Explaining BDI Agent Behaviour Through Dialogue

Louise A. Dennis
Department of Computer Science
University of Manchester
louise.dennis@manchester.ac.uk

Nir Oren
Department of Computing Science
University of Aberdeen
n.oren@abdn.ac.uk

ABSTRACT

BDI agents act in response to external inputs and their internal plan library. Understanding the root cause of BDI agent action is often difficult, and in this paper we present a dialogue based approach for explaining the behaviour of a BDI agent. We consider two dialogue participants who may have different views regarding the beliefs, plans and external events which drove agent action (encoded via traces). These participants make utterances which incrementally reveal their traces to each other, allowing them to identify divergences in the traces, or to conclude that their traces agree. In practice, we envision a human taking on the role of a dialogue participant, with the BDI agent itself acting as the other participant. The dialogue then facilitates explanation, understanding and debugging of BDI agent behaviour. After presenting our formalism and its properties, we describe our implementation of the system and provide an example of its use in a simple scenario.

KEYWORDS

BDI; Dialogues; Explanation

ACM Reference Format:
Louise A. Dennis and Nir Oren. 2021. Explaining BDI Agent Behaviour Through Dialogue. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 8 pages.

1 INTRODUCTION

Belief, Desire Intention (BDI) based approaches to agent reasoning are very popular, with applications ranging from air traffic management [**?**], to e-Health [**?**]. The formal basis of BDI systems facilitate formal validation and verification, and provide guarantees as to their actions [**?**] and to the states the system will, or will not reach. However, understanding why a BDI-based system acted as it did is difficult, requiring working through plans and subplans while tracking the system's internal state.

Researchers have noted that dialogue is a potentially useful tool to explain the behaviour of complex AI artefacts [**?**], and in this paper we propose a dialogue based approach to reasoning about BDI system behaviour. As our departure point, we consider the case where two dialogue participants (which we may also refer to as agents) hold — possibly different — views about the content of a BDI program and the environment in which it executes. Our dialogue is then designed to pinpoint where disagreement between dialogue participants exists. Such disagreement could, for example, lie in different views regarding what plans drive the BDI system; their priorities; or differences in inputs or initial beliefs of the system. Our dialogue then enables at least one dialogue participant to locate a disagreement (if one exists); and alternatively allows it to determine if no disagreement exists. We assume that in the case of multiple disagreements the dialogue can take place multiple times, with the dialogue participants' beliefs updated between each dialogue instance. Importantly, we do not consider how participants update their beliefs during or following a dialogue, with such belief revision lying outside the scope of the current work.

Our main contribution is the description and formalisation of the explanatory dialogue, enabling the identification of, and explanation for, the reasons why a BDI system behaved as it did. We focus on this formal aspect here and emphasise that natural language generation and utterance presentation from similar work (e.g., [**?** **?**]) lies outside the scope of this paper. Unlike e.g., [**?** **?**] we do not provide a formal argumentation-based underpinning to our dialogue.

Section 2 introduces a simple BDI language and formalises our environment. Section 3 introduces the dialogue. We examine the properties of the dialogue in Section 4 and provide an illustrative example in Section 5. Section 6 contains detailed discussion, including a comparison with existing work. Section 7 concludes by considering avenues for future research.

2 THE SIMPLEBDI LANGUAGE

We introduce a very simple BDI language, SimpleBDI, which we will use to illustrate our ideas. SimpleBDI is designed to be as simple as possible, as its primary purpose is to demonstrate the feasibility of our approach and enable its formalisation. SimpleBDI contains the constructs which lie at the heart of more complex BDI languages, and is therefore an appropriate underlying representation.

