On Observability Analysis in Multiagent Systems

Chunyan Mu\textsuperscript{a,\ast} and Jun Pang\textsuperscript{b}

\textsuperscript{a}Department of Computing Science, University of Aberdeen
\textsuperscript{b}Department of Computer Science, University of Luxembourg

Abstract. In multiagent systems (MASs), agents’ observation upon system behaviours may improve the overall team performance, but may also leak sensitive information to an observer. A quantified observability analysis can thus be useful to assist decision-making in MASs by operators seeking to optimise the relationship between performance effectiveness and information exposure through observations in practice. This paper presents a novel approach to quantitatively analysing the observability properties in MASs. The concept of opacity is applied to formally express the characterisation of observability in MASs modelled as partially observable multiagent systems. We propose a temporal logic \textit{\textsc{opatl}} to reason about agents’ observability with quantitative goals, which capture the probability of information transparency of system behaviours to an observer, and develop verification techniques for quantitatively analysing such properties. We implement the approach as an extension of the PRISM model checker, and illustrate its applicability via several examples.

1 Introduction

The multiagent computing paradigm pervades nearly all aspects of the modern intelligent computational world, enabling the creation of net-based solutions to communication, collaboration, and coordination problems in different fields such as commerce, cyber, and conflict prevention. Agents often exploit machine learning methods, which allow them to learn from experience, and to implement decision-making mechanisms. Observation of other agents’ behaviours may improve the overall team performance in the learning mechanisms [26]. On the other hand, in practice, due to the frequently adversarial nature of multiagent systems (MASs) such solutions can also bring additional channel threat and information leakage risks. Sensitive information can be leaked to malicious (inside/outside) agents during the process of collaboration and interaction. Information exposure issue should also play a role in making decisions for agents. Therefore, rigorous analysis and verification of (sensitive) information transparency properties constitutes an important challenge. In particular, a quantified observability analysis can be useful in MASs design to address such concerns, for instance, for decision-making by operators seeking to optimise the relationship between performance effectiveness and information exposure security risks in MASs, which are the key underpinning elements of a progressive artificially intelligent society.

This paper addresses the problem of specifying, verifying and thus reasoning about observability properties of MASs. Specifically, we specify the observability properties from novel perspective of information transparency in the \textit{opacity} framework, which is formally described in the logic \textit{\textsc{opatl}}. With this logic, we can express the degree of transparency of system behaviours to an observer under a coalition of agents’ strategy, given predefined observability of atomic actions to the observer. We model the system in partially observable probabilistic game structure, which maps infinite (input) sequences onto partially observable infinite (output) sequences. The properties of observability can then be captured by measurement upon output sequences and input sequences. Intuitively, a transparent system, in which the observability is maximised, reveals most information in the input sequence; while an opaque system, in which the observability is minimised, hides some information (with properties of interest) contained in the input sequence. Probabilistic model checking techniques can be applied to reason about the quantitative observability analysis of the system, and allow us to calculate the degree of the observability of the system behaviours.

The main contributions of the paper are summarised below:

\begin{itemize}
  \item A partially observable multi-agent system (POMAS) is proposed to model probabilistic action outcomes of system behaviours with characterisation of multi-agents, actions and the relevant observables, and atomic state propositions.
  \item The logic of \textit{\textsc{opatl}} is presented to allow us to express (probabilistic) observability properties.
  \item Probabilistic verification technique against \textit{\textsc{opatl}} is presented to allow for automatic verification of quantified observability properties in MASs modelled as POMAS.
  \item A prototype of the proposed framework is built upon the PRISM model checker [22].
\end{itemize}

Related work. In the field of formal methods for artificial intelligence, logics have gained a great importance in expressing properties and providing powerful formalisms for reasoning about agents behaviours in MASs. There have been several multiagent logics proposed to express and reason about agents’ observation properties including [19, 5, 12, 17]. These logics have centred on knowledge representation where knowledge is built from what the agents observe. In these logics, the formation of knowledge is modelled via epistemic connectives which can be defined as modal operators of the form $K_i \varphi$ specifying “agent $i$ knows property $\varphi$”, and the observability of agents is modelled via Kripkean accessibility relations with respect to the visibility atoms of propositional variables: agent $i$ cannot distinguish valuation $w$ from $w'$ if every variable that agent $i$ observes has the same value at $w$ and $w'$. A number of works [18, 17, 31] have studied multiagent planning model to adapt strategy for cooperation and analyse trade-off between local observation and capability of coordination based on estimation of quantified communicating and variant costs. Various methods and accompanying implementations

\* Corresponding Author. Email: chunyan.mu@abdn.ac.uk

Please check ArXiv or contact the authors for any appendices or supplementary material mentioned in the paper.
2 Partially Observable Multiagent Systems

Let \( \mathbb{N} \) be the set of natural numbers with zero, \( \text{Ag} = \{1, 2, \ldots, n\} \) be a set of agents. An alphabet \( \Sigma \) is a non-empty, finite set of actions, \( |\Sigma| \) is its cardinality. \( \Sigma^* \) denotes the set of all finite words over \( \Sigma \) including the empty word \( \varepsilon \), \( \Sigma^+ = \Sigma^* \setminus \{\varepsilon\} \), \( \Sigma^\omega \) denotes the set of all infinite words, \( \Sigma^\infty \) denotes the set of all finite and infinite words. Subsets \( L \subseteq \Sigma^* \) are called languages, and \( L \subseteq \Sigma^\infty \) are called \( \omega \)-languages. Let \( \text{Dist}(X) \) denote the set of discrete probability distribution over a set \( X \), i.e., all functions \( \mu : X \rightarrow [0, 1] \) s.t. \( \sum_{x \in X} \mu(x) = 1 \) and \( \mu(x) \geq 0 \). \( 2^X \) denotes the power set of \( X \).

