set of agents and $\mathcal{V}$ a set of propositions. Given a set U, $\overline{U}$ denotes its complement (w.r.t. some $V \supseteq U$).

We first define the notion of *interpreted system* [12], to represent formally the execution of a multi-agent system.

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

- *for each agent* $i \in Ag$*,*
  - L_i *is the (possibly infinite) set of* local states*;*
  - Act_i *is the set of* actions*;*
  - $P_i : L_i \to (2^{Act_i} \setminus \{\emptyset\})$ *is the* local protocol*;*
  - $t_i \subseteq L_i \times ACT \times L_i$ *is the* local transition relation*, where* $ACT = Act_1 \times \cdots \times Act_{|Ag|}$ *is the set of* joint actions*;*
- $I \subseteq L_1 \times \cdots \times L_{|Ag|}$ *is the set of* global initial states*;*
- $\Pi : L_1 \times \cdots \times L_{|Ag|} \times \mathcal{V} \to \{\mathrm{tt}, \mathrm{ff}, \mathrm{uu}\}$ *is the* labelling function.

By Def. 1 each agent i in an interpreted system is assumed to perform the actions in Act_i, according to protocol P_i. Differently from the standard notion of IS [12], here the transition function is local [24], and propositional atoms can receive three truth values: true tt, false ff, and undefined uu. We say that a truth value t is *defined* whenever $t \neq \mathrm{uu}$. Also, $v.i$ denotes the $i+1$-th element of tuple v.

Given an IS M, we introduce the *global transition relation* $T \subseteq S \times ACT \times S$ such that $T(s, a, s')$ holds iff for all $i \in Ag$,

- $t_i(s.i, a, s'.i)$, and
- $a.i \in P_i(s.i)$.

Then, $S \subseteq L_1 \times \cdots \times L_{|Ag|}$ denotes the set of *global states*, reachable by the global transition relation T from the set I of initial states. Finally, for every $i \in Ag$, $\sim_i \subseteq S^2$ is the *epistemic indistinguishability relation* defined as $s \sim_i s'$ iff $s.i = s'.i$ [12]. Given a set $\Gamma \subseteq Ag$ of agents, the relation $\sim_\Gamma$ is the transitive closure of $(\bigcup_{i\in\Gamma} \sim_i)$. In the following we assume that our models are *non-terminating*, i.e., for every $s \in S$ and enabled joint action $a \in ACT$, $T(s, a, s')$ holds for some state $s' \in S$.

In this paper we analyse two logics built on the same syntax, but with different semantics: the two-valued logic ATLK^{2v} and the three-valued logic ATLK^{3v}.

Definition 2 (ATLK) *Formulas in the logics ATLK^{2v} and ATLK^{3v} are defined as follows:*

$$\varphi ::= q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle\Gamma\rangle\rangle X\varphi \mid \langle\langle\Gamma\rangle\rangle(\varphi U \varphi) \mid \langle\langle\Gamma\rangle\rangle G\varphi \mid C_{\Gamma'}\varphi$$

where $q \in \mathcal{V}$*,* $i \in Ag$*,* $\Gamma, \Gamma' \subseteq Ag$*, and* $\Gamma' \neq \emptyset$*.*

The logic ATLK is an epistemic extension of Alternating-time Temporal Logic [17, 27], including the common knowledge operator $C_{\Gamma'}$. We refer to [27, 21] for the reading of modalities and derived operators in the context of the present semantics. We use abbreviations to introduce $\langle\langle\Gamma\rangle\rangle F\varphi$, the remaining propositional connectives, and ATLK operators. In particular, for every agent $i \in Ag$, we define the individual knowledge operator K_i as $C_{\{i\}}$.

In order to provide a semantics to ATLK by means of interpreted systems, we introduce the notion of a *memoryless strategy* for agent $i \in Ag$ as a function $f_i : L_i \to (2^{Act_i} \setminus \emptyset)$ such that for every local state $l \in L_i$, $f_i(l) \subseteq P_i(l)$. Given a path $p = s_0 s_1 \dots$, p^i denotes the $i+1$-th element s_i in p. Given a set $F_\Gamma = \{f_i \mid i \in \Gamma\}$ of strategies, one for each agent $i \in \Gamma$, a set X of paths is F_Γ*-compatible* if it is a minimal, non-empty set of paths such that for every $p \in X$, position $j \geq 0$, joint actions a, a', and state s', if $T(p^j, a, p^{j+1})$, $T(p^j, a', s')$, and for every $i \in \Gamma$, $a'.i = a.i \in f_i(p^j.i)$, then there exists some path $p' \in X$ starting with $p^0, \dots, p^j, s'$. Let $out(s, F_\Gamma)$ be the family of all F_Γ-compatible sets of paths starting from s.

We briefly comment on the notions of strategy and compatible path just introduced. Specifically, we assume that strategies are non-uniform in the sense of [27], i.e., agents can execute different actions at different global states in which their own local state is the same. This is in contrast with both the original semantics for ATL [2], which stipulates complete information of the global state, and with successive proposals to accommodate imperfect information [17]. However, it can be proved that the present formulation and the perfect information account of [2] are logically equivalent in the sense that an ATL formula is true in the setting we here adopt if and only if the formula is true in the semantics adopted in [2]. It follows that, for the two-valued fragment, an ATLK formula holds in an interpreted system under the present semantics if it holds in the ATEL logic in [15]. In particular, the fixed point characterisations of ATL operators hold in the present setting.

Finally, we report the three-valued satisfaction relation $\models^3$ from [21]. We assume the Kleene semantics for the standard boolean connectives. For the ATL and knowledge modalities, the semantic is defined as follows.

Definition 3 (Satisfaction) *The 3-valued satisfaction relation* $\models^3$

for an IS M, state $s \in S$, and formula φ is defined as follows:

$$M, s \models^3 \langle\langle\Gamma\rangle\rangle X\varphi =$$

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

$$M, s \models^3 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 =$$

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

$$M, s \models^3 \langle\langle\Gamma\rangle\rangle G\varphi =$$

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

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

In all other cases, the value of formula φ is undefined (uu*).*

The two-valued satisfaction relation $\models^2$ can be derived from $\models^3$ by considering clauses for tt only, as well as classic negation. An IS M satisfies property φ in ATLK, or $M \models^2 \varphi$, iff for all initial states $s \in I$, $(M, s) \models^2 \varphi$. Similarly, $(M \models^3 \varphi) = \mathrm{tt}$ (resp. ff) iff for all (resp. some) $s \in I$, $((M, s) \models^3 \varphi) = \mathrm{tt}$ (resp. ff). Otherwise, $(M \models^3 \varphi) = \mathrm{uu}$.

In [21] it is shown that that for every φ in ATLK, $(M \models^3 \varphi) = \mathrm{tt}$ implies $M \models^2 \varphi$ and $(M \models^3 \varphi) = \mathrm{ff}$ implies $M \not\models^2 \varphi$. That is, defined truth values are preserved.

Since interpreted systems might have a possibly infinite state space, abstraction techniques have been developed to make verification feasible [7, 6]. In this section we describe the agent-based abstraction techniques put forward in [21, 22] that uses predicates derived from the system description and the specification to be checked.