A SimpleBDI program consists of an ordered list of plans Π of the form $\pi_{id} : B \rightarrow I$. B (the plan's *guard*) is a set of first order ground predicates over some language $\mathcal{L}$, and I is a $[U, do(a)]$ pair. In turn, U is a set of *belief updates* of the form $+b, -b$ where b is a ground first order predicate, and a is an action, again denoted using a ground first order predicate. Since some plans may only update beliefs rather than execute an action, we introduce a special symbol *null* to denote the lack of action. In addition, we assume the existence of an *empty plan* $\pi_{null} : [] \rightarrow [[], do(null)]$. Plans within Π are assumed to be ordered by preference, and we write $\pi > \pi'$ if π is preferable to π' (i.e., iff the index of π is smaller than the index of π' in Π). Unlike most BDI languages, SimpleBDI does not explicitly model goals. However, goals can be encoded through the introduction of a predicate of the form $goal(g)$, which is added and removed as a belief at appropriate times as part of plan execution.

SimpleBDI programs execute plans based on beliefs and changes in the environment (percepts). The latter is captured by an *input trace* τ_e of events external to the agent. Each event is a list of belief updates, V, of the form $+b$ or $-b$ containing a single ground first

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

order predicate. The list of belief updates for a single event cannot contain contradictory belief updates, i.e., $+b, -b \notin V$.

The executor[1] of a SimpleBDI program maintains a set of internal beliefs — denoted $\mathcal{B}$ — encoded as a set of ground first order predicates, and is formally represented at a point in time as a tuple

$$E = \langle \mathcal{B}, \Pi, \pi, \tau_e, a_{ex}, stage \rangle$$

Here, $\mathcal{B}$ is a set of the executor's beliefs; Π is its plan library; π the current plan selected for execution; τ_e is the input trace; and a_{ex} the (external) action executed by the executor agent at that time. $stage \in \{\mathsf{s}, \mathsf{p}, \mathsf{e}\}$ captures the current state of the executor; SimpleBDI programs run through repeated perception (p), plan selection (s), and plan execution (e) stages. Given a set of plans Π representing a SimpleBDI program and an input trace τ_e the initial state of the executor is

$$E = \langle [], \Pi, null, \tau_e, null, \mathsf{p} \rangle$$

Figure 1 summarises the semantics of SimpleBDI describing how the tuple representing the executor evolves as a transition system (i.e., if $E_i \rightarrow E_{i+1}$ in Fig. 1 then E_i becomes E_{i+1} as the system executes). A *program execution trace* is then the sequence of tuples $[E_1, \ldots, E_n]$ where E_{i+1} is obtained by executing a program over the input trace found in E_i until the input trace is empty; the $\emptyset$ symbol denotes the end of program execution.

In the perception phase (p), the top of the input trace (τ_e) is consumed, updating the set of beliefs $\mathcal{B}$. The update itself is done through the *update* function, which takes a set of belief updates and a set of beliefs as input, and returns an updated set of beliefs. Note that during the remaining phases, no beliefs are consumed; a *null* perception is therefore consumed during these phases.

The plan selection phase (s) proceeds by selecting an *applicable* plan using the *select* and *applicable* functions respectively. The former iterates through the plan library in order of preference, while the latter deems a plan applicable by checking whether the plan's beliefs do, or do not, appear in the belief base. If no applicable plan is found, then the empty plan π_{null} is returned. The selected plan is recorded, to be used in the next phase.

Finally, the plan execution phase (e) takes the selected plan and updates the belief base according to the plan's effects. In addition, any action a executed due to the plan is recorded.

Example 1. *Listing 1 shows a simple program in SimpleBDI (i.e., the plans,* Π*, used by the program executor). In this program a robotic system (for instance a Mars Rover), must move from its starting position to a waypoint and then on to a final location to take a sample. It does this if it believes it has received a message take_sample_message. It then uses move1 to move from the starting point to the waypoint (if it believes the terrain is safe) and then uses move2 to move from the waypoint to the location where it should take the sample by drilling (again if it believes the terrain is safe). We omit "do(null)" for plans with no associated actions.*