2.1 Probabilistic (concurrent) game structure

**Definition 1.** A probabilistic game structure (PGS) is a tuple \( G = (S, \text{Act}, \delta) \), where:

- \( S \) is a finite set of states;
- \( \text{Act} = \text{Act}_1 \times \text{Act}_2 \times \ldots \times \text{Act}_n = \prod_{j \in \text{Ag}} \text{Act}_j \) is a finite set of joint actions (decisions) of the agents in \( \text{Ag} \), \( \text{Act}_j \subseteq \Sigma \) is the set of actions that \( j \in \text{Ag} \) can perform;
- \( \delta : S \rightarrow 2^{\text{Dist}(\text{Act} \times S)} \) is the probabilistic transition relation; for state \( s \in S \), \( \delta(s) \) is the distribution for next state; \( s \) is a terminal state if \( \delta(s) = \emptyset \).

We write \( s \xrightarrow{\nu} s' \) for \( s \subseteq S \) and \( \mu \in \delta(s) \). Each agent \( j \in \text{Ag} \) chooses action \( a_j \) in state \( s \in S \), we write \( s \xrightarrow{a_j} s' \) and sometimes \( s \xrightarrow{\pi_j \in \text{Act}_j} s' \) for \( s, s' \in S \) whenever \( s \xrightarrow{\nu} s' \) and \( \mu(s, \pi_j) > 0 \), where \( \pi \) denotes the probability of the transition from \( s \) to \( s' \) through joint action \( \prod_{j \in \text{Ag}} a_j \). We use non-deterministic PGS in this paper to accommodate the agents’ probabilistic behaviour. While the game structure determines the probability of each action, agents can still make decisions based on their probabilistic strategies or beliefs. To enable a broader range of strategies, an MDP-like transition function that maps a state-action pair to a distribution over the next state would provide greater flexibility for agent behaviour. This extension is a potential area for future work.

**Definition 2.** We say \( G \) is circular, if every state has an outgoing transition, i.e., for all \( s \in S \), there is \( s \xrightarrow{\nu} s' \). We say \( G \) is fully probabilistic if \( |\delta(s)| \leq 1 \) for all \( s \in S \). For a fully probabilistic game structure, when \( \delta(s) \neq \emptyset \), we use \( \delta(s) \) to denote the distribution outgoing from \( s \).

**Definition 3.** A path in \( G \) is a sequence \( \rho = s_0 \xrightarrow{a_{i_0}} s_1 \xrightarrow{a_{i_1}} \ldots \) of states and joint actions, where \( a_{i_1} = \prod_{j \in \text{Ag}} a_{i_j} \in \text{Act}, a_{i_j} \in \text{Act}_j(s_i) \) for \( i \geq 0 \) and \( j \in \text{Ag} \), for all \( t \geq 0 \), \( s_t \in S \), \( \alpha_t \in \text{Act} \) and \( \delta(s_t) \xrightarrow{a_{i_{t+1}}} \rho_{t+1} \geq 0 \). Let \( \rho_t(i) \) denote the \( i \)-th state of \( \rho \), and \( \rho_t(i) \) the \( i \)-th joint action of \( \rho \), so for all \( i \), we have \( \rho_t(i) = \rho_t(i+1) \). Let \( \rho' \) denote the prefix of \( \rho \) up to the \( i \)-th state, i.e., \( \rho' = s_0 \xrightarrow{a_{i_0}} s_1 \xrightarrow{a_{i_1}} \ldots \xrightarrow{a_{i-1}} s_i \). Let \( \text{Post}(\rho) \) denote immediate state successors of \( s \) in a path, and \( \text{Pre}(\rho) \) denote the immediate state predecessors of \( s \) in a path. A path is finite if it ends with a state. A path is complete if it is either infinite or finite ending in a terminal state. Given a finite path \( \rho \), last(\( \rho \)) denotes its last state. The length of a path \( \rho \), denoted by \( |\rho| \), is the number of transitions appearing in the path. Let \( \text{Paths}(s) \) denote the set of \( G \)-paths, \( \text{Paths}(s)^* \) denote the set of all \( G \)'s finite paths, \( \text{CPaths}(s) \) denote the set of all \( G \)'s complete paths, starting from state \( s \). Paths are ordered by the prefix relations, denoted by \( \leq \): \( \text{Pref}(\rho') = \{\rho \mid \rho \leq \rho'\} \).

**Definition 4.** The trace of a path is the sequence of joint actions in \( \text{Act}^* \cup \text{Act}^\omega \) obtained by erasing the states, so for the above \( \rho \), we have the corresponding trace of \( \rho \): \( \text{tr}(\rho) = a_{i_0} a_{i_1} \ldots \). We use \( \text{Traces}(s) \) to denote the set of \( G \)-traces starting from state \( s \).

Let \( G = (S, \text{Act}, \delta) \) be a PGS, \( \rho \in \text{Paths}(s)^* \) be a finite path starting from \( s \in S \). The cone generated by \( \rho \) is the set of complete
paths \( \langle \rho \rangle = \{ \rho' \in \text{CPaths}_s (s) \mid \rho \leq \rho' \} \). Given a \( G = (S, \text{Act}, \delta) \) and a state \( s \in S \), we can then calculate the probability value, denoted by \( P_s (\rho) \), of any finite path \( \rho \) starting at \( s \) as follows:

- \( P_s (s) = 1 \), and
- \( P_s (\rho \xrightarrow{a} s') = P_s (\rho) \mu (s', a) \) for \( \text{last}(\rho) \rightarrow a \).