Assume an IS M and a list $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$ of tuples of predicates, where intuitively each predicate represents a condition on an agent's protocol, or transition relation. The satisfaction of conjunctions c of literals (predicates and their negation), called *cubes*, can naturally be given at an agent's local state, denoted as $l_i \models c$. A cube is *satisfiable* iff it is satisfied by some local state. By using predicates, agent descriptions can be abstracted as follows.

Definition 4 (Abstract Agent) *Given an agent $i \in Ag$ and a list $\vec{p}_i$ of predicates, the* abstract agent *is a tuple $i^A = \langle L_i^A, Act_i, P_i^{may}, P_i^{must}, t_i^{may}, t_i^{must} \rangle$ such that:*

- *L_i^A is the set of all satisfiable cubes;*
- *the* may protocol *P_i^{may} is such that $a \in P_i^{may}(c)$ iff for some $l \in L_i$, $l \models c$ and $a \in P_i(l)$;*
- *the* may relation *t_i^{may} is such that $t_i^{may}(c, a, c')$ iff for some local states $l, l' \in L_i$, $l \models c$, $l' \models c'$, and $t_i(l, a, l')$;*
- *the* must protocol *P_i^{must} is such that $a \in P_i^{must}(c)$ iff for every $l \in L_i$, $l \models c$ implies $a \in P_i(l)$;*
- *the* must relation *t_i^{must} is such that $t_i^{must}(c, a, c')$ iff for all $l \in L_i$, if $l \models c$ then $t_i(l, a, l')$ for some l' satisfying c'.*

We say that a global state $s \in S$ satisfies a tuple $b = (c_1, \ldots, c_{|Ag|})$ of cubes, denoted as $s \models b$, if each $s.i$ satisfies c_i.

Definition 5 (Abstract IS) *The* predicate abstraction *of an IS M w.r.t. predicates $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$ is the IS $M^A = (Ag^A, I^A, \Pi^A)$, where:*

- *Ag^A is the set of abstract agents i^A w.r.t. $\vec{p}_i$, for $i \in Ag$;*
- *for every state $b \in S^A$ (where $S^A = L_1^A \times \cdots \times L_{|Ag|}^A$), $q \in \mathcal{V}$ and $t \in \{\mathrm{tt}, \mathrm{ff}\}$, $\Pi^A(b, q) = t$ iff $\Pi(s, q) = t$ for all states s satisfying b;*
- *$I^A = \{b \mid$ for some $s \in I, s \models b\}$.*

Furthermore, for every $\Gamma \subseteq Ag$, the abstract transition relation $T_\Gamma^A(b, a, b')$ holds iff

- for all $i \in \Gamma$, $a.i \in P^{must}(b.i)$ and $t_i^{must}(b.i, a, b'.i)$;
- for all $i \notin \Gamma$, $a.i \in P^{may}(b.i)$ and $t_i^{may}(b.i, a, b'_i)$.

Intuitively, the *may* and *must* components of abstract IS can be seen respectively as over- and under-approximations of the strategic abilities of agents. In the following we use the notion of *(immediate) successor* according to relations T_Γ^A and $T_{\overline{\Gamma}}^A$.

An abstraction M^A can be used to interpret the language ATLK according to the three-valued semantics. In particular, the following preservation result applies [21].

Theorem 6 *Let M be an IS with predicate abstraction M^A. For every ATLK property φ,*

$$(M^A \models^3 \varphi) = \mathrm{tt} \quad \text{implies} \quad M \models^2 \varphi;$$
$$(M^A \models^3 \varphi) = \mathrm{ff} \quad \text{implies} \quad M \not\models^2 \varphi.$$

In [22] the result above is exploited to give a methodology for verifying infinite-state MAS. Starting from the infinite-state agents' descriptions and specifications, the relevant predicates, which are then used to construct the abstract, finite-state interpreted system are derived. The MAS specifications are then evaluated on it. If the truth value is defined, it can be deduced whether or not the specification holds on the original MAS. If the specification is undefined, no conclusion can be drawn. Indeed, [22] provides such an example where the technique is unable to determine the value of a specification.

In what follows we put forward a methodology for iteratively refining the agent-specific predicates so that finer and finer abstractions can be constructed and the truth value of the specification may be determined.

3 Identifying Failure Pairs

In this section, inspired by [3], we define a refinement procedure based on Craig's interpolants. Specifically, given an ATLK formula φ, undefined in some state c of an abstract IS M, we investigate the reason for the undefinedness of φ. To do so, we introduce the notion of failure pair and provide an algorithm for their identification. Differently from [3], which considers in detail only the sublanguage of ATLK containing operator $\langle\langle A\rangle\rangle X$ (i.e., Alternating Modal Logic), here we account for ATLK, including epistemic operators. The procedure we define is agent-based and therefore modular, whereas in [3] the abstraction is defined at the system level.

Since in this section we only work on abstract models, for convenience we will denote them simply as M.

Definition 7 (Relevant Pair) *Given a (abstract) state c and a formula φ, the function R, which returns the set of* pairs relevant *for the truth of φ in c, is the smallest function (w.r.t. the Lorenz order) satisfying the following conditions for each c, ψ and ψ'.*

- $R(c, p) = \{(c, p)\}$, *for* $p \in AP$
- $R(c, \neg\psi) = \{(c, \psi)\} \cup R(c, \psi)$
- $R(c, \psi \wedge \psi') = \{(c, \psi), (c, \psi')\} \cup R(c, \psi) \cup R(c, \psi')$
- $R(c, \langle\langle\Gamma\rangle\rangle X\psi) = \{(c', \psi) \mid T_\Gamma(c, a, c')$ *or* $T_{\overline{\Gamma}}(c, a, c')$, *for some joint action* $a \in ACT\} \cup \bigcup_{c'} R(c', \psi)$
- $R(c, K_i\psi) = \{(c', \psi) \mid c' \sim_i c\} \cup \bigcup_{c'} R(c', \psi)$
- $R(c, \langle\langle\Gamma\rangle\rangle G\psi) = \{(c, \psi), \langle\langle\Gamma\rangle\rangle X\psi\} \cup R(c, \psi) \cup R(c, \langle\langle\Gamma\rangle\rangle X\psi) \cup R(c, \langle\langle\Gamma\rangle\rangle X \langle\langle\Gamma\rangle\rangle G\psi)$
- $R(c, \langle\langle\Gamma\rangle\rangle(\psi U \psi')) = \{(c, \psi), (c, \psi')\} \cup R(c, \psi) \cup R(c, \psi') \cup R(c, \langle\langle\Gamma\rangle\rangle X\psi) \cup R(c, \langle\langle\Gamma\rangle\rangle X\psi') \cup R(c, \langle\langle\Gamma\rangle\rangle X \langle\langle\Gamma\rangle\rangle(\psi U \psi'))$
- $R(c, C_\Gamma\psi) = \{(c, \psi)\} \cup R(c, \psi) \cup \bigcup_{i\in\Gamma} R(c, K_i\psi) \cup R(c, \psi) \cup \bigcup_{i\in\Gamma} R(c, K_i C_\Gamma \psi)$

