Mining and Validating Belief-Based Agent Explanations

Ahmad Alelaimat(✉), Aditya Ghose, and Hoa Khanh Dam

Decision Systems Lab, School of Computing and Information Technology University of Wollongong, Wollongong 2522, Australia
{aama963,aditya,hoa}@uow.edu.au

Abstract. Agent explanation generation is the task of justifying the decisions of an agent after observing its behaviour. Much of the previous explanation generation approaches can theoretically do so, but assuming the availability of explanation generation modules, reliable observations, and deterministic execution of plans. However, in real-life settings, explanation generation modules are not readily available, unreliable observations are frequently encountered, and plans are non-deterministic. We seek in this work to address these challenges. This work presents a data-driven approach to mining and validating explanations (and specifically belief-based explanations) of agent actions. Our approach leverages the historical data associated with agent system execution, which describes action execution events and external events (represented as beliefs). We present an empirical evaluation, which suggests that our approach to mining and validating belief-based explanations can be practical.

Keywords: Explainable agents · Mining explanations · BDI agents

1 Introduction

Explainable agents have been the subject of considerable attention in recent literature. Much of this attention involves folk psychology [1], which seeks to explain the actions of an agent by citing its mental state (e.g., the beliefs of the agent, and its goals and intentions). Roughly speaking, when an explainee requests explanations about a particular action, two common explanation styles might be adopted: (1) a goal-based explanation and (2) a belief-based explanation [2–4]. This paper focuses on the latter style[1] in the context of the well-known Belief-Desire-Intention (BDI) paradigm [5]. Fundamentally, belief-based explanations help answer the following question: *What must have been known for the agent to perform a particular action over another?* Arguably, a sufficiently

A. Ghose-Passed away prior to the submission of the manuscript. This is one of the last contributions by Aditya Ghose.

[1] We submit that goal-based explanations are also of great value to develop explainable agents, and we believe that an extension of the techniques presented in this work can address these but are outside the scope of the present work.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
D. Calvaresi et al. (Eds.): EXTRAAMAS 2023, LNAI 14127, pp. 3–17, 2023.
https://doi.org/10.1007/978-3-031-40878-6_1

detailed explanation (e.g., one that justifies an action with extensive information about the agent reasoning) will require no additional information to answer this question. Nevertheless, in certain settings (e.g., time-constrained environments), explanations are more useful when they are relatively unfaithful [6,7]. Explaining by beliefs can also help solve a range of problems, such as encouraging behaviour change [4], enhancing human-agent teaming [8], and transparency [9].

Although there is a large and growing body of work on explanation generation in the field of autonomous agents [1], much of this work has traditionally assumed the availability of explanation generation modules, reliable observations, and deterministic execution of plans. However, in real-life settings, autonomous agents are not explainable by design, unreliable observations are frequently encountered, and plans are non-deterministic. We seek in this work to address these challenges by proposing techniques that mine the historical data associated with agent system execution to generate belief-based explanations of agents' past actions. We shall refer to this problem as *mining belief-based explanations*. Our proposal relies on the notion of audit logging. Many of the existing MAS frameworks (e.g., JACK framework [10]) support logging different aspects of agent behaviour. Of these, we are interested in two particular aspects: (1) a behaviour log that records the creation and completion of the past executed actions, and (2) a belief log that describes the belief set activity of the agent during the recording of (1). One such implementation of audit logging is the tracing and logging tools for JACK Intelligent Agents [10]. Our approach to mining belief-based explanation involves two steps:

1. We leverage these two audit logs in chronological order to generate one sequence database taken as input by a sequential pattern miner. The intuition behind this is to identify commonly occurring patterns of action execution events preceded by sequences of beliefs. Here, we intend to mine the enabling beliefs of each action referred to in the behaviour log.
2. We define a validation technique that leverages a state update operator (i.e., an operator that defines how the specification of a belief state is updated as a consequence of the agent's perception of the environment), agent's past experiences provided by the above-cited sequence database to compute the soundness and completeness of the mined belief-based explanations.

Mining and validating belief-based explanations can be outlined as follows: Given as inputs (1) a behaviour log of past executed actions, (2) a belief log, (3) a plan or plans that execution generated these logs, (4) a state update operator, compute: the belief-based explanations of every action referred to in the behaviour log. While inputs (1) and (2) are used for mining belief-based explanations, inputs (3) and (4) are used for validating the mined explanations. As we show later in this work, inputs (3) and (4) can also be used to generate detailed belief-based explanations.

The remainder of this work is organized as follows. Section 2 introduces our running example and some required preliminaries. Section 3 describes our approach to updating belief-based explanations, which sits at the core of this work.