Code Listing 1

```
take_sample_message ->                                  1
    -take_sample_message , +goal_at_location            2
                                                        3
safe_terrain , at_start , goal_at_location ->           4
    +at_waypoint , -at_start ,                          5
    do(move1)                                           6
                                                        7
safe_terrain , at_waypoint , goal_at_location ->        8
    +at_location , -at_waypoint ,                       9
    -goal_at_location , +goal_take_sample ,             10
    do(move2)                                           11
                                                        12
at_location , goal_take_sample ->                       13
    -goal_take_sample ,                                 14
    do(drill)                                           15
```

So for instance, the second plan (lines 5-8) states that if the agent perceives that it is at the start, and the terrain is safe and it has received a message telling it to take a sample then it will move to the waypoint (via the external action `move1`*). At this point, if it no longer perceives the terrain is safe it will not move further. However if it continues to believe the terrain is safe it will move to the final location (using the plan in lines 10-15) and which in turn triggers the remaining plan in the program: to drill for a sample (lines 17-19).*

Given a program and an initial state — which includes an input trace — the state of the executor at each step of the program execution trace describes the internal state of the executor and its effects (actions) on the environment.

3 DIALOGUES

The semantics of SimpleBDI allow us to determine how a program will execute (for a given initial state). However, systems executing such programs are often opaque. The aim of this paper is to help facilitate an understanding of program behaviour.

To this end we consider a dialogue between two participants who may have partial access to the program execution trace, and have their own model of the executor. Our dialogue seeks to identify differences between the participants' models so as to identify disagreements. Such differences could arise due to differences in the plans the dialogue participants believe the executor has; a divergence with regards to the beliefs they believe the executor holds; or different beliefs they have with regards to the various traces. If one of the participants is the program executor (whose trace is correct), and another is a human or system trying to understand the executor's behaviour, then identifying a disagreement means that an error in the latter's assumptions or reasoning has been identified, and doing so serves as a form of explanation of the executor's behaviour.

We begin by providing the intuition behind our dialogue, after which we describe a model of the dialogue participants. Finally, we formalise the dialogue by describing the utterances participants may make in the dialogue (c.f., dialogue games [**?**]).

3.1 Dialogue — Intuitions

When applying the semantics correctly, differences between execution traces for dialogue participants arise due to differing plans within agent plan libraries or plan precedence, or due to different perceptions from the environment.The only externally visible effects of a running system are the actions it executes, and our dialogue therefore begins by having one participant asking the other why, or why not, an action was performed at some time.

[1]We use the term "executor" rather than agent to differentiate this entity from the agents undertaking dialogue about the execution of the SimpleBDI program.

$$\langle \mathcal{B}, \Pi, \pi, [], a_{ex}, \mathsf{p}\rangle \rightarrow \emptyset$$
$$\langle \mathcal{B}, \Pi, \pi, [V|\tau_e], a_{ex}, \mathsf{p}\rangle \rightarrow \langle update(V, \mathcal{B}), \Pi, \pi, \tau_e, null, \mathsf{s}\rangle$$
$$\langle \mathcal{B}, \Pi, \pi, [null|\tau_e], a_{ex}, \mathsf{s}\rangle \rightarrow \langle \mathcal{B}, \Pi, select(\mathcal{B}, \Pi), \tau_e, null, \mathsf{e}\rangle$$
$$\langle \mathcal{B}, \Pi, \pi_{id} : B \rightarrow [U, do(a)], [null|\tau_e], a_{ex}, \mathsf{e}\rangle \rightarrow \langle update(U, \mathcal{B}), \Pi, \pi_{id} : B \rightarrow [U, do(a)], \tau_e, a, \mathsf{p}\rangle$$

$$update([], \mathcal{B}) = \mathcal{B}$$
$$update([+b|B], \mathcal{B}) = \mathcal{B} \cup \{b\}$$
$$update([-b|B], \mathcal{B}) = \mathcal{B} \backslash \{b\}$$