Let \( \Omega_s = \text{CPaths}_s (s) \) be the sample space, and let \( G_s \) be the smallest \( \sigma \)-algebra induced by the cones generated by all the finite paths of \( G \). Then \( P_s \) induces a unique probabilistic measure on \( G_s \) such that \( P_s (\langle \rho \rangle) = P_s (\rho) \) for every finite path \( \rho \) starting in \( s \).

### 2.2 Observations

To model the observability of agents, we need to make a distinction between the actions that are observable and those that are not, regarding different agents’ view. For each agent, we use a set of observables, distinct of the actions of the ambient PGS. Actions and observables are connected by an observation function.

**Definition 5.** Let \( \Theta \) be a finite alphabet for observables, and \( \Theta^e = \text{obs} \cup \{ \varepsilon \} \) where \( \varepsilon \) denotes the invisible/hidden action. An observation function on paths is a labelled-based function \( \text{obs} : \text{Paths}_s (s) \rightarrow (\Theta_1 \times \Theta_2 \times \cdots \times \Theta_n)^\infty \), where \( \Theta_j \subseteq \Theta \) denotes a finite set of observables for \( j \in \text{Ag} \). Specifically, we consider static observation function, i.e., there is a map \( \zeta : \text{Act} \rightarrow \Theta \) s.t. for every path \( \rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_t \) of \( G; \text{obs}(\rho) = \beta_0 \beta_1 \cdots \beta_{t-1} \), where for all \( 0 \leq i < t, a_i = \prod_{j \in \text{Ag}} \zeta (a_i) \).

Observation functions on traces are defined similarly.

### 2.3 Partially observable MASs

**Definition 6.** A partially observable multiagent system (POMAS) is a tuple \( M = (\text{Ag}, G, s_0, \text{Ap}, L, \{ \text{obs} \}_i \in \text{Ag}) \), where:

- \( \text{Ag} = \{ 1, \ldots, n \} \) is a finite set of intelligent agents;
- \( G = (S, \text{Act}, \delta) \) is a fully PGS that is circular;
- \( s_0 \in S \) is the initial state;
- \( \text{Ap} \) is a finite set of atomic propositions;
- \( L : S \rightarrow 2^\text{Ap} \) is the state labelling function mapping each state to a set of atomic state proposition taken from \( \text{Ap} \);
- \( \text{obs} : \text{Paths}_s (s_0) \rightarrow (\Theta_1 \times \cdots \times \Theta_n)^\infty \) is an observation function for agent \( i \in \text{Ag} \).

<table>
<thead>
<tr>
<th>Actions</th>
<th>( \Theta_1 )</th>
<th>( \Theta_2 )</th>
<th>( \Theta_3 )</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{V}_1 )</td>
<td>( \text{X}_1 )</td>
<td>( \text{Y}_1 )</td>
<td>( \text{Z}_1 )</td>
<td>the chair opens a voting session</td>
</tr>
<tr>
<td>( \text{V}_2 )</td>
<td>( \text{X}_2 )</td>
<td>( \text{Y}_2 )</td>
<td>( \text{Z}_2 )</td>
<td>the chair closes a voting session</td>
</tr>
<tr>
<td>( \text{V}_3 )</td>
<td>( \text{X}_3 )</td>
<td>( \text{Y}_3 )</td>
<td>( \text{Z}_3 )</td>
<td>agent ( i ) is waiting, ( i \in { 0, 1, 2, 3 } )</td>
</tr>
<tr>
<td>( \text{V}_4 )</td>
<td>( \text{X}_4 )</td>
<td>( \text{Y}_4 )</td>
<td>( \text{Z}_4 )</td>
<td>voter 1 votes candidate ( X )</td>
</tr>
<tr>
<td>( \text{V}_5 )</td>
<td>( \text{X}_5 )</td>
<td>( \text{Y}_5 )</td>
<td>( \text{Z}_5 )</td>
<td>voter 2 votes candidate ( Y )</td>
</tr>
<tr>
<td>( \text{V}_6 )</td>
<td>( \text{X}_6 )</td>
<td>( \text{Y}_6 )</td>
<td>( \text{Z}_6 )</td>
<td>voter 3 votes candidate ( Y )</td>
</tr>
</tbody>
</table>

Table 1: Actions and observation functions in Example 1.

### 2.4 Strategies for agents in POMASs

Given a POMAS \( M = (\text{Ag}, G, s_0, \text{Ap}, L, \{ \text{obs} \}_i \in \text{Ag}) \), a mixed strategy of an agent \( i \in \text{Ag} \) specifies a way of choosing actions, based on her observation on the finite path starting with \( s_0 \) so far.

**Definition 7.** A mixed strategy for agent \( i \) is a function \( \pi_i : \text{Dist}(\text{Act}_i) \rightarrow \text{Dist}(\text{Act}_i) \) such that, if \( \pi_i (\rho) (a_i) > 0 \) then \( a_i \in \text{Act}_i (\text{last}(\rho)) \). The set of all strategies of agent \( i \) is denoted \( \Pi_i \).

**Definition 8.** A strategy profile for POMAS \( M \) is a tuple \( \pi = (\pi_1, \ldots, \pi_n) \in \Pi_1 \times \cdots \times \Pi_n \) producing a strategy for each agent of the system.

**Definition 9.** A path \( \rho \) is consistent with a strategy profile \( \pi \), denoted by \( \rho_\pi \), if it can be obtained by extending its prefixes using \( \pi \). Formally, \( \rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{t-1}} s_t \) is consistent with \( \pi \) if for all \( t \geq 0, i \in \text{Ag}, \) and history \( h \), we have: \( a_i^t \in \text{Act}_i (\pi_i (t)) \) and \( \delta (s_i, a_i^t) \xrightarrow{a_i} \rightarrow s_{t+1} \).