Section 4 describes our approach to mining belief-based explanations. In Sect. 5, we describe how the mined explanations can be validated. Section 6 reports an empirical evaluation of this work. Related work is discussed in Sect. 7 before we conclude and outline future work in Sect. 8.

2 Preliminaries and Running Example

Agents with BDI architecture are designed to imitate human practical reasoning using beliefs, desires and intentions. The beliefs represent what the agent knows about the environment, the desires are goals (i.e., objectives) that the agent would like to bring about, and the intentions are plans that the agent is committed to executing. With these anthropomorphic handles, the BDI agent derives its actions and, consequently, can explain them. As the literature (e.g., [11]) suggests, this is an elegant means to explain agent systems with considerable underlying complexity. Although an explainable BDI agent must faithfully reflect its reasoning cycle (i.e., including its beliefs, goals, plans, and intentions), explaining by beliefs can provide value in ways other handles cannot. To illustrate this, we study two scenarios for Engine failure on take-off (EFTO): (1) accelerate-stop and (2) accelerate-go, as described in [12].

Example 1. As shown in Fig. 1, the pilot agent has two plans written on the basis of Jason agent programming language [13] to handle EFTO in large twin-engine jet aircraft: (`p1`) accelerate-stop and (`p2`) accelerate-go. `p1` involves reducing thrust, applying speed breaks and notifying the tower of the emergency. `p2` begins with ensuring full power is applied (including mixture, throttle, landing gear and flaps), followed by liftoff, and then notifying the tower of the emergency and the intended landing. To handle an EFTO successfully, two critical speeds must be calculated before each take-off, namely V1 (the speed at which the pilot can abort the take-off safely) and V2 (the speed at which the pilot can take-off safely).

```
@p1 +efto(aircraft):  V1 < Speed < V2
   <- idle(throttle);
      deploy(brakes);
      .send(tower,tell,stop(accelerate)).
@p2 +efto(aircraft):  V1 < Speed < V2
   <- increase(mixture);
      increase(throttle);
      take_up(flap);
      pull(yoke);
      take_up(gear);
      .send(tower,tell,go(accelerate));
      .send(tower,tell,return(landing)).                   ,
```

Fig. 1. Jason plans for handling EFTO

A particularly stressful situation for the pilot is when EFTO occurs between V1 and V2 (i.e., the aircraft is going too fast to accelerate-stop but too low to accelerate-go). For this particular situation, the two plans ($\mathsf{p1}$ and $\mathsf{p2}$) are applicable (i.e., the plan selection is based on some conditions not represented in the agent code).

Existing explanation generation techniques can be summarized as follows: A BDI agent triggers an action with respect to its goals and beliefs, which can be represented in terms of a Goal Hierarchy Tree (GHT) [2–4]. A GHT is a tree structure representing a high-level abstraction of an agent's reasoning. At the root of the tree, the agent's main goal is placed. A link from the top-level goal to one or more sub-goals means that these sub-goals must be achieved as part of the top-level goal. Tree leaves represent actions that the agent can execute. For the agent to execute an action, certain beliefs placed directly above the action must be true. What existing explanation algorithms do is select the beliefs and goals that are directly above the selected action node in the GHT to design explanation patterns. We argue that such explanation generation techniques can provide irrational explanations in many settings. To illustrate this consider the following scenarios.

Example 2. Assume that the pilot decided to accelerate-stop at one instance and accelerate-go at another instance in the past. Now, consider the following queries that may arise by an aviation candidate:

– *Why did the pilot agent pull the throttle lever to idle immediately after the EFTO?*
– *Why did the pilot agent move the mixture knob to rich after the EFTO?*

Following the current norms of explanation generation, the two queries in Example 2 can be readily answered using the goal $\mathsf{efto(aircraft)}$ and the belief that $\mathsf{V1} < \mathsf{Speed} < \mathsf{V2}$. Another way to say that both queries are explained by the same goal and same belief. It is clear that these explanations are inaccurate - they do not accurately describe how the pilot agent came to its decision to accelerate-stop at the first instance nor to accelerate-go at the later instance.

2.1 Audit Logs

During agent systems execution, a wide variety of data on changes in the agent's mental attitude and environment can be represented in the form of audit logs. Collecting such data can be implemented using audit logging tools such as Mind Inspector in Jason platform [13] and Design Tracing Tool (DTT) in JACK platform [10]. We are interested in two modes of audit logging: (1) behaviour logs and (2) belief logs. A behaviour log describes the historical execution of plans as sequences of events where each event refers to some action. A behaviour log can be represented as a set of triples $\langle \mathsf{p\_label}, \mathsf{t_i}, \mathsf{a_i} \rangle$, where the value of $\mathsf{t_i}$ refers to the starting time of the action $\mathsf{a_i}$, which has been executed as part of a plan labelled $\mathsf{p\_label}$. An excerpt of the behaviour log associated with the plans in our running example during two different flights is recorded in Table 1.