Observe that $R(c, \varphi)$ is well defined and can be computed by using standard fix-point algorithms, which are indeed validities in the proposed semantics. Specifically, the clauses for G-, U-, and C-formulas make use of the following fixed-point characterisations:

$$\begin{aligned}
\langle\langle\Gamma\rangle\rangle G\psi &\equiv \psi \wedge \langle\langle\Gamma\rangle\rangle X \langle\langle\Gamma\rangle\rangle G\psi \\
\langle\langle\Gamma\rangle\rangle(\psi U \psi') &\equiv \psi' \vee (\psi \wedge \langle\langle\Gamma\rangle\rangle X \langle\langle\Gamma\rangle\rangle(\psi U \psi')) \\
C_\Gamma\psi &\equiv \psi \wedge \bigwedge_{i\in\Gamma} K_i C_\Gamma \psi
\end{aligned}$$

Example 8 *Consider an abstract interpreted system IS with two agents $1, 2$, both having two states 0, 1 and two actions A, B, whose model is depicted on Figure 1.*

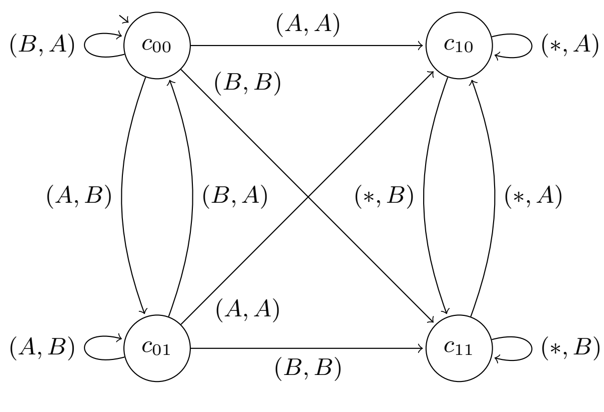

Figure 1. A model for Example 8. We assume that $T_\emptyset = T_{\{1\}} = T_{\{2\}} = T_{\{1,2\}}$ are all as depicted.

This may be interpreted as follows: both agents start in a local state 0. Agent 1 stays in the state 0 until both agents play the same actions; when this happens agent 1 moves to state 1 where it stays for the rest of the run. The second agent changes its state only on the basis of its action: if it performs action A it then moves to 0; if it performs action B, then it moves to state 1.

Assume that the labelling is such that the only state labelled with p is c_{11}. Consider the formula $\varphi = \langle\langle 1, 2\rangle\rangle Gp$.

By definition $R(c_{00}, \varphi)$ contains elements of $\{(c_{00}, \varphi)\}$, $R(c_{00}, p)$, $R(c_{00}, \langle\langle 1, 2\rangle\rangle X \langle\langle 1, 2\rangle\rangle Gp)$, and for all i, j, $R(c_{ij}, \langle\langle 1, 2\rangle\rangle Xp)$.

Then, $R(c_{00}, \langle\langle 1, 2\rangle\rangle X \langle\langle 1, 2\rangle\rangle Gp)$ contains $(c_{00}, \langle\langle 1, 2\rangle\rangle X \langle\langle 1, 2\rangle\rangle Gp)$ and all the pairs of $R(c_{ij}, \langle\langle 1, 2\rangle\rangle Gp)$, for all i, j. For every j, $R(c_{1j}, \langle\langle 1, 2\rangle\rangle Gp)$ contains $(c_{1j}, \langle\langle 1, 2\rangle\rangle Gp)$ and elements of $R(c_{10}, p)$, $R(c_{10}, \langle\langle 1, 2\rangle\rangle X \langle\langle 1, 2\rangle\rangle Gp)$ and for all j, $R(c_{1j}, \langle\langle 1, 2\rangle\rangle Xp)$.

The minimal function R satisfying the above conditions is as follows, for each $i, j \in \{0, 1\}$

$$\begin{aligned}
R(c_{ij}, p) =& \{(c_{ij}, p)\} \\
R(c_{1j}, \langle\langle 1, 2\rangle\rangle Xp) =& \{(c_{1j}, \langle\langle 1, 2\rangle\rangle Xp), (c_{10}, p), (c_{11}, p)\} \\
R(c_{0j}, \langle\langle 1, 2\rangle\rangle Xp) =& \{(c_{0j}, \langle\langle 1, 2\rangle\rangle Xp), (c_{10}, p), (c_{11}, p), \\
& (c_{00}, p), (c_{01}, p)\} \\
R(c_{1j}, \varphi) =& \{(c_{1k}, \varphi), (c_{1k}, \langle\langle 1, 2\rangle\rangle G\varphi), \\
& (c_{1k}, \langle\langle 1, 2\rangle\rangle Gp), (c_{1k}, p) \mid k \in \{0, 1\}\} \\
R(c_{0j}, \varphi) =& \{(c_{lk}, \varphi), (c_{lk}, \langle\langle 1, 2\rangle\rangle G\varphi), \\
& (c_{lk}, \langle\langle 1, 2\rangle\rangle Gp), (c_{lk}, p) \mid l, k \in \{0, 1\}\}
\end{aligned}$$

The significance of relevant pairs is given by the following immediate lemma.

Lemma 9 *If the truth value of φ in c is defined, then truth values for all relevant pairs in $R(c, \varphi)$ are also defined.*

Notice that because of loops and the expansions above, for a G-, U-, or a C-formula φ and state c it might be that (c, φ) belongs to $R(c, \varphi)$. Moreover, we show below that the cases for these formulas can be reduced to those for X- and K-formulas. As a consequence, we will be able to focus on refining single steps in a temporal or epistemic transition in the model.

Next, we introduce a notion of failure pair, inspired by [3]. Intuitively, for an abstract state c and formula φ, (c, φ) is a failure pair iff φ is undefined at c, albeit IS M has definite values for all relevant pairs for (c, φ) different from (c, φ) itself.

Definition 10 (Failure Pair) *A tuple (c, φ) is a* failure pair *iff $((M, c) \models^3 \varphi) = \mathrm{uu}$ and for all relevant pairs $(c', \psi) \in R(c, \varphi)$, where ψ is a strict subformula of φ, we have that $((M, c') \models^3 \psi) \in \{\mathrm{tt}, \mathrm{ff}\}$.*

A failure pair (c, φ) singles out a formula φ whose undefined value in state c is due to the structural features of the abstract IS itself. Hence, to provide a defined value to φ we have to refine the abstraction by using the information in (c, φ).

Clearly, propositional formulas are defined whenever all relevant pairs are. Hence, failure pairs can only be determined by atomic propositions and ATL and epistemic operators, as detailed in the following lemma.