$$select(\mathcal{B}, []) = \pi_{null}$$
$$select(\mathcal{B}, [\pi_{id} : B \rightarrow I|\Pi]) = \begin{cases} \pi_{id} & \text{if } applicable(\mathcal{B}, B) \\ select(\mathcal{B}, \Pi) & \text{otherwise} \end{cases}$$

$$applicable(\mathcal{B}, []) = \top$$
$$applicable(\mathcal{B}, b : B) = \begin{cases} applicable(\mathcal{B}, B) & \text{if } b \in \mathcal{B} \\ \bot & \text{otherwise} \end{cases}$$
$$applicable(\mathcal{B}, \neg b : B) = \begin{cases} applicable(\mathcal{B}, B) & \text{if } b \notin \mathcal{B} \\ \bot & \text{otherwise} \end{cases}$$

Figure 1: SimpleBDI semantics. $\emptyset$ denotes termination of execution.

Let us consider the evolution of a possible explanatory dialogue. If a dialogue participant asks another why an action did not take place, the latter can respond by asking the former why they believe an action did take place. If on the other hand, a participant asks why an action did take place, the explanation (i.e., response) involves identifying the (executed) plan which triggered the action. When asked why a plan was executed, the response involves demonstrating that the set of beliefs which triggered the plan held. When asked why some belief held, a response involves either presenting the percept which caused the belief, or the plan which led to the belief being adopted. In the latter case, the dialogue can continue by providing an explanation for the plan.

When an assertion regarding a belief is presented it is also possible for a disagreement to occur, with the other dialogue participant asserting that the belief does not hold at the relevant point in time. In such a situation, the dialogue can continue with the presentation of a plan or percept which removes the belief. In the former case, the dialogue can continue by providing an explanation for the plan. In the latter case, the presentation of the percept should identify a disagreement between the dialogue participants.

The above paths through the dialogue help us identify natural points of dialogue termination. When a percept justifying a belief is presented, no further explanation is possible, as such a percept originates from outside the BDI system. When stating that a plan was executed, if the other dialogue participant is not aware of the plan (i.e., the plan is not present in their plan library), or if they believe that a higher precedence plan exists, then a disagreement has been identified which cannot be resolved by further discussion regarding system execution. Finally, If one dialogue participant asks another why an action took place (or didn't take place), and the latter believes that the action didn't take place (did take place), then no further discussion is possible.

3.2 Dialogue Participant Model

A dialogue participant is a tuple $\langle \mathcal{M}, \mathcal{O}, \overline{\mathcal{O}}\rangle$ where $\mathcal{M}, \mathcal{O}$ and $\overline{\mathcal{O}}$ are program execution traces of a BDI program, and $|\mathcal{M}| = |\mathcal{O}| = |\overline{\mathcal{O}}|$.

Informally $\mathcal{M}$ represents the participants model of what *should* have happened – i.e., the program execution trace they believe to be correct, $\mathcal{O}$ represents their (partial) understanding of what the other participant's trace looks like. $\overline{\mathcal{O}}$ then captures commitments or constraints that emerge on the other participant due to their utterances – specifically plans the other participant has explicitly committed to *not* having been selected; beliefs explicitly committed to *not* having been perceived on the input trace; and actions explicitly committed to *not* having been performed[2].

We index a specific time point within the execution trace using array notation (e.g., $\mathcal{M}[5]$). Where the context is clear, we index individual portions of a BDI executor's state at a specific time in the same manner, identifying the program with a superscript. For example, $a_{ex}^{M}[5]$ refers to a_{ex} of BDI program execution trace M at time 5. Equivalently, if — for example — some $x \in \tau_e^M[5]$, we may say that x holds in τ_e^M at time 5. We refer to elements within $\overline{\mathcal{O}}$ as $\overline{\mathcal{B}}, \overline{\pi}$ etc, indexing individual entries by time. We note that — in the present system — the plan library Π does not change, and therefore abuse notation by referring to it without identifying a specific time point; we assume that any operations on $\Pi^{\mathcal{M}}, \Pi^{\mathcal{O}}$ and $\overline{\Pi}$ apply to all time indices. Finally, we also assume that $stage^{\mathcal{M}}[T] = stage^{\mathcal{O}}[T]$ for all $0 \leq T < |\mathcal{M}|$.