Table 1. A behaviour log of the pilot agent.

Flight No.	Plan	Timestamp	Action
2065	p1	t75	idle(throttle)
2065	p1	t77	deploy(brakes)
2065	p1	t80	send(tower, msg)
2072	p2	t1027	increase(mixture)
2072	p2	t1029	increase(throttle)
2072	p2	t1031	take up(flap)
2072	p2	t1033	pull(yoke)
2072	p2	t1035	take up(gear)
2072	p2	t1037	send(tower, msg)
2072	p2	t1038	send(tower, msg)

A belief log records the history of the external events perceived by the target agent. It consists of a set of couples $\langle t_i, q_i \rangle$, where t_i value indicates the time when the agent added the belief q_i to its belief base. Note that belief logs[2] record new beliefs as they are added to the belief base but do not record persistent beliefs. Determining which beliefs hold at a certain point of system execution, therefore, requires updating machinery (e.g., the state update operator described in Subsect. 3). Table 2 illustrates an excerpt of a belief log describing external events perceived by the pilot agent during two different flights.

Table 2. A belief log of the pilot agent

Flight No.	Timestamp	Beliefs	Flight No.	Timestamp	Beliefs
2065	t70	runway(dry)	2072	t1024	efto
2065	t71	wind(cross)	2072	t1025	V1 = 156
2065	t72	efto	2072	t1025	V2 = 166
2065	t73	V1 = 129	2072	t1025	Flaps = 15
2065	t73	V2 = 145	2072	t1026	Speed = 161
2065	t73	Flaps = 15	2072	t1028	escalating(fuel flow)
2065	t74	Speed = 135	2072	t1030	accelerating(thrust)
2065	t76	decelerating(thrust)	2072	t1032	Flaps = 0
2065	t78	steady(aircraft)	2072	t1034	liftoff(aircraft)
2072	t1022	runway(wet)	2072	t1036	up(gear)
2072	t1023	wind(head)	2072	t1040	liftoff(aircraft)

[2] One can leverage JACK capability methods to make belief set activities available at agent level [14]. This manipulation allows, in turn, to store of enabling beliefs based on the user-defined data structure.

Normally, it is more convenient to represent beliefs in first-order sentences to maintain consistency and constraints. To avoid handling different groundings of the variables as distinct beliefs, we need to neglect the precise grounding of valuables. Nevertheless, there are settings where we need precise instantiations of the variables.

3 Updating Belief-Based Explanations

Updating belief-based explanations is useful for at least two reasons. First, it could be used to contextualise explanations (i.e., providing users with detailed explanations). Another way to say that updated belief-based explanations can help answer the following question for any step of plan execution: *What must have been known in detail for the agent to perform a particular action over another?* As a second reason, it could also be used to validate the mined explanations, which we describe in detail in later sections.

At each action step in a plan execution, we derive the updated belief-based explanation of an action by combining the enabling beliefs of the preceding actions with the enabling beliefs of the action we are at. For the purpose of this work, we ignore other constructs appearing in a plan body (e.g., achievement and test goals). We assume that each action in a plan is associated with enabling beliefs (i.e., no provisional execution of actions) written as conjunctive normal-form sentences using a state description language that might involve propositional variables (i.e., variables that can be true or false) and non-Boolean variables (i.e., new value assignments). We allow the updated belief-based explanations to be non-deterministic for two reasons (1) in any plan with OR branching, one might arrive at an action through multiple trajectories, and (2) much of the existing state update operators resolve inconsistencies in multiple different ways. Among the two well-known state update operators in the literature - the Possible Worlds Approach (PWA) [15], and the Possible Models Approach (PMA) [16] - our work relies on the PWA. More precisely, we use the state update operator $\oplus$ as defined below, assuming the presence of a background knowledge base KB.