Lemma 11 *A tuple (c, φ) is a failure pair iff $((M, c) \models^3 \varphi) = \mathrm{uu}$ and one of the following cases hold*

- $\varphi = p \in \mathcal{V}$

- *$\varphi = \langle\langle\Gamma\rangle\rangle X\psi$ and for all states c', if $T^A_\Gamma(c,a,c')$ or $T^A_{\overline{\Gamma}}(c,a,c')$ for some joint action a, then $((M,c') \models^3 \psi) \in \{\mathrm{tt}, \mathrm{ff}\}$*
- *$\varphi = K_i\psi$ and $((M,c') \models^3 \psi) \in \{\mathrm{tt}, \mathrm{ff}\}$ for all states $c' \sim_i c$*
- *φ is a G- or U-formula, in which case there is also a failure pair (c', ψ), where c' is reachable from c and ψ is a X-formula.*
- *φ is a C-formula, in which case there is also a failure pair (c', ψ), where c' is (epistemically) reachable from c and ψ is a K-formula.*

Proof sketch. The first part of the lemma follows from Def. 7 and 10. As an example, by contraposition suppose that $\varphi = \langle\langle\Gamma\rangle\rangle X\psi$, $((M,c) \models^3 \varphi) = \mathrm{uu}$, but for some state c', $T^A_\Gamma(c,a,c')$ or $T^A_{\overline{\Gamma}}(c,a,c')$ for some joint action a, and $((M,c') \models^3 \psi) = \mathrm{uu}$. In particular, (c', ψ) is a relevant pair for (c, φ). Hence, we derive that (c, φ) is not a failure pair. The result follows by contraposition.

For the second part, we show that failure pairs for G-formulas can be reduced to the case for X-formulas. Hence, suppose that

(i) $((M,c) \models^3 \langle\langle\Gamma\rangle\rangle G\varphi) = \mathrm{uu}$

(ii) $((M,c) \models^3 \varphi) \in \{\mathrm{tt}, \mathrm{ff}\}$

(iii) $((M,c') \models^3 \langle\langle\Gamma\rangle\rangle X\varphi) \in \{\mathrm{tt}, \mathrm{ff}\}$ for every reachable c' and for $c' = c$.

From (ii) and (iii) it follows that the truth value of ψ is defined in c and in all of its successors. So, $\langle\langle\Gamma\rangle\rangle G\psi$ is indeed defined in c against (i), which is a contradiction.

The cases for the U- and C-formulas are similar. □

As a consequence of Lemma 11, we can focus the search for failure pairs and the refinement procedure on X- and K-formulas only. The following result follows from Lemma 11, where a_Γ (resp. $a_{\overline{\Gamma}}$) are vectors of actions for the agents in Γ (resp. $\overline{\Gamma}$).

Lemma 12 *(i) If $(c, \langle\langle\Gamma\rangle\rangle X\psi)$ is a failure pair, then for every $a_\Gamma \in P^{must}_\Gamma(c)$, for some $a_{\overline{\Gamma}} \in P^{may}_{\overline{\Gamma}}(c)$, $T_A(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ implies $((M,c') \models^3 \psi) = \mathrm{ff}$, and for every $a_{\overline{\Gamma}} \in P^{must}_{\overline{\Gamma}}(c)$, for some $a_\Gamma \in P^{may}_\Gamma(c)$, $T_{\overline{\Gamma}}(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ implies $((M,c') \models^3 \psi) = \mathrm{tt}$.*
(ii) If $(c, K_i\psi)$ is a failure pair, then for some $c' \neq c$, $c'_i = c_i$ and $((M,c') \models^3 \psi) = \mathrm{ff}$.

Proof sketch. To derive a contradiction, suppose that $(c, \langle\langle\Gamma\rangle\rangle X\psi)$ is a failure pair, but some $a_\Gamma \in P^{must}_\Gamma(c)$ is such that for every $a_{\overline{\Gamma}} \in P^{may}_{\overline{\Gamma}}(c)$, $T_A(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ implies $((M,c') \models^3 \psi) \neq \mathrm{ff}$. Since $(c, \langle\langle\Gamma\rangle\rangle X\psi)$ is a failure pair, by Lemma 11 the truth value of ψ at c' has to be defined, and therefore $((M,c') \models^3 \psi) = \mathrm{tt}$. But then, by the semantics in Def. 3 we obtain that $((M,c) \models^3 \langle\langle\Gamma\rangle\rangle X\psi) = \mathrm{tt}$, against the hypothesis that $(c, \langle\langle\Gamma\rangle\rangle X\psi)$ is a failure pair. □

By building on the preliminaries results illustrated above, we now introduce the algorithm $FRFP$ to find relevant failure pairs. The procedure, presented as Algorithm 1, takes as input a finite abstract IS M, a state c and a formula φ such that $((M,c) \models^3 \varphi) = \mathrm{uu}$. As we show hereafter, the algorithm returns a relevant failure pair for (c, φ).

Lemma 13 *The algorithm $FRFP$ terminates provided that the interpreted system is finite. Moreover, the algorithm is sound, that is, if $FRFP(c, \varphi)$ returns (c', φ'), then $(c', \varphi') \in R(c, \varphi)$ is a failure pair relevant for (c, φ).*

Proof sketch. Since abstract IS are finite, the algorithm terminates in the case of X-, or K-formulas. Termination in the other cases follows from the fact that φ is finite. The output of the algorithm are failure pairs in view of Lemma 11. For instance, consider $\varphi = \langle\langle\Gamma\rangle\rangle G\psi$. If $((M,c) \models^3 \varphi) = \mathrm{uu}$, then by Lemma 11 either $((M,c) \models^3 \psi) = \mathrm{uu}$ or for some successor c' or $c' = c$, $((M,c) \models^3 \langle\langle\Gamma\rangle\rangle X\psi) = \mathrm{uu}$. In the former case, $FRFP(c, \varphi) = FRFP(c, \psi)$ and the algorithm is sound by the inductive step. In the latter case, $FRFP(c, \varphi) = FRFP(c', \langle\langle\Gamma\rangle\rangle X\psi)$, and once again

Algorithm 1 The algorithm $FRFP$.
INPUT: Model M; state c; formula φ s.t. $((M,c) \models^3 \varphi) = \mathrm{uu}$.
OUTPUT: (c', φ') s.t. $(c', \varphi') \in R(c, \varphi)$.