**Definition 10.** Given a POMAS \( M = (\text{Ag}, G, s_0, \text{Ap}, L, \{ \text{obs} \}_i \in \text{Ag}) \), a history is a finite path starting with \( s_0 \), the set of histories in \( M \) is written as \( \text{Hist}(M) \) and the set of histories in \( M \) starting with history \( h \) is written as \( \text{Hist}(M, h) \). For any agent \( i \in \text{Ag} \), and two histories \( h \) and \( h' \), we say \( h \) and \( h' \) are observationally equivalent to each other from \( i \)'s view, denoted by \( h \sim_i h' \), if \( \text{obs}_i (h) = \text{obs}_i (h') \).
Example 2. Consider the second scenario proposed in Example 1. The observer’s information of knowledge obtained from her observation might influence her decision on voting. Assume the observer (voter 3) is not able to see whom other voters have voted, but she can see how many ballots each candidate has received as specified in Table 1 and thus she can indicate the dominant candidate so far. A basic strategy to reflect such an influence is that she will vote the dominant candidate if there is one, otherwise she will vote the candidates under her preferred distribution.

3 Observability Specification

This section studies the problem of formally specifying observability of an agent on system behaviours modelled in POMAS.

3.1 Observability and opacity

Given a property $\varphi$ and an observation function $\text{obs}_i$ of an agent $i \in \text{Ag}$, we are interested in quantitatively expressing the observability of the agent that a set of agents has a strategy to enforce the property $\varphi$. The property can be viewed as a predicate, i.e., a set of execution paths for which it holds. The concept of Opacity [27] provides an intuitive approach for this task via distinguishing the observed behaviour and the original one. Intuitively, a property $\varphi$ is opaque (not observable), provided that for every behaviour (say path $\rho$) satisfying $\varphi$ there is another behaviour (say path $\rho'$), not satisfying $\varphi$, such that $\rho$ and $\rho'$ are observationally equivalent. So the observer is not able to determine whether the property in a given path of the system is satisfied or not. More precisely, opacity specifies whether an agent can establish a property $\varphi$, enforced by a strategy of a coalition $\text{A}$ of agents, at some specific state(s) of the executions of the system, according to her observation on the system behaviours. We use $[[\varphi]]$ to denote the set of paths satisfying property $\varphi$.

Definition 11. Let $\mathcal{M} = (\text{Ag}, G, s_0, Ap, L, \{\text{obs}_i\}_{i \in \text{Ag}})$. Given a predicate $\varphi$ over $\text{Paths}_G(s_0)$, we say $\varphi$ is opaque w.r.t. $\text{obs}_i$ if for every path $\rho \in [[\varphi]]$, there is a path $\rho' \in [[\varphi]]$ s.t. $\text{obs}(\rho) = \text{obs}(\rho')$, i.e., all paths satisfying $\varphi$ are covered by paths in $[[\varphi]]$, $\text{obs}_i([[\varphi]]) \subseteq \text{obs}_i([[\varphi]])$ under $\text{obs}_i$, where $[[\varphi]] = \text{Paths}_G(s_0) \setminus [[\varphi]]$.

3.2 The logic pPATL

To express the observability of an agent, we would consider the transparent paths, i.e., behaviours observable (non-opaque) to her. The level of observability can be considered as the degree of transparency of the property enforced by the strategy of a coalition, which can be measured by calculating the probability of the transparent paths satisfying the property. We now present pPATL, an extension of probabilistic alternating-time temporal logic (PATL) [14], that characterises agents’ quantified ability to enforce temporal properties. The key additions of pPATL include an observability operator and a probabilistic (observability) operator.

Definition 12. Let $\mathcal{M} = (\text{Ag}, G, s_0, Ap, L, \{\text{obs}_i\}_{i \in \text{Ag}})$. The syntax of pPATL includes three classes of formulae: state and path formulae, and observability formulae ranged over by $\psi$ and $\Phi$, respectively.

- $\phi ::= a | \neg \phi | \phi \land \phi | \text{P}_{\text{op}}(\text{A})[\psi] | \text{D}_{\text{op}}(\text{A})[\Phi]$
- $\psi ::= X\phi | \phi U \phi | \phi R \phi | \neg \psi | \psi \land \psi$
- $\Phi ::= O_i[\psi] | \Phi \land \Phi$

where $a \in \text{Ap}$ is an atomic proposition, $\text{A} \subseteq \text{Ag}$ is a set of agents, $\text{A}$ is the strategy quantifier, $\text{A}[\psi]$ expresses the property that coalition $\text{A}$ has a strategy to enforce $\psi$, $i \in \text{A} \subseteq \text{Ag}$ is an agent, $\preceq \in \{\leq, <, \geq, >\}$, $p \in [0, 1]$ is a probability bound.

Note that pPATL formula is defined relative to a state, path formulae are only allowed inside the observability operator $O_i[\cdot]$ and the probabilistic operator $\text{P}_{\text{op}}(\text{A})[\cdot]$. The formula $\text{P}_{\text{op}}(\text{A})[\psi]$ expresses that $A$ has a strategy such that the probability of satisfying path formula $\psi$ is $\preceq p$, when the strategy is followed. The observability formula $O_i[\psi]$ expresses the property of behaviours satisfying $[[\psi]]$ are observable to agent $i$. Intuitively, it is satisfied if for each path $\rho$ satisfying $\psi$ one cannot find a path $\rho'$ violating $\psi$ such that $\rho$ and $\rho'$ observationally equivalent to each other - from agent $i$’s view. This operator would allow us to reason about the observability of agent $i$ on system behaviours to enforce the property $\psi$. The quantitative observability formula $\text{D}_{\text{op}}(\text{A})[\Phi]$ expresses that $A$ has a strategy $\pi_A$ such that the degree of the observability enforcing path property considered in $\Phi$ is $\preceq p$. $\text{Paths}_G(s, \pi_A)$ is used to denote the set of all paths of $G$ starting from $s$ and consistent with $\pi_A$.