Definition 1 ($\oplus$ Operator). For the two belief states $\mathsf{s_i}$, $\mathsf{s_j}$, and the knowledge base KB, the state update operator $\oplus$ can be defined as follows:

$$\mathsf{s_i} \oplus \mathsf{s_j} = \{\mathsf{s_j} \cup \mathsf{s_i'} \mid (\mathsf{s_i} \wedge \mathsf{s_i'} \cup \mathsf{s_j} \cup \mathsf{KB} \not\models \perp) \wedge (\nexists\ \mathsf{s_i''} \text{ such that } \\ \mathsf{s_i'} \subset \mathsf{s_i''} \subseteq \mathsf{s_i} \wedge \mathsf{s_i''} \cup \mathsf{s_j} \cup \mathsf{KB} \not\models \perp)\},$$

in which if $\mathsf{s_j} \cup \mathsf{s_i}$ is consistent, then the resulting updated explanation is $\mathsf{s_j} \cup \mathsf{s_i}$. Otherwise, we need to define $\mathsf{s_i'} \subseteq \mathsf{s_i}$ such that $\mathsf{s_j} \cup \mathsf{s_i'}$ is consistent and there is no exists $\mathsf{s_i''}$ such that $\mathsf{s_i'} \subset \mathsf{s_i''} \subseteq \mathsf{s_i}$ and $\mathsf{s_j} \cup \mathsf{s_i''}$ is consistent. Note that we might need to refer to a general version of the state update operator, i.e., if $\mathsf{S} = \{\mathsf{s_1}, \ldots, \mathsf{s_n}\}$ is a finite set of belief states, then $\mathsf{S} \oplus \mathsf{s} = \{\mathsf{s_i} \oplus \mathsf{s} \mid \mathsf{s_i} \in \mathsf{S}\}$. Note also that the output of the state update operator is not always unique state specifications. Actually, the output might be a set of non-deterministic possible belief states. For the purpose of illustrating why this might be the case, we consider the following example.

Example 3. Consider the following knowledge base

$$\mathsf{KB} = \mathsf{r} \rightarrow \neg\,(\mathsf{d}\ \wedge\ \mathsf{q})$$

representing a rule for the pilot agent, where the propositional letter r can be read as there is an EFTO, the letter d as the thrust is accelerating, and the letter q as the aircraft is ascending. Now, let $(\mathsf{d}\ \wedge\ \mathsf{q})$ hold in some previous belief state, and r came to be held in the belief state where we are at. Applying $\oplus$, the generated two alternative scenarios describing the updated belief states are

1. $\{\mathsf{d} \wedge \mathsf{r}\}$ and
2. $\{\mathsf{q} \wedge \mathsf{r}\}$

which is to say, the rule in KB expresses that whenever the pilot agent believes that there is an EFTO, then it is believed that either the thrust is accelerating or the aircraft is ascending (i.e., the thrust cannot accelerate unless descending after engine failure).

With the intention of obtaining complete detailed belief-based explanations of the agent behaviour, we need to apply the state update operator over each pair of actions in the behaviour log repeatedly, with the previous updated belief-based explanations associated with the former action as the first argument and the current enabling beliefs associated with the later action as the second argument.

4 Mining Belief-Based Explanations

Mining belief-based explanations starts with transforming the observations in the audit logs into observation sequences, each of which involves the execution of an action and the manifestations of its enabling beliefs. Note that logging tools are usually designed to record one mode of observation per log. That is to say, action execution and external events manifestation are recorded in separate logs. To that end, we also need to start a correlation between the two logs to obtain an observation log that serves as a sequential database, which will be mined to extract belief-based explanations using a sequential rule miner. We define this correlation as follows.

Definition 2 (Observation sequence, and log). Let A be an actions space, B be a belief states space, $\mathsf{a}_0, \ldots, \mathsf{a}_n \in \mathsf{A}$, and $\mathsf{b}_0, \ldots, \mathsf{b}_n \in \mathsf{B}$. An observation instance (t) is an alternating sequence of the form $\mathsf{b}_0, \mathsf{a}_1, \ldots, \mathsf{b}_n, \mathsf{a}_n$. $\mathsf{D}_{\mathsf{All}}$ is an observation log, such that $\mathsf{D}_{\mathsf{All}} \in 2^{\mathsf{T}}$, where T is the set of all observation instances.

Mining belief-based explanations relies on the two following premises: (1) that the beliefs observed in the belief log immediately before executing an action can be the enabling beliefs of that action, and (2) that the persistent beliefs observed a long time before the execution of an action are typically not the enabling beliefs of that action, but may be of that action plus some others. Hence, we use the basic relation "direct successor" [17] over the actions and beliefs in the $\mathsf{D}_{\mathsf{All}}$ as follows:

Definition 3 (Direct successor). Let $\mathsf{D_{All}}$ be an observation sequence over T. Let $\mathsf{b}, \mathsf{a}, \mathsf{b}' \in \mathsf{T}$.

1. Direct predecessor state: $\mathsf{b} >_{\mathsf{D}} \mathsf{a}$ *iff* $\langle \mathsf{b}, \mathsf{a} \rangle$ is a subsequence of T, and
2. Direct successor state: $\mathsf{a} >_{\mathsf{D}} \mathsf{b}'$ *iff* $\langle \mathsf{a}, \mathsf{b}' \rangle$ is a subsequence of T.