procedure $FRFP(c, \varphi)$
  if $\varphi = p \in \mathcal{V}$ **then**
    return (c, p)
  else if $\varphi = \neg\varphi'$ **then**
    return $FRFP(c, \varphi')$
  else if $\varphi = \varphi_1 \vee \varphi_2$ **then**
    let i be the minimum s.t. $((M,c) \models^3 \varphi_i) = \mathrm{uu}$;
    return $FRFP(c, \varphi_i)$
  else if $\varphi = \langle\langle\Gamma\rangle\rangle X\varphi'$ **then**
    if for all c', $T_\Gamma(c,a,c')$ or $T_{\overline{\Gamma}}(c,a,c')$ for some joint action $a \in ACT$ implies $((M,c') \models^3 \varphi') \in \{\mathrm{tt}, \mathrm{ff}\}$ **then**
      return (c, φ)
    else
      let c' be a successor of c s.t. $((M,c') \models^3 \varphi') = \mathrm{uu}$;
      return $FRFP(c', \varphi')$
    end if
  else if $\varphi = K_i\varphi'$ **then**
    if for every $c' \sim_i c$, $((M,c') \models^3 \varphi') \in \{\mathrm{tt}, \mathrm{ff}\}$ **then**
      return (c, φ)
    else
      let c' be s.t. $c' \sim_i c$ and $((M,c') \models^3 \varphi') = \mathrm{uu}$;
      return $FRFP(c', \varphi')$
    end if
  else if $\varphi = \langle\langle\Gamma\rangle\rangle G\varphi'$ **then**
    if $((M,c) \models^3 \varphi') = \mathrm{uu}$ **then**
      return $FRFP(c, \varphi')$;
    else
      let c' be c or a successor of c s.t. $((M,c') \models^3 \langle\langle\Gamma\rangle\rangle X\varphi') = \mathrm{uu}$
      return $FRFP(c', \langle\langle\Gamma\rangle\rangle X\varphi')$.
    end if
  else if $\varphi = \langle\langle\Gamma\rangle\rangle(\varphi_1 U \varphi_2)$ **then**
    if $((M,c) \models^3 \varphi_2) = \mathrm{uu}$ **then**
      return $FRFP(c, \varphi_2)$;
    else if $((M,c) \models^3 \varphi_1) = \mathrm{uu}$ **then**
      return $FRFP(c, \varphi_1)$;
    else
      let c' be c or a successor of c s.t. $((M,c') \models^3 \varphi_1 \wedge \langle\langle\Gamma\rangle\rangle X\varphi_2) = \mathrm{uu}$
      return $FRFP(c', \varphi_1 \wedge \langle\langle\Gamma\rangle\rangle X\varphi_2)$.
    end if
  else if $\varphi = C_\Gamma\varphi'$ **then**
    if $((M,c) \models^3 \varphi') = \mathrm{uu}$ **then**
      return $FRFP(c, \varphi')$;
    else
      let $c' \sim_\Gamma c$ and $i \in Ag$ be s.t. $((M,c') \models^3 K_i\varphi') = \mathrm{uu}$
      return $FRFP(c', K_i\varphi')$.
    end if
  end if
end procedure

the result follows by the inductive hypothesis. The other cases are similar. □

As a consequence, Algorithm 1 together with Lemma 13, defines a procedure to identify failure pairs, which will be used in the next section to refine the list of predicates, and therefore the abstraction.

4 Refining Abstractions

In this section we introduce and analyse a methodology for refining an abstract model on which a specification is initially evaluated as undefined. The method is based on the iterative application of Algorithm 2 below, which takes as input the present list of predicates, upon which the abstraction is built, and a failure pair, and returns as output the revised predicate list upon which a further abstraction can be built.

A key aspect in the derivation of the updated list of predicates is the use of *Craig's interpolants* [28]. Recall that the Craig's interpolant of formulas A and B, whose conjunction $A \wedge B$ is unsatisfiable, is a formula I such that

- $A \rightarrow I$ is valid;
- $I \wedge B$ is unsatisfiable; and
- I contains only non-logical symbols appearing in both A and B.

Craig's interpolants have previously been proven effective in refining abstractions in the context of different semantics and less expressive specification languages [28, 29]. Intuitively, the refinement methodology can be summarised as follows. Assume that formulas A and B represent witnesses for the current and the successive state in the abstract model. If the transition from A to B is spurious, that is, the transition in the abstract model does not correspond to a transition in the concrete system, the conjunction $A \wedge B$ is unsatisfiable. The interpolant I for $A \wedge B$ typically gives useful evidence as regards the reasons of the transition's spuriousness and can usefully provide guidance to refine the model [6].

Hereafter we describe the interpolation procedure we use to generate new predicates. Since the specification of interpreted systems includes first-order features, namely, linear arithmetic over the integers, in the following we adapt the approach originally put forward in [29] by applying concepts from [5] to account for the particular setting.

To begin, recall from Lemma 11 that all failure pairs can be reduced to the cases of atomic, X-, or K-formulas. Therefore, we present the refinement procedure via Craig's interpolations for these three cases in Algorithm 2, and discuss its rationale in the following. Algorithm 2 takes as input a failure pair (c, φ) and a tuple $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$ consisting of vectors of predicates and it returns an updated tuple $(\vec{p}'_1, \ldots, \vec{p}'_{|Ag|})$ of vectors of (possibly new) predicates. We assume Algorithm 2 operates on the abstract model under analysis and that φ is either an X- or a K-formula. We will address the atomic case later in the section.

X-Formulas (lines 2-7). The procedure takes as input the failure pair $(c, \langle\langle\Gamma\rangle\rangle X\varphi)$, as provided by Algorithm 1. By Lemma 12, we have that (i) for every $a_\Gamma \in P^{must}_\Gamma(c)$, for some $a_{\overline{\Gamma}} \in P^{may}_{\overline{\Gamma}}(c)$, $T_\Gamma(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ implies $((M, c') \models^3 \varphi) = \mathrm{ff}$, and (ii) for every $a'_{\overline{\Gamma}} \in P^{must}_{\overline{\Gamma}}(c)$, for some $a'_\Gamma \in P^{may}_\Gamma(c)$, $T_{\overline{\Gamma}}(c, a'_\Gamma \cdot a'_{\overline{\Gamma}}, c'')$ implies $((M, c'') \models^3 \varphi) = \mathrm{tt}$, which corresponds to line 3 in Algorithm 2.

In both cases we check whether the abstract transitions $T_\Gamma(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ and $T_{\overline{\Gamma}}(c, a'_\Gamma \cdot a'_{\overline{\Gamma}}, c'')$ correspond to actual transitions in the concrete IS; that is, whether there exists concrete states $s, s', s'' \in S$ such that $s \models c$, $s' \models c'$, $s'' \models c''$, and both $T(s, a_\Gamma \cdot a_{\overline{\Gamma}}, s')$ and $T(s, a'_\Gamma \cdot a'_{\overline{\Gamma}}, s'')$. This check is performed modularly, on the various agents $i \in Ag$. We comment on the case for $T_\Gamma(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$; the other case is similar.

Algorithm 2 The algorithm Refine.
INPUT: Failure pair (c, φ); $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$.
OUTPUT: $(\vec{p}'_1, \ldots, \vec{p}'_{|Ag|})$.

1: **procedure** Refine((c, φ),$(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$)
2: **if** $\varphi = \langle\langle\Gamma\rangle\rangle X\varphi'$ **then**
3: **let** c' be s.t. $T_\Gamma(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ and $((M, c') \models^3 \varphi') = \mathrm{ff}$, or $T_{\overline{\Gamma}}(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ and $((M, c') \models^3 \varphi') = \mathrm{tt}$;
4: **if** there is $i \in Ag$ and $l, l' \in L_i$ s.t. $l \models c.i$, $l' \models c'.i$ and $t_i(l, a_\Gamma \cdot a_{\overline{\Gamma}}, l')$ does not hold **then**
5: **let** I be an interpolant for $(l \wedge a_\Gamma \cdot a_{\overline{\Gamma}}) \wedge l'$
6: **return** $(\vec{p}_1, \ldots, \vec{p}_i \cdot I, \ldots, \vec{p}_{|Ag|})$
7: **else return** $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$
8: **else if** $\varphi = K_i\varphi'$ **then**
9: **let** c' be s.t. $c' \neq c$, $c'.i = c.i$ and $((M, c') \models^3 \varphi) = \mathrm{ff}$
10: **if** there are $l, l' \in L_i$ s.t. $l \models c.i$, $l' \models c'.i$ and $l \neq l'$
11: **let** I be an interpolant for $l \wedge l'$
12: **return** $(\vec{p}_1, \ldots, \vec{p}_i \cdot I, \ldots, \vec{p}_{|Ag|})$
13: **else return** $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$
14: **else return** $(\vec{p}_1, \ldots, \vec{p}_{|Ag|})$
15: **end if**
16: **end procedure**