Definition 13. Let $\mathcal{M} = (\text{Ag}, G, s_0, Ap, L, \{\text{obs}_i\}_{i \in \text{Ag}})$. Semantics for pPATL include three satisfaction relations regarding the three notions of formulae (state, path, observability formulae).

For a state $s \in S$ of $G$, the satisfaction relation $s \models_M \phi$ for state formulae denotes “$s$ satisfies $\phi$”:

- $s \models_M a$ if $a \in L(s)$.
- $s \models_M \neg \phi$ if $s \not\models_M \phi$.
- $s \models_M \phi \land \phi'$ if $s \models_M \phi$ and $s \models_M \phi'$.

For a path $\rho \in \mathcal{G}$, the satisfaction relation $s \models_M \text{P}_{\text{op}}(\text{A})[\psi]$ expresses that for each path $\rho$ satisfying $\psi$, i.e., all paths satisfying $\psi$ are covered by paths in $[[\psi]]$, $\text{obs}_i([[\psi]]) \subseteq \text{obs}_i([[\psi]])$ under $\text{obs}_i$, where $[[\psi]] = \text{Paths}_G(s_0) \setminus [[\psi]]$.

For a path $\rho$ of $G$, we define:

- $\rho \models_M X\phi$ if $\rho_s(1) = \phi$.
- $\rho \models_M \phi U \phi'$ if there exists $i \in \mathbb{N}$ s.t. $\rho_s(i) \models_M \phi'$ and $\rho_s(j) \models_M \phi$ for all $j < i$.
- $\rho \models_M \phi R \phi'$ if for all $i \in \mathbb{N}$ at least one of the following is true: $i$) $\rho_s(i) \models_M \phi'$, ii) $\rho_s(j) \models_M \phi$ for some $j < i$.
- $\rho \models_M \neg \phi$ if $\rho \not\models_M \phi$.
- $\rho \models_M \phi \land \phi'$ if $\rho \models_M \phi$ and $\rho \models_M \phi'$.

Finally, for $s \models_M \Phi$, we define observability formulae $\Phi$:

- $s \models_M O_i[\psi]$ if for each path $\rho \in \text{Paths}_G(s)$ s.t. $\rho \models_M \psi$, and for all $\rho' \in \text{Paths}_G(s)$ s.t. $\rho' \models_M \psi$: $\text{obs}_i(\rho) \neq \text{obs}_i(\rho')$.
- $s \models_M \neg \Phi$ if $s \not\models_M \Phi$, i.e., for each path $\rho \in \text{Paths}_G(s)$ s.t. $\rho \models_M \psi$, there exists a path $\rho' \in \text{Paths}_G(s)$ s.t. $\rho' \models_M \psi$: $\text{obs}_i(\rho) = \text{obs}_i(\rho')$.
- $s \models_M \Phi \land \Phi'$ if $s \models_M \Phi$ and $s \models_M \Phi'$.

Example 3. Consider the model in Example 1. Assume we are interested in analysing the observability of voter 2 regarding her observation function in Table 1. Consider the property “eventually candidate
X wins, i.e., ψ = F(cx > cy ∧ o = 2), where cx and cy denote the final ballots X and Y received, and o = 2 indicates the state of voting process being closed. The operator Fψ is defined as "true ∪ φ", so the probabilistic observability property is specified as: D_{φ}(1, 2, 3)[O_2[ψ]], where p = 1/4 is a probability threshold. It is easy to notice that the above formula would return true, since ("wait" actions have been omitted here for simplifying the expression without introducing any confusion):

$$tr([φ]) = \{ \frac{1}{12} p_0 V_1 V_2 V_3 X C_1, \frac{1}{6} p_0 V_1 V_2 V_3 Y C_1, \frac{1}{12} p_0 V_1 V_2 V_3 X C_1, \}$$

and thus,

$$tr([O_2[ψ]]) = \{ \frac{1}{12} p_0 V_1 V_2 V_3 X C_1, \frac{1}{6} p_0 V_1 V_2 V_3 Y C_1, \}$$

this is because under the observation function specified in Table 1, traces p_0 V_1 V_2 V_3 X C_1 and p_0 V_1 V_2 V_3 Y C_1 are covered by violating traces p_0 V_1 V_2 V_3 Y C_1 and p_0 V_1 V_2 V_3 Y C_1 respectively from voter 2’s view:

$$obs_2(p_0 V_1 V_2 V_3 Y C_1) = obs_2(p_0 V_1 V_2 V_3 Y C_1) = p_0 V_2 C_1,$n
$$obs_2(p_0 V_1 V_2 V_3 X C_1) = obs_2(p_0 V_1 V_2 V_3 X C_1) = p_0 X_2 C_1.\]$$

Therefore,

$$\text{Prob}([1, 2, 3]O_2[ψ]) = \text{Prob}(\{ \frac{1}{12} p_0 V_1 V_2 V_3 X C_1, \frac{1}{6} p_0 V_1 V_2 V_3 Y C_1, \}) = \frac{1}{4}$$

which is less than p = 1/4 and thus D_{φ}(1, 2, 3)[O_2[ψ]] returns true.

4 Verification of Observability Properties

Intuitively, verification of probabilistic observability answers the question "to which degree the system is observable to an agent i ∈ Ag?", relative to a task expressed as property [ψ] following the strategy of a coalition A ⊆ Ag, and the observation function of the agent obs_i. Since oPATL is a branching time logic, the overall approach is to recursively compute the satisfaction set Sat(φ) of states satisfying formula φ over the structure of the formula.