Relation $>_{\mathsf{D}}$ describes which external events directly follow/precede a given action. Direct predecessor relation over $\mathsf{D_{All}}$ would offer learning entries of the form $\langle \mathsf{b}, \mathsf{a} \rangle$, where a is an action applicable at the belief state b. For our purposes, we do not distinguish between $\langle \mathsf{q}, \mathsf{p}, \mathsf{a} \rangle$ and $\langle \mathsf{p}, \mathsf{q}, \mathsf{a} \rangle$, because we are only interested in relating actions to their enabling beliefs but not in the relation amongst enabling beliefs. Against this background, we create $\mathsf{O_{Direct}}$, which is a sequence of the following form

$$\langle \langle \langle \mathsf{b_{11}}, .., \mathsf{b_{1n}} \rangle, \mathsf{a_1} \rangle \rangle, .., \langle \langle \mathsf{b_{i1}}, .., \mathsf{b_{im}} \rangle, \mathsf{a_i} \rangle \rangle, .., \langle \langle \mathsf{b_{p1}}, .., \mathsf{b_{pk}} \rangle, \mathsf{a_p} \rangle \rangle \rangle$$

where each $\langle \mathsf{a_{i-1}}, \mathsf{a_i} \rangle$ represents an ordered pair of actions, and each $\langle \mathsf{b_{i1}}, .., \mathsf{b_{im}} \rangle$ represents the observed beliefs before the execution of action $\mathsf{a_i}$ and after action $\mathsf{a_{i-1}}$ execution. An exception is required for the first recorded action in $\mathsf{D_{All}}$, as there is no preceding action. In this case, we use the timestamp of the initial high-level event as the start time of the system execution. Given $\mathsf{D_{All}}$, we view the problem of mining belief-based explanations as finding all the sequences $\langle \mathsf{b}, \mathsf{a} \rangle$ that satisfy some predefined measures of interestingness, assuming unique activity execution (i.e., there is no concurrent execution of actions).

Again, we are interested in discovering all the beliefs that are observed always, or most of the time, directly before the execution of each action referred to in the behaviour log. Association rule learning can be an effective means for discovering regularities between beliefs and actions. Fundamentally, given two itemsets X and Y, the sequential rule $\mathsf{X} \rightarrow \mathsf{Y}$, states that if the elements in X occur, then it will be followed by the elements in Y. For a sequential rule to be an interesting one, it must satisfy minimum support (how frequently the rule appears in $\mathsf{D_{All}}$) and minimum confidence (the accuracy of the rule). Such rules are considered in our context as belief-based explanations. We use the well-known CMRules algorithm [18] to discover this form of rules.

Example 4. Continuing with our running example, Table 3 shows the results of applying the CMRules algorithm to $\mathsf{D_{All}}$ obtained from Table 1 and Table 2.

CMRules algorithm can discover all the interesting association rules from $\mathsf{D_{All}}$. Nevertheless, additional post-processing is required, where we rule out any association rule that its consequent label is not a single action name or its antecedent label is not beliefs.

Table 3. Mined belief-based explanations

plan	action	belief-based explanations
p1	idle(throttle)	runway(dry) $\wedge$ wind(cross) $\wedge$ efto $\wedge$ Flaps $=$ 15 $\wedge$ V1 $>$ Speed $>$ V2
p1	deploy(brakes)	decelerating(thrust)
p1	send(tower, msg)	steady(aircraft)
p2	increase(mixture)	runway(wet) $\wedge$ wind(head) $\wedge$ efto $\wedge$ Flaps $=$ 15 $\wedge$ V1 $>$ Speed $>$ V2
p2	increase(throttle)	escalating(fuel_flow)
p2	take_up(flap)	accelerating(thrust)
p2	pull(yoke)	liftoff(aircraft)

5 Validating the Explanation Process

Our guiding intuition here is that the state update operator and the available data used to update belief-based explanations can also be leveraged to validate the mined explanations. Our validation approach involves some mechanisms that take as inputs (1) A $\mathsf{D_{All}}$, (2) A state update operator, and compute the soundness and completeness of the mind explanations. We keep assuming unique action execution through this section.

First, we need inputs (1) and (2) to generate a sequence (denoted hereafter as $\mathsf{D_{Updated}}$) that associates each action in $\mathsf{D_{All}}$ to the set of updated beliefs of all actions executed up to that point. Each object in $\mathsf{D_{Updated}}$ is a pair of the form $\langle \beta_\mathsf{i}, \mathsf{a_i} \rangle$, where β the set of updated beliefs of all actions executed up to $\mathsf{a_i}$. $\mathsf{D_{Updated}}$ can be simply obtained using $\oplus$ over $\mathsf{D_{All}}$ assuming the presence of a KB defined in the same language as that in which the beliefs are described. Table 4 illustrates the results of applying the operator $\oplus$ to input $\mathsf{D_{All}}$.