By definition of the predicate abstraction, $T_\Gamma(c, a_\Gamma \cdot a_{\overline{\Gamma}}, c')$ holds iff $t^{must}_i(c.i, a_\Gamma \cdot a_{\overline{\Gamma}}, c'.i)$ for $i \in A$ and $t^{may}_i(c.i, a_\Gamma \cdot a_{\overline{\Gamma}}, c'.i)$ for $i \notin A$. Now consider an agent $i \in Ag$ and witnesses $l, l' \in L_i$, that is, $l \models c.i$ and $l' \models c'.i$. Depending on $i \in Ag$, we refine either $t^{must}_i(c.i, a_\Gamma \cdot a_{\overline{\Gamma}}, c'.i)$ or $t^{may}_i(c.i, a_\Gamma \cdot a_{\overline{\Gamma}}, c'.i)$. If $t_i(l, a_\Gamma \cdot a_{\overline{\Gamma}}, l')$ holds, then l and l' witness indeed the abstract transition (line 7 in Algorithm 2). Otherwise, we consider the conjunction $\theta = l \wedge a_\Gamma \cdot a_{\overline{\Gamma}} \wedge l'$, where the atoms and variables in l' are primed, while actions are interpreted as equalities between variables and their primed versions (line 4 in Algorithm 2). If $t_i(l, a_\Gamma \cdot a_{\overline{\Gamma}}, l')$ does not hold, then θ is unsatisfiable, and we can make use of interpolation to refine the abstract transition. To do this, we need to find two new abstract states $d.i$ and $d'.i$ such that $l \models d.i$, $l' \models d'.i$, and either $t'^{may}_i(d.i, a_\Gamma \cdot a_{\overline{\Gamma}}, d'.i)$ or $t'^{must}_i(d.i, a_\Gamma \cdot a_{\overline{\Gamma}}, d'.i)$ does not hold (depending on whether $i \in Ag$ or $i \notin Ag$), where t'^{may}_i and t'^{must}_i are intuitively the new, refined transitions.

As a result, the new transition t'^{may}_i is "finer" than t^{may}_i, or t'^{must}_i is "coarser" than t^{must}_i; that is, t'^{may}_i relates *fewer* concrete local states than t^{may}_i, while t'^{must}_i relates *more* concrete local states than t^{must}_i. More specifically, by interpolation we obtain an interpolant I such that $l \wedge a_\Gamma \cdot a_{\overline{\Gamma}} \rightarrow I$ is valid and $l' \wedge I$ is unsatisfiable (line 5 in Algorithm 2).

We now shoe how I can be used as a predicate to eliminate the spurious transition from l to l'. In particular, I is built on non-logical symbols appearing in both l and l'. Hence, it is local to agent i and can be introduced as a new predicate. The updated list $(\vec{p}_1, \ldots, \vec{p}'_i, \ldots, \vec{p}_{|Ag|})$ of predicates, where $\vec{p}'_i = \vec{p}_i \cdot I$ (line 6 in Algorithm 2) is then returned and used to construct a further abstracted model M'^A. Notice that M'^A does not contain the state $c.i$, but it includes at least one of the new states $c.i \wedge I$ and $c.i \wedge \neg I$. we now have that either $l \models c.i \wedge I$ or $l \models c.i \wedge \neg I$. Then, let $d.i$ be the state satisfied by l. Now, if $l \models d.i$ and $t_i(l, a_\Gamma \cdot a_{\overline{\Gamma}}, l'')$ for some local state $l'' \in L_i$ such that $l'' \models c'.i$, then $l'' \models c'.i \wedge I = d''.i$ as well, as

$l \wedge a_\Gamma \cdot a_{\overline{\Gamma}} \to I$ is a validity. On the other hand, $l' \models c'.i \wedge \neg I = d'.i$, and therefore the successors l'' and l' of l belong to different abstract states $d''.i$ and $d'.i$. As a result, the transition $t_i'^{may}$ in M'^A is finer than t_i^{may}, or $t_i'^{must}$ is coarser than t_i^{must} (depending on whether $i \in Ag$ or $i \notin Ag$.) This terminates the procedure for X-formulas.

K-formulas (lines 8-13). The case of K-formulas is similar to the previous one. By Lemma 12, for some $c' \neq c$, $c'.i = c.i$ and $((M, c') \models^3 \varphi) = \mathrm{ff}$ (line 9 in Algorithm 2). Now consider witnesses $l, l' \in L_i$ such that $l \models c.i$, $l' \models c'.i$, but $l \neq l'$, if any (line 10 in Algorithm 2). This means that l and l' are spurious witnesses for the i-indistinguishability of abstract states c and c'. It follows that the conjunction $l \wedge l'$ is unsatisfiable, as l and l' are two different assignments of values to the variables and propositional atoms of agent i. Hence, we can find a Craig's interpolant I such that $l \to I$ is valid and $l' \wedge I$ is unsatisfiable (line 11 in Algorithm 2). As in the case of X-formulas, we use I as a predicate to refine the spurious indistinguishability relation. Specifically, I can be used as a new predicate, as it is built on non-logical symbols appearing in both l and l'. The revised list $(\vec{p}_1, \ldots, \vec{p}_i', \ldots, \vec{p}_{|Ag|})$ of predicates can now be returned (line 12 in Algorithm 2), where $\vec{p}_i' = \vec{p}_i \cdot I$; this can be used to generate a refinement M'^A. On the refined model M'^A we will obtain $l \models c.i \wedge I = d.i$, while $l' \models c.i \wedge \neg I = d'.i$. Therefore, the states $d.i$ and $d'.i$, refining the states $c.i$ and $c'.i$ respectively, are different and therefore not i-indistinguishable. By considering all $c' \neq c$ such that $c'.i = c.i$, $((M, c') \models^3 \theta) = \mathrm{ff}$, and $l \models c.i$, $l' \models c'.i$ for some $l \neq l'$, we can eventually decide the truth value of failure formula $K_i\varphi$. This terminates the procedure for K-formulas.

Since Algorithm 2 returns an updated list of predicates, the abstraction M'^A built on it is also an abstraction of the concrete IS M. Hence, Theorem 6 applies, and therefore formulas defined in M'^A are preserved in M. Moreover, the initial abstraction M^A can be thought of as an abstraction of M'^A as well, where state c' abstracts state c iff $c'.i \models c.i$, for every $i \in Ag$. We summarise these remarks in the next immediate result.

Theorem 14 *Given an abstraction M^A of an IS M, the refinement M'^A is also an abstraction of M and it is abstracted by M^A. Thus, the refinement procedure defines a sequence $M, \ldots, M'^A, M^A$ of IS such that any element in the sequence is an abstraction of its predecessors.*

Algorithm 2 does not address the case of atomic propositions that were also identified in Lemma 11 as possible components of failure pairs. Observe that if (c, p) is indeed a failure pair, for p atomic, then p refers to more than one agent. This follows immediately from the fact that the list of predicates for an agent i contains all atoms referring to i itself. Hence, the truth value of such atoms is always immediately defined in the abstraction. As a consequence, an agent-based refinement procedure cannot be given for atomic predicates, as their satisfaction cannot be established by evaluating local states only. This limitation does not appear to be significant as in most cases of interest we expect to be able to resolve the value of the specification of interest by refining the temporal and epistemic transitions. Observe that any abstraction procedure is in any case incomplete as the verification problem is undecidable in general.