For the propositional logic fragment of oPATL, the computation of this set for atomic propositions and logical connectives follows the conventional CTL model checking [4] and is sketched below:

1. Convert the oPATL formulae in a positive normal form, that is, formulae built by the basic modalities O[Xφ], O[φ U φ'], and O[φ R φ'], and successively pushing negations inside the formula at hand: ¬true → false, ¬false → true, ¬¬φ → φ, ¬(φ ∧ φ') → ¬φ ∨ ¬φ', ¬(φ ∨ φ') → ¬φ ∧ ¬φ', ¬Xφ → X¬φ, ¬(Oφ) → ¬φRφ', ¬φRφ' → ¬φU¬φ';

2. Recursively compute the satisfaction sets Sat(φ') = { s ∈ S | s |= φ'} for all state subformulae φ' of φ: the computation carries out a bottom-up traversal of the parse tree of the state formula φ starting from the leaves of the parse tree and completing at the root of the tree which corresponds to φ, where the nodes of the parse tree represent the subformulae of φ and the leaves represent an atomic proposition α ∈ Ap or true or false. All inner nodes are labelled with an operator. For positive normal form formulae, the labels of the inner nodes are ¬, ∧, O[X], O[U], O[R]. At each inner node, the results of the computations of its children are used and combined to build the states of its associated subformula. In particular, satisfaction sets for the propositional logic fragment state formula are given as follows:

- Sat(true) = S,
- Sat(α) = { t ∈ S | t ∈ η(t) },
- Sat(¬φ) = S \ Sat(φ),
- Sat(φ ∧ φ') = Sat(φ) ∩ Sat(φ'),
- Sat(φ ∨ φ') = Sat(φ) ∪ Sat(φ'),
- Sat(Oφ) = { s ∈ S | ∃ s' ∈ S: s' |= φ ∧ s = s' U φ'

(3) Check whether s ∈ Sat(φ).

For the treatment of subformulae of the form φ = P_{obs}(A)[ψ], in order to determine whether s ∈ Sat(φ), the probability of consistent paths with π_a under coalition A for behaviour specified by ψ, i.e., Prob(s |= M(A)[ψ]), needs to be established, then:

Sat(P_{obs}(A)[ψ]) = { s ∈ S | Prob(s |= M(A)[ψ]) ≥ p }

The computation of the probability can follow the conventional PATL model checking algorithms, e.g., [14].

We now focus on the treatment state formulae of the form D_{obs}(O_1_A[ψ]). The problem reduces to computing the probability of observable paths that are satisfying property ψ and consistent with strategies of coalition A, from agent i’s view.

Definition 14. Given a POMAS \( M = (G, s_0, Ag, Ap, \{ obs_i \}_{i∈Ag}) \), a task in property ψ required to be completed under a strategy π_A of a coalition A ∈ Ag, the probabilistic verification problem of observability property is to decide whether s_0 |= M D_{obs}(O_1_A[ψ]), i.e.,

$$\mathbb{P}_{s_0}([\{A[ψ] \mid obs_i([\{A[ψ] \mid ¬ψ_i]]) \}]) = \exists p \cdot p \geq p.$$

Therefore, we focus on computing \( \mathbb{P}_{s_0}([A[ψ] \mid obs_i([A[ψ] \mid ¬ψ_i]]) \} \) for a given POMAS M and coalition A. We assume that the available actions of agent i ∈ Ag of M in state s are \( \{ a_{i1}, \ldots, a_{iK} \} \). The brief procedure for checking s_0 |= M D_{obs}(O_1_A[ψ]) is sketched as follows.

- Find all consistent paths Π and the corresponding traces Λ, represented in regular-expression-like format (denoted by Reg(·)), satisfying ψ under mixed strategy π_A of coalition A, denoted by:

\[ \Pi = \{ \text{Reg}(\rho_{π_A}) \mid ρ_{π_A} \models M ψ \} \]

Λ = { erase(ρ) | ρ ∈ Π }.

- Find all consistent paths Π’ and the corresponding traces Λ, represented in regular-expression-like format (denoted by Reg(·)), violating ψ under mixed strategy π_A of coalition A:

\[ \Pi' = \{ \text{Reg}(ρ'_{π_A}) \mid ρ'_{π_A} \not\models M ψ \} \]

Λ’ = { erase(ρ’) | ρ’ ∈ Π’ }.

- Find all ψ-opaque traces:

\[ \Lambda'' = \{ \lambda'' | \lambda'' ∈ Λ ∧ \exists \lambda' (obs_i(\lambda') = obs_i(\lambda'')) \} \]

- Compute the probability of ψ-observable traces:

\[ d = \mathbb{P}_{s_0}([O_1(A)[ψ]]) = \sum_{ξ ∈ (Λ'\Lambda'')} \text{Prob}(ξ). \]

- Return true if d ≥ p, return false otherwise.

We present the detailed procedure of computing the probability of ψ-observable traces starting at s under mixed strategy π_A of coalition A from the observation of i ∈ Ag, in Algorithm 1. Algorithm 2 computes a set of regular-expression-like formatted paths satisfying φRφ’. Similarly, an algorithm can be proposed to compute a set of regular-expression-like formatted paths satisfying φUφ’. We can thus compute all regular-expression-like formatted paths Π (and Π’) starting from s and satisfying (and violating) ψ and consistent with mixed strategy π_A.

Soundness. Given a POMAS M, a probability threshold p, and a task specified in ψ to be completed:

\[ s_0 |= M D_{obs}(A)[O_1[ψ]] \quad \text{iff} \quad \mathbb{P}([\{A[ψ] \mid ¬ψ_i]]) \geq p. \]
The satisfaction relation of $D_{spec}(A)[O_1[\psi]]$ and the computation of the probability of $\psi$-observable consistent traces under mixed strategy $\pi_A$ of coalition A from observer i’s view is described in Algorithm 1. The algorithm will terminate since $Sat(\psi)$ are processed and computed as a set of regular-expression-like formatted traces satisfying $\psi$ (such as Algorithm 2 presented in the technical appendix). Probability of such a trace is calculated by multiplication of the probability of each transition label for non-cycle part, and multiplication of $p/(1-p)$ for a cycle with probability $p$.