Table 4. Updated belief-based explanations of plan p1

plan	action	belief-based explanations
p1	idle(throttle)	runway(dry) $\wedge$ wind(cross) $\wedge$ efto $\wedge$ Flaps $=$ 15 $\wedge$ V1 $>$ Speed $>$ V2
p1	deploy(brakes)	decelerating(thrust) $\wedge$ runway(dry) $\wedge$ wind(cross) $\wedge$ efto $\wedge$ Flaps $=$ 15 $\wedge$ V1 $>$ Speed $>$ V2
p1	send(tower, msg)	decelerating(thrust) $\wedge$ steady(aircraft) $\wedge$ runway(dry) $\wedge$ wind(cross) $\wedge$ efto Flaps $=$ 15 $\wedge$ V1 $>$ Speed $>$ V2

It should be noted that a single action in $\mathsf{D_{Updated}}$ can be associated with a set of sets of beliefs due to the non-determinism nature of the $\oplus$ operator. Now, to validate the mined explanations, it is useful to establish the following:

– **Soundness**. A sound belief-based explanation is one that is mined correctly (i.e., observed in $\mathsf{D_{All}}$). Another way to say that a detailed belief-based explanation to a given point in the plan execution should contain the mined belief-based explanation updated using $\oplus$ at that point in the plan execution. Formally, for each plan execution sequence in $\mathsf{D_{Updated}}$ and each action $\mathsf{a_i}$ the following condition must hold: $\beta_i \cup \mathsf{KB} \models \mathsf{b}$ for some $\mathsf{b} \in \mathsf{b_{a_1}}, \ldots, \mathsf{b_{a_i}}$, where $\mathsf{b_{a_i}}$ is the mined belief-based explanation of action $\mathsf{a_i}$.
– **Completeness**. A complete belief-based explanation requires that all the enabling beliefs of a given action are mined. Actually, this can be viewed as a reversal of the above-cited entailment relation (i.e., $\mathsf{b} \cup \mathsf{KB} \models \beta_i$)

Where it is possible for the mined explanations to be unsound or incomplete, further post-processing may be required for more reliable results. We overtake this problem by seeking more observations and/or re-mining with lower support and confidence thresholds.

6 Evaluation

Agent explanation generation can be evaluated on several grounds. Commonly, the first is a human-rated evaluation of the explainability of the generated explanations. However, we consider this more as a part of explainable agent development, so our focus in this section is on evaluating our mining technique, which includes the performance measures of the mined explanations (i.e., precision and recall).

This section presents the evaluation of our approach to mining belief-based explanations. First, we present the setup and implementation for our experiments. Next, our evaluation is detailed.

6.1 Data and Implementation

We implemented our approach as a plugin for our Toolkit XPlaM[3] [19] and evaluated it using a synthetic log of 1000 execution instances, representing firefighter agent past behaviour as described in [3]. We also used a plan library consisting of 15 plans[4].

[3] The source code for XPlaM Toolkit (including the code for the approach presented here) has been published online at https://github.com/dsl-uow/xplam.

[4] We published the datasets supporting the conclusions of this work online at https://www.kaggle.com/datasets/alelaimat/explainable-bdi-agents.

6.2 Performance Results

Our goal of the evaluation is to establish that the proposed approach can generate generally reliable explanations. To that end, we recorded the precision (i.e., number of correctly mined explanations over the total number of mined explanations) and recall (i.e., number of correctly mined explanations over the total number of actual explanations) obtained from applying the CMRules algorithm and the above-cited validation technique. We consider minimum confidence (*min conf*) and minimum support (*min supp*) of the mined rules as the most important factors for mining belief-based explanations. The results are depicted in Table 5 and are summarized below.

Table 5. Performance for different *min conf* and *min supp*

min conf	1.00	0.95	0.9	0.85	0.8
Precision	0.8948	0.8910	0.8731	0.8487	0.8235
Recall	0.5678	0.6387	0.7265	0.7854	0.8547
min supp	0.5	0.4	0.3	0.2	0.1
Precision	0.9216	0.8741	0.8254	0.7703	0.6987
Recall	0.2257	0.3458	0.5361	0.6966	0.8815

min conf:

1. Confidence threshold considerably impacts performance results, i.e., higher *min conf* leads to higher precision but lower recall. For example, our approach achieved the highest precision of 0.89 with the lower recall of 0.56 for *min conf* of 1.00.
2. It is necessary, thus, to find a trade-off between precision and recall when varying *min conf* threshold. Hence, we use *min conf* of 0.9 for testing the impact of varying *min supp* threshold.

min supp

1. Similar to the *min conf*, varying *min supp* has a significant impact on the performance results. For example, we achieved the highest precision of 0.92 with an insignificant recall of 0.22 for *min supp* of 0.5.
2. Note that extracting association rules related to infrequent events needs *min supp* to be low. However, this, in turn, sacrifices the precision of the mined explanations.