Furthermore, note that the complexity of the refinement procedure is determined by lines 6, 7, 12, and 13 in Algorithm 2. These return the states satisfying a given constraint or interpolants. Both these problems can be reduced to solving linear inequalities; therefore the complexity of the procedure is bounded by the complexity of linear programming.

We conclude by remarking that, since M can be an infinite-state IS and the refinement procedure adds finitely-many states only, the sequence $M, \ldots, M'^A, M^A$ in Theorem 14 is infinite in principle. Hence, as in other predicate abstraction approaches, the refinement procedure is not guaranteed to terminate. Nonetheless, in many cases of interest the methodology can resolve the truth value of specifications that cannot be evaluated on the initial abstraction. We consider one such case in the following section.

5 Verification of an Infinite-state English Auction Protocol

We now illustrate the methodology presented in the paper on an example of an ascending English auction [10]. Consider an auction with an auctioneer A and a finite number of bidders $B_1, \ldots, B_n$. Each bidder B_i has a fixed amount of resources (e.g., money) $m_i > 2$; they follow a protocol of the form "if the latest bid was less than m_i, then non-deterministically decide to bid or not; otherwise do nothing". We assume that the protocols that the agents run are commonly known. All the bidders and the auctioneer have an integer value to store the latest highest bid. We assume that bids start from 0 and each bid increases the previous bid by 1.

We would like to establish whether it is a common knowledge that the auction terminates and whether the bidders have a strategy to buy the item for 1 resources in two rounds.

We formalise this auction as an infinite-state interpreted system M. Each bidder B_i has a variable lb of type integer representing the latest bid, initially set to 0, and two actions: bid and $skip$. The protocol of B_i is such that: $P_i(lb) = \{bid, skip\}$ when $lb \leq m_i$; $P_i(lb) = \{skip\}$ when $lb > m_i$. We can further encode that the transition function t_i is such that lb remains unchanged if bidder B_i uses the action $skip$; lb is increased by 1 otherwise.

We model the auctioneer A by considering an integer variable lb initially set to 0, a variable $top\_bidder \in \{0, \ldots, n\}$ initially set to 0 and a boolean variable $sold$, initially set to false, as well as the actions $skip$ and $sold_i$, where $i \in \{0, \ldots, n\}$. The transition function for A is such that when all the bidders perform the action skip, the variable $sold$ is set to true. At that time the auction is over and the winner is announced; from that point onwards A loops in the same state announcing winner i using action $sold_i$. If any of the bidders uses the action bid, then the top_bidder is chosen non-deterministically among all the bidders who bidded. The variable lb is updated as in the case of bidders. We assume a labelling function with atoms $A.sold$ and $A.lb = 1$ depending on A's local variables.

We investigate the properties specified above by evaluating the formulas

$$\varphi = C_{\{B_1, \ldots, B_n\}} \langle\langle \emptyset \rangle\rangle F(A.sold)$$

$$\rho = \langle\langle \{B_1, \ldots, B_n\} \rangle\rangle X \langle\langle \{B_1, \ldots, B_n\} \rangle\rangle X(A.sold \wedge A.lb = 1)$$

on the infinite-state interpreted system described above.

By conducting the initial abstraction as in [22] we obtain a model M^A based on the predicates $sold$, $top\_bidder = 0$, and $lb = 0$ for A (from the definition of the initial state) and the predicates $lb = 0, lb < m_i$ for agent B_i (the first from B_i's initial state, the second from B_i's protocol).

The may and must transition relations for auctioneer A in M^A coincide. Specifically, from the initial state denoted by $\neg sold \wedge (top\_bidder = 0) \wedge (lb = 0)$, if all bidders perform skip, then the next state is $\neg sold \wedge (top\_bidder = 0) \wedge (lb = 0)$; otherwise it becomes $\neg sold \wedge (top\_bidder \neq 0) \wedge (lb \neq 0)$. From the latter state, A loops whenever at least one bidder bids, or moves to the state $sold \wedge (top\_bidder \neq 0) \wedge (lb \neq 0)$ otherwise, where it can only loop.

The initial state for agent B_i is $(lb = 0) \wedge (lb < m_i)$, where B_i stays if all bidders skip, or moves to $(lb \neq 0) \wedge (lb < m_i)$ otherwise (recall that $m_i > 2$.) These are both may and must transitions. From the latter state, B_i has no must transition, but has two may transitions: a loop and a transition to $(lb \neq 0) \wedge (lb \not< m_i)$ over any action where one of the agents bids. In $(\neg lb = 0) \wedge (lb \not< m_i)$, B_i loops for both the may and must transitions.

It can be checked that the initial abstraction built on these predicates is such that ρ is evaluated to true; it follows that ρ is satisfied in the infinite state system. However, φ is undefined in the initial abstraction. We now show how the refinement procedure introduced in this paper enables us to determine the truth value of φ.

By using algorithm $FRFP$ from Section 3, we obtain the failure pair $(\neg sold \wedge (top\_bidder \neq 0) \wedge (lb \neq 0), (c_1, \ldots, c_n)), \langle\langle\emptyset\rangle\rangle X(A.sold))$, where for each i, $c_i = (lb \neq 0) \wedge (lb < m_i)$. This enables us to apply the refinement procedure given in Algorithm 2 for the case of X-formulas.

We illustrate this by considering bidder 1 with $m_1 = 10$. Consider a transition from $c = (lb \neq 0) \wedge (lb < 10)$ to $c' = (lb \neq 0) \wedge (lb \not< 10)$, which is the result of abstracting the transition encoded by the condition $lb' = lb + 1$. Let l be the state $lb = 1$, and l' be the state $lb = 10$. Clearly, $l \models c$ and $l' \models c'$. Therefore, we find an interpolant for $l \wedge lb' = lb + 1$ and l'. We can derive a refutation as shown in Figure 2 (see [29]).

$$\dfrac{\dfrac{\dfrac{lb = 1}{0 \leq -lb + 1}\,\text{LE} \qquad \dfrac{0 = -lb' + lb + 1}{0 \leq -lb' + lb + 1}\,\text{LE}}{0 \leq -lb' + 2} \qquad \dfrac{0 = lb' - 10}{0 \leq lb' - 10}\,\text{LE}}{0 \leq -8}\,\text{C}$$

Figure 2. Rule LE allows to derive disequalities from equalities, while C returns $0 \leq t + t'$ from the premises $0 \leq t$ and $0 \leq t'$.

From the refutation of Figure 2 we can obtain an interpolant [5], simply by setting all disequalities in branches for ψ' to $0 \leq 0$, as shown in Figure 3.

$$\dfrac{\dfrac{\dfrac{lb = 1}{0 \leq -lb + 1}\,\text{LE} \qquad \dfrac{0 = -lb' + lb + 1}{0 \leq -lb' + lb + 1}\,\text{LE}}{0 \leq -lb' + 2} \qquad \dfrac{0 = 0}{0 \leq 0}\,\text{LE}}{0 \leq -lb' + 2}\,\text{C}$$

Figure 3. Obtaining an interpolant from a refutation.