**Complexity.** The worst case of checking satisfaction of the observability formula, specified in Algorithm 1, is EXPSPACE in general. The formula of observability is essentially in the form of $\forall \forall V$, the algorithm traverses all consistent traces under mixed strategy $\pi_A$ satisfying $\psi$ and all traces of those violating $\psi$, and conducts observation equivalence comparison. So the worst case complexity here follows the complexity of the hyper property model checking problem with two quantifier ($\forall$) alternations, and thus EXPSPACE. We hypothesise the time complexity of checking satisfaction of $\Psi_{LTL}$ formula is exponential to the size of the POMAS, and doubly exponential in the size of the formula itself, similar to model checking HyperLTL [16]. If all pairs of traces are evaluated in parallel, the evaluation of each individual pair corresponds to the evaluation of an LTL formula over a single trace, which can be done in polylogarithmic time on a parallel computer with a polynomial number of processors [9].

**Example 4.** The proposed work has been implemented on top of PRISM, which allows to specify properties which evaluate to a value using, e.g., $D_{spec}(A)[O_1[\psi]]$. The result of Example 3 can be automatically produced below, which meets our calculation by hand.

**Algorithm 1: Computing the probability of $\psi$-observable consistent traces under $\pi_A$ from i’s view - $D(\langle A \rangle O_1[\psi])$.**

<table>
<thead>
<tr>
<th>Data: $M, s, i, A, \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result: The probability $D(\langle A \rangle O_1[\psi])$</td>
</tr>
<tr>
<td>switch $\psi$ do</td>
</tr>
<tr>
<td>case $X\phi$:</td>
</tr>
<tr>
<td>$Sat(\psi) \leftarrow { s \cup { s' \mid Post(s) = s' \land s' \in Sat(\phi) } }$</td>
</tr>
<tr>
<td>$Sat(\neg \psi) \leftarrow { s \cup { s' \mid Post(s) = s' \land s' \in Sat(\neg \phi) } }$</td>
</tr>
<tr>
<td>case $\phi U \phi'$: $Sat(\psi) \leftarrow compU(M, s, \phi, \phi')$</td>
</tr>
<tr>
<td>$Sat(\neg \psi) \leftarrow compR(M, s, \neg \phi, \neg \phi')$</td>
</tr>
<tr>
<td>case $\phi R \phi'$: $Sat(\psi) \leftarrow compR(M, s, \phi, \phi')$</td>
</tr>
<tr>
<td>$Sat(\neg \psi) \leftarrow compU(M, s, \neg \phi, \neg \phi')$</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>$pA \leftarrow { pA</td>
</tr>
<tr>
<td>$pA^* \leftarrow { pA</td>
</tr>
<tr>
<td>$pA'' \leftarrow { pA</td>
</tr>
<tr>
<td>$pA,pr \leftarrow pA',pr$ for each $pA' \in P$ do</td>
</tr>
<tr>
<td>$pA'' \leftarrow pA'' \cup { pA }$; break;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>$d = \sum_\lambda pA,pr$</td>
</tr>
<tr>
<td>return $d$.</td>
</tr>
</tbody>
</table>

**Algorithm 2: Computing Sat($\phi U \phi'$) - $compU(M, s, \phi, \phi')$.**

<table>
<thead>
<tr>
<th>Data: $M, s, \phi, \phi'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result: Regular-expression-like formatted paths satisfying $\phi U \phi'$</td>
</tr>
<tr>
<td>$\Pi \leftarrow { ; i \leftarrow 0 ;$</td>
</tr>
<tr>
<td>for each $t_i \in Sat(\phi')$ do</td>
</tr>
<tr>
<td>$T_i \leftarrow { t_i }; \Pi_i \leftarrow { \pi</td>
</tr>
<tr>
<td>while</td>
</tr>
<tr>
<td>${ s_j \in Sat(\phi) \mid (T_i \cup Sat(\phi') } \land Post(s_j) \neq 0 \neq \emptyset$ do</td>
</tr>
<tr>
<td>let</td>
</tr>
<tr>
<td>$s_j \leftarrow { s_j \in Sat(\phi) \mid (T_i \cup Sat(\phi')) } \land Post(s_j) \neq 0 \neq \emptyset$;</td>
</tr>
<tr>
<td>if $s_j \in Post(s_j) \cap T_i$ then</td>
</tr>
<tr>
<td>/* There is a self-loop: wrap it with a star and concatenate paths starting from a state in Post(s_j) \cap T_i */</td>
</tr>
<tr>
<td>for each $\pi' \in \Pi_i s.t. \pi'(0) = Post(s_j) \cap T_i$ do</td>
</tr>
<tr>
<td>$\Pi_i \leftarrow \Pi_i \cup { (s_j \xrightarrow{\pi' \ast} s_j') \ast \pi'[1...]) }$</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>
| for each $q_1 \in Post(s_j) \cap T_i, q_2 \in Post(q_1) \cap T_i, ... q_n \in Post(q_{n-1}) \cap T_i s.t. |$
| Post(q_n) \cap T_i = \emptyset$ do |
| if $Pre(s_j) \subseteq \{ q_1, q_2, ... q_n \}$ then |
| for each $\pi' \in \Pi_i s.t. \pi'(0) \in Post(s_j) \cap T_i$ do |
| $\Pi_i \leftarrow \Pi_i \cup \{ (s_j \xrightarrow{\pi' \ast} q_1, \xrightarrow{\pi' \ast} ... q_n) \}$ |
| end |
| else if $Pre(s_j) = q_n \land s_j \in Post(q_n)$ then |
| /* There is a cycle, wrap it with a star and concatenate paths starting from a state in Post(s_j) \cap T_i */ |
| for each $\pi' \in \Pi_i s.t. \pi'(0) \in Post(s_j) \cap T_i$ do |
| $\Pi_i \leftarrow \Pi_i \cup \{ (s_j \xrightarrow{\pi' \ast} q_1 \xrightarrow{\pi' \ast} ... \xrightarrow{\pi' \ast} q_n) \}$ |
| end |
| end |
| $\Pi_i \leftarrow \Pi_i \cup \{ s_j \}$ |
| end |
| $i \leftarrow i + 1;$ |
| end |
| return $\Pi$. |