We noticed some insightful observations through our experiments. First, the size of $\mathsf{D}_{\mathsf{All}}$ does not impact the performance results of the mined explanations *iff* $\mathsf{D}_{\mathsf{All}}$ includes all possible behaviours of the observed agent. For example, we achieved very close precision values for $\mathsf{D}_{\mathsf{All}}$ of 200 and 800 execution instances, which were 0.836 and 0.842, respectively. Second, varying *min conf* and *min supp* have a significant impact on the performance of the mined explanations and, consequently, come with several limitations (e.g., time and effort). Finally, the premise that the beliefs observed in the belief log immediately before executing an action can be the enabling beliefs of that action can be deceptive sometimes (i.e., the enabling beliefs of a late action can hold before plan execution). It is necessary, therefore, for the user to determine precise *min conf* and *min supp* thresholds.

7 Related Work

Although a large and growing body of work exists on developing explainable agents and robots [1], few works focus on generating explanations for intelligent agents.

Harbers et al. [3] described four algorithms to design explainable BDI agents: one using parent goals, one using top-level goals, one using enabling beliefs, and one using the next action or goal in the execution sequence. They found that goal-based explanations were slightly preferable to belief-based expansions to explain procedural actions (i.e., a sequence of actions and sub-goals) based on users' evaluation. Nevertheless, belief-based explanations were preferable in explaining conditional and single actions. Similar explanation algorithms were proposed by Kaptein et al. [2], but to investigate the difference in preference of adults and children for goal-based and belief-based explanations. They found that both adults and children preferred goal-based explanations. Related, but in a different ontology, is the work presented in Kaptein et al. [20], in that the agent explains its action in terms of its beliefs, goals and emotions.

Sindlar et al. [21] proposed an abductive approach to infer the mental states of BDI agents in terms of beliefs and goals. To that end, they described three explanatory strategies under three perceptory presumptions: complete, late, and partial observations. Sindlar et al. extend their work to an explanation approach that takes into account three organizational principles: roles, norms, and scenes in [22]. The extended work proposes an approach to how the observed behaviour of game players can be explained and predicted in terms of the mental state of virtual characters. Related, but in a different domain, is the work presented in [23]. Sequeira and Gervasio propose a framework for explainable reinforcement learning that extracts relevant aspects of the RL agent interaction with its environment (i.e., interestingness elements) in [23]. They suggested four dimensions of analysis: frequency, execution certainty, transition-value, and sequence analysis to extract the interestingness elements, which are used to highlight the behaviour of the agent in terms of short video clips.

All the previous approaches presented in this paper can theoretically generate explanations of agents' actions, but assuming reliable observations, availability of an explanation generation module and deterministic execution of plans. On the other hand, we leverage the past execution experiences of the target agent, which allows performing various techniques to handle unreliable observations (e.g., measures of interestingness). Although historical data might be hard to be obtained, our approach still can be effective in settings where no more than common patterns (i.e., plans), information that other players would observe or expect the target agents to be aware of (i.e., external events) and the external behaviour of the target agent are available. Much of the work done on agent explanation generation shares the common judgment that relatively short explanations are more useful to explainees. Nevertheless, as shown in our running example, detailed explanations are critical in some cases (e.g., explaining BDI plan selection). Finally, and in contrast with the literature, our explanation generation mechanism allows explanations to be non-deterministic through the updating of belief-based explanations, as described in Sect. 3.

8 Conclusion

In this paper, we addressed the problem of agent explanation mining (and specifically belief-based explanations) in the context of the well-known BDI paradigm. This problem was formulated as follows: "Given the past execution experiences of an agent and an update operator, generate the belief-based explanation of each action referred to in the agent's past execution experiences". We presented an update operator that is able to generate detailed explanations. Through examples, we also showed that detailed explanations could be useful in explaining BDI plan selection. We have tackled the problem of explanation generation in non-deterministic settings. At this point in time, we are trying to extend the application of the proposed approach to other BDI handles.