By doing so, we obtain the formula $lb' \leq 2$, which is indeed an interpolant for the pair (ψ, ψ'), and can be used in the refinement procedure. Therefore, the revised list of predicates for B_i is $[lb = 0, lb \leq 2, lb < m_i]$.

This refinement step results in an abstraction that is still insufficient to decide the value of φ. However, by conducting a number of further refinement steps for B_1 bounded by m_1, we may derive the list of predicates that fully characterise all the possible values of lb below m_1. When this is done for all the bidders, the abstract interpreted system is such that the states of bidder B_i correspond to numbers $0, \ldots, m_i$. On such a system, it can be checked that the property φ holds, and therefore it is satisfied in the original infinite-state system.

6 Conclusions

Little attention has so far been devoted to the practical verification of infinite-state MAS. A key requirement of any predicate abstraction technique is not only the initial generation of the predicates, but also their refinement to produce a sequence of abstractions approximating the concrete system. As we discussed in Section 1, present approaches for MAS against ATL specifications fall short in this respect.

In this paper we have put forward a refinement methodology for MAS abstractions to be verified against ATLK specifications. The proposed approach uses state-of-the-art automatic deduction techniques based on interpolants via SMT calls, as pioneered by [29] in the context of purely temporal logic. We showed that the method is sound and illustrated its potential on an infinite state MAS implementation of a simple auction protocol.

A noteworthy feature of the approach lies in the choice and development of both the semantics and the specification language, which are both oriented towards MAS. In terms of semantics we use and extended interpreted systems, that have long been used as a formal model to reason about MAS. In particular, as discussed in the Introduction and differently from [3], we here adopt incomplete information and memoryless strategies. In terms of specifications we follow our previous work in this line [20, 21, 22] by combining strategic concepts given in a weaker form of ATL with an epistemic language. The branching-time temporal-epistemic logic CTLK is entirely subsumed in the approach should this be found to be preferable in some applications.

A further aspect of the work concerns its potential applicability. The choice of adopting non-uniform strategies keeps the decision problem against explicit models in PTIME, which is important in practical verification. It is well known that this comes at the cost of expressivity and it is reflected in the reading of the ATL modalities.

In future work, we intend to implement the technique and algorithms introduced here. We anticipate this will be challenging given the complexity of devising efficient heuristics resolving the non-determinism of some of the refinement steps here described. Since the abstraction methodology is necessarily incomplete, this will also require a considerable amount of tuning of the heuristics against several benchmarks, so that any resulting tool offers the concrete possibility of solving actual MAS programs.

Acknowledgements

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

REFERENCES

[1] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge, 'Knowledge and ability', in *Handbook of Logics for Knowledge and Belief*, College Publications, (2015).

[2] R. Alur, T. A. Henzinger, and O. Kupferman, 'Alternating-time temporal logic', *Journal of the ACM*, **49**(5), 672–713, (2002).

[3] T. Ball and O. Kupferman, 'An abstraction-refinement framework for multi-agent systems', in *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS06)*, pp. 379–388. IEEE, (2006).

[4] N. Bulling, J. Dix, and W. Jamroga, 'Model checking logics of strategic ability: Complexity', in *Specification and Verification of Multi-agent Systems*, 125–159, Springer, (2010).

[5] A. Cimatti, A. Griggio, and R. Sebastiani, 'Efficient generation of craig interpolants in satisfiability modulo theories', *ACM Transactions of Computational Logic*, **12**(1), 7:1–7:54, (2010).

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

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

[8] 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)*, pp. 721–726, (2009).

[9] S. Das, D. Dill, and S. Park, 'Experience with predicate abstraction', in *Proceedings of the 11th International Conference on Computer Aided Verification (CAV99)*, pp. 160–171. Springer, (1999).

[10] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, New York, NY, USA, 2010.

[11] E. A. Emerson and E. M. Clarke, 'Using branching-time temporal logic to synthesize synchronization skeletons', *Science of Computer Programming*, **2**(3), 241–266, (1982).

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

[13] 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 *Lecture Notes in Computer Science*, pp. 479–483. Springer, (2004).

[14] P. Gonzalez, A. Griesmayer, and A. Lomuscio, 'Verification of GSM-based artifact-centric systems by predicate abstraction', in *Proceedings of the 13th International Conference on Service Oriented Computing (ICSOC15)*, volume 9435 of *Lecture Notes in Computer Science*, pp. 253–268. Springer, (2015).

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

[16] W. Jamroga and J. Dix, 'Model checking abilities under incomplete information is indeed δ_p^2-complete', in *Proceedings of the 4th European Workshop on Multi-Agent Systems (EUMAS'06)*, pp. 14–15, (2006).

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

[18] G. Jonker, *Feasible Strategies in Alternating-time Temporal Epistemic Logic*, Master's thesis, University of Utrech, The Netherlands, 2003.

[19] S. Lahiri, R. Nieuwenhuis, and A. Oliveras, 'Smt techniques for fast predicate abstraction', in *Proceedings of the 18th International Conference on Computer Aided Verification (CAV06)*, pp. 424–437. Springer, (2006).

[20] 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)*, pp. 428–437. AAAI Press, (2014).

[21] A. Lomuscio and J. Michaliszyn, 'Verifying multi-agent systems by model checking three-valued abstractions', in *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS15)*, pp. 189–198, (2015).

[22] A. Lomuscio and J. Michaliszyn, 'Verification of multi-agent systems via predicate abstraction against ATLK specifications', in *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS16)*, (2016).

[23] A. Lomuscio, W. Penczek, and H. Qu, 'Partial order reduction for model checking interleaved multi-agent systems', *Fundamenta Informaticae*, **101**(1–2), 71–90, (2010).

[24] 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*, pp. 682–688. Springer, (2009).

[25] A. Lomuscio, H. Qu, and F. Raimondi, 'MCMAS: A model checker for the verification of multi-agent systems', *Software Tools for Technology Transfer*, (2015). http://dx.doi.org/10.1007/s10009-015-0378-x.

[26] A. Lomuscio and F. Raimondi, 'The complexity of model checking concurrent programs against CTLK specifications', in *DALT*, volume 4327 of *Lecture Notes in Computer Science*, pp. 29–42. Springer, (2006).

[27] 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)*, pp. 161–168. ACM Press, (2006).

[28] K. L. McMillan, 'Interpolation and sat-based model checking', in *Proceedings of the 15th International Conference on Computer Aided Verification (CAV03)*, pp. 1–13, (2003).

[29] K. L. McMillan, 'An interpolating theorem prover', *Theoretical Computer Science*, **345**(1), 101–121, (2005).

[30] J.-J. Ch. Meyer and W. van der Hoek, *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 1995.

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

[32] S. Shoham and O. Grumberg, 'Monotonic abstraction-refinement for CTL', in *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS04)*, volume 2988 of *Lecture Notes in Computer Science*, pp. 546–560. Springer, (2004).