5 Implementation and Examples

A prototype tool for specifying and verifying the observability problem in MASs has been built on the top of the PRISM model checker [22]. Models are described in an extension of the PRISM modelling language with observations and transition labels, the new model type is denoted as “pomas”. Properties are described in an extension of the PRISM’s property specification language with the observability operator. The tool and examples are available from [30].

Example 4: A simple supply chain. Nowadays supply chain is a core part of businesses concerned with transporting products between different parties such as customers, retailers, coordinators, delivery services, and suppliers. Agents of those parties communicate with each other for buying and selling items. Suppliers compete with each others to obtain more jobs and profit, they might partially observe the procedure of the supply chain and try to induce commercial information. Customer might partially observe the pipeline of the supply chain, and try to learn information about the origin of the products. Such a scenario can be naturally modelled as a POMAS, and we are interested in studying the quantified observability by agents, which may cause information flow and affect future decision-making.

To illustrate our framework and its implementation, we consider a basic commercial supply chain shown in Fig. 1 as an example. Assume there are a number of agents in the system: 1) customer: buying products (denoted by order) from the retailer; 2) retailer: requesting to order products (denoted by order) from suppliers through the coordinator; 3) coordinator: the coordinator, processing requests/orders from the retailer, sending requests to and receiving response from suppliers.
pliers, making decisions such as which supplier provides products, returning decision to the retailers; 4) supi: the \( i \)th supplier, receiving requests from/responding availability to the coordinator. The agents and their observability are given as follows: action \( \text{ordc} \) is hidden to \( \text{supi} \) and is observed as \( \text{Ordc} \) from the rest of the agents’ view; action \( \text{ordr} \) is hidden to \( \text{customer} \) and is observed as \( \text{Ordr} \) from the rest of the agents’ view; action \( \text{reqi} \), denoting \( \text{coord} \) sending requests to \( \text{supi} \), is hidden to \( \text{customer} \) & \( \text{retailer} \), is observed as \( \text{Req} \) to \( \text{supi} \), and is observed as \( \text{Reqi} \) to \( \text{coord} \); action \( \text{resi} \), denoting \( \text{supi} \) responding to \( \text{coord} \), is hidden to \( \text{customer} \) & \( \text{retailer} \), is observed as \( \text{Resi} \) to \( \text{supi} \), and is observed as \( \text{Resi} \) to \( \text{coord} \); action \( \text{decisioni} \), denoting \( \text{coord} \) deciding \( \text{supi} \) to provide the products, is hidden to \( \text{customer} \) & \( \text{retailer} \), is observed as \( \text{Deci} \) to \( \text{supi} \), and is observed as \( \text{Deci} \) to \( \text{coord} \); action \( \text{delivery} \) is observed \( \text{Dlv} \) to all of the agents.

Let \( A \) denote the set of agents defined above. We could ask questions such as “what is the degree of the observability by \( \text{supi} \) if the product is successfully delivered to the customer but the supplier is not \( \text{supi} \)”, specified as \( \text{P} = \{ [A_{\text{supi}} \mid F \text{(dec} = 1 \& \text{dlv} = 1)] \} \), where \( \text{deci} \) is the variable defined in the module to specify the decision made by the coordinator: \( \text{dec} = 1 \) denotes supplier \( i \) will provide the requested product, \( \text{dlv} = 1 \) is the variable defined in the module to specify the status of product delivery: \( \text{dlv} = 1 \) denotes the product has been successfully delivered to the customer. The result generated by the tool is presented as follows:

\[
\text{Result: 0.5.}
\]

This meets our intuition, since the listed two traces satisfying \( F(\text{dec} = 1 \& \text{dlv} = 1) \) are not covered by traces violating the property, are thus observable to \( \text{supi} \). If we ask question “what is the degree of the observability by \( \text{customer} \) if the product is successfully delivered to him but the supplier is not \( \text{supi} \)”, which can be specified as \( \text{P} = \{ [A_{\text{customer}} \mid F \text{(dec} = 1 \& \text{dlv} = 1)] \} \). The result generated by the tool would be:

\[
\text{Result: 0.0.}
\]

Example: A peer-to-peer (P2P) file sharing network. This case study considers a variant of a Gnutella-like P2P network for file sharing, allowing users to communicate and access files without the need for a server. The individual users in this network are referred to as peers. Gnutella protocol defines a decentralised approach making use of distributed systems, where the peers are called nodes, and the connection between peers is called an edge between the nodes, thus resulting in a graph-like structure. A peer wishing to download a file would send a query request \( \text{Qry} \) packet to all its neighbouring nodes under a probability distribution. If those nodes don’t have the required file, they pass on the query to their neighbours and so on. When the peer with the requested file is found, the query flooding stops and it sends back a query hit packet \( \text{Hit} \) following the reverse path. If there are multiple query hits, the client selects one of these peers. The client thus builds a connection with the peer offering the resource and download the resource. Fig. 2 shows an example process of downloading a file using the Gnutella-like P2P network.
References