Utterances made by one dialogue participant can affect the other participant's view of the utterer. Therefore, given one dialogue participant $\langle \mathcal{M}, \mathcal{O}, \overline{\mathcal{O}}\rangle$, we refer to the other dialogue participant as $\langle \mathcal{M}', \mathcal{O}', \overline{\mathcal{O}}'\rangle$. We can, for example, index the other dialogue participant's view of its own input trace at time T as $\tau_e'^{\mathcal{M}}[T]$.

The purpose of our dialogue is to allow a participant to identify disagreements or inconsistencies between itself and the other participant. Such disagreements can be recognised as occurring between the $\mathcal{M}$ and $\mathcal{O}$ traces, or between the $\mathcal{M}$ and $\overline{\mathcal{O}}$ traces.

- For an index T, if $\mathcal{B}^{\mathcal{O}}[T] \not\subseteq \mathcal{B}^{\mathcal{M}}[T]$, or if $\overline{\mathcal{B}}[T] \cap \mathcal{B}^{\mathcal{M}}[T] \neq \emptyset$ then a disagreement in belief has been identified. More specifically, the disagreement rests on beliefs $(\mathcal{B}^{\mathcal{O}}[T] \backslash \mathcal{B}^{\mathcal{M}}[T]) \cup (\overline{\mathcal{B}}[T] \cap \mathcal{B}^{\mathcal{M}}[T])$.
- If $\Pi^{\mathcal{O}} \not\subseteq \Pi^{\mathcal{M}}$, or if $\pi > \pi'$ according to $\Pi^{\mathcal{M}}$, and $\pi \not> \pi'$ according to $\Pi^{\mathcal{O}}$, or if there is some $\pi \in \overline{\Pi}, \Pi^{\mathcal{M}}$ then a

[2]We observe that some elements of the tuples stored within $\overline{\mathcal{O}}$ are therefore not used within our system.

disagreement w.r.t. the plan library has been identified. Such a dispute revolves around plans π, π' or π respectively.
- For an index T, if $\pi^O[T], \overline{\pi}[T] \neq null$ and $\pi^{\mathcal{M}}[T] \neq \pi^O[T]$ or $\pi = \overline{\pi}[T]$ then a disagreement in the executing plan has been identified. This dispute centers on plan π.
- If at any time T the head of $\tau_e^{\mathcal{M}}$ differs from the head of τ_e^O and $\tau_e^O \neq null$, then a disagreement in perception (w.r.t. the respective heads of the lists) has been identified.
- Finally, if at any time T, $a_{ex}^{\mathcal{M}}[T] \neq a_{ex}^O[T] \neq null$ or $a_{ex}^{\mathcal{M}}[T] = \overline{a_{ex}}[T]$, then a disagreement in action has been identified, based on the actions identified.

Note that in the above, for a disagreement to occur, the relevant element of O and $\overline{O}$ must not be $\emptyset$. This reflects the fact that as the dialogue progresses, O and $\overline{O}$ are updated and a disagreement only occurs when an explicit difference is found.

3.3 Dialogue Initiation and Termination

At the start of a dialogue all elements of O and $\overline{O}$ at all times are set to *null* ($\emptyset$ for beliefs), reflecting a lack of knowledge a dialogue participant has about the other participant.

The dialogue begins when one dialogue participant makes a $why(A, T)$ or $why(\neg A, T)$ move, asking why an action, A was, or was not performed at time T.

The dialogue continues as additional utterances are made by the participants in response to previous utterances. Utterances are *open* until their *closure condition* occurs in the dialogue, at which point they are *closed*. The dialogue terminates when no open utterances exist, i.e., when there is no legal move that any dialogue participant can make. Once the dialogue terminates, disagreement(s) can be identified using the procedure described in the previous section.