References

1. Anjomshoae, S., Najjar, A., Calvaresi, D., Främling, K.: Explainable agents and robots: results from a systematic literature review. In: 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, 13–17 May 2019, pp. 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems (2019)
2. Kaptein, F., Broekens, J., Hindriks, K., Neerincx, M.: Personalised self-explanation by robots: the role of goals versus beliefs in robot-action explanation for children and adults. In: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 676–682. IEEE (2017)
3. Harbers, M., van den Bosch, K., Meyer, J.-J.: Design and evaluation of explainable bdi agents. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 125–132. IEEE (2010)
4. Abdulrahman, A., Richards, D., Bilgin, A.A.: Reason explanation for encouraging behaviour change intention. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pp. 68–77 (2021)
5. Georgeff, M., Rao, A.: Modeling rational agents within a bdi-architecture. In: Proceedings of 2nd International Conference on Knowledge Representation and Reasoning (KR 1991), pp. 473–484. Morgan Kaufmann (1991)
6. Harbers, M., van den Bosch, K., Meyer, J.-J.C.: A study into preferred explanations of virtual agent behavior. In: Ruttkay, Z., Kipp, M., Nijholt, A., Vilhjálmsson, H.H. (eds.) IVA 2009. LNCS (LNAI), vol. 5773, pp. 132–145. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04380-2_17

7. Calvaresi, D., Mualla, Y., Najjar, A., Galland, S., Schumacher, M.: Explainable multi-agent systems through blockchain technology. In: Calvaresi, D., Najjar, A., Schumacher, M., Främling, K. (eds.) EXTRAAMAS 2019. LNCS (LNAI), vol. 11763, pp. 41–58. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30391-4_3
8. Verhagen, R.S., Neerincx, M.A., Tielman, M.L.: A two-dimensional explanation framework to classify AI as incomprehensible, interpretable, or understandable. In: Calvaresi, D., Najjar, A., Winikoff, M., Främling, K. (eds.) EXTRAAMAS 2021. LNCS (LNAI), vol. 12688, pp. 119–138. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-82017-6_8
9. Mualla, Y.: Explaining the behavior of remote robots to humans: an agent-based approach. PhD thesis, Université Bourgogne Franche-Comté (2020)
10. Winikoff, M.: Jack^{TM} intelligent agents: an industrial strength platform. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming. MSASSO, vol. 15, pp. 175–193. Springer, Boston, MA (2005). https://doi.org/10.1007/0-387-26350-0_7
11. Abdulrahman, A., Richards, D., Ranjbartabar, H., Mascarenhas, S.: Belief-based agent explanations to encourage behaviour change. In: Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents, pp. 176–178 (2019)
12. Multi-engine aeroplane operations and training. Technical report, Civil Aviation Safety Authority (July 2007)
13. Bordini, R,H., Hübner, J.M., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons (2007)
14. Howden, N., Rönnquist, R., Hodgson, A., Lucas, A.: Jack intelligent agents-summary of an agent infrastructure. In: 5th International Conference on Autonomous Agents, vol. 6 (2001)
15. Ginsberg, M.L., Smith, D.E.: Reasoning about action i: a possible worlds approach. Artifi. intell. **35**(2), 165–195 (1988)
16. Winslett, M.S.: Reasoning about action using a possible models approach, pp. 1425–1429. Department of Computer Science, University of Illinois at Urbana-Champaign (1988)
17. Maruster, L., Weijters, A.J.M.M.T., van der Aalst, W.M.P.W., van den Bosch, A.: Process mining: discovering direct successors in process logs. In: Lange, S., Satoh, K., Smith, C.H. (eds.) DS 2002. LNCS, vol. 2534, pp. 364–373. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36182-0_37
18. Fournier-Viger, P., Faghihi, U., Nkambou, R., Nguifo, E.M.: Cmrules: mining sequential rules common to several sequences. Knowl.-Based Syst. **25**(1), 63–76 (2012)
19. Alelaimat, A., Ghose, A., Dam, K.H.: Xplam: a toolkit for automating the acquisition of BDI agent-based digital twins of organizations. Comput. Industry **145**, 103805 (2023)
20. Kaptein, F., Broekens, J., Hindriks, K., Neerincx, M.: The role of emotion in self-explanations by cognitive agents. In: 2017 Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW), pp. 88–93. IEEE (2017)
21. Sindlar, M.P., Dastani, M.M., Dignum, F., Meyer, J.-J.C.: Mental state abduction of BDI-based agents. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 161–178. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-93920-7_11

22. Sindlar, M.P., Dastani, M.M., Dignum, F., Meyer, J.-J.C.: Explaining and predicting the behavior of BDI-based agents in role-playing games. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS (LNAI), vol. 5948, pp. 174–191. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11355-0_11
23. Sequeira, P., Gervasio, M.: Interestingness elements for explainable reinforcement learning: understanding agents' capabilities and limitations. Artif. Intell. **288**, 103367 (2020)