3.4 Utterances

During a dialogue, participants make different utterances (a.k.a. *moves*). Table 1 describes these, when they can be made, and their intuitive meaning, aligning with the high level dialogue description provided in Section 3.1. Note that $why(A, T)$ can be used to initiate the dialogue, or made in response to a $why(\neg A, T)$ move. In other words, if a participant asks "Why did action A not occur?", asking "Why do you think action A should have occurred?" is a valid response, as it will allow for a disagreement in views to be detected. Also note that the assertion of a plan ($assert(\pi, T)$) can be made in response to asking why an action took place, why a belief was instantiated, or in response to the claim that some other plan should have been executed. The intuition behind the latter is that a dialogue participant suggests that another plan should have been executed. The dialogue can then continue to investigate why this is the case.

While Table 1 specifies what utterance can be made in response to a move, the contents of a legal utterance are further constrained. Table 2 provides a semi-formal description of each utterance, stating when a move can be made (the move condition), the move's closure condition, and move's effect on dialogue participants. Within the table, _ is used, as in Prolog, to indicate that any instantiation of the relevant value may exist.

We assume that the same move cannot be repeated. A dialogue D is then a sequence of moves $[D_1, \ldots, D_n]$ obeying all dialogue constraints (i.e., "Follows" requirements of Table 1, and "Move" and "Closure" conditions of Table 2). Note that we do not specify an explicit turn taking mechanism. Rather, dialogue participants make utterances in response to an open move subject to the move conditions. Different instantiations of the dialogue are therefore possible whereby, for example, an agent can respond to a question about why a plan holds by responding with a single belief assertion at a time, or by asserting all elements of the plan's guard simultaneously. While this may have an impact on dialogue understanding and dialogue length (which we will categorise as part of future work), the entire dialogue family will yield equivalent results in terms of the dialogue's goals (i.e., in identifying disagreements).

Moves such as $accept(\pi, T)$ which always close a dialogue branch either explicitly indicate agreement or disagreement with an utterance previously made by the other dialogue participant. In the latter case, they typically end the dialogue. Other moves are closed when the appropriate closure move exists. This closure move either identifies the disagreement, or refines where scope for disagreement exists. For example, when one participant asserts a plan was executed, and the other responds by asserting that some other plan was executed, participants no longer needs to discuss the former plan to identify disagreement. Instead, identifying *why* the latter plan was (believed to be) executed is enough to identify the disagreement.

The *precedence* utterance sets a constraint between π and π'. We assume in addition that the effect maintains a total ordering over plans in Π. We omit the requirement that $\pi' > \pi$ appear in $\overline{\Pi'}$ as the utterance's effect is sufficient to detect disagreement.

Note that there is an asymmetry with regards to closure conditions between $assert(B, T, T')$ and $assert(\neg B, T, T')$. The former can be closed by the latter, but not the other way around. The intuition behind this is that the assertion of a belief must identify the maximal interval during which the belief held. Providing an overlapping interval where it does not hold counters the assertion, but the new assertion must be explained (via a $why(\neg B, T)$ move) rather than requiring another assertion for the belief holding.

Finally, note that *why* moves have no effect on the dialogue participants, as such moves simply request more information without committing the utterer to any specific stance. However, the condition for uttering such a why move requires that the utterer have appropriate beliefs (e.g., for $why(B, T)$, the utterer has to believe that belief B held at time T. We do not impose a similar constraint when asking why an action did/didn't take place. Such utterances initiate the dialogue and requires a participant to believe that the other believes the action did/didn't take place but places no requirements on the utterer (i.e., constraints on $\mathcal{M}$), and without a response, does not constrain the other (i.e., does not constrain O).

4 DIALOGUE PROPERTIES

Having described the utterances dialogue participants can make, as well as how a dialogue is initiated and terminates, we now turn our attention to the properties of the dialogue. Due to space limitations, we provide proof sketches for these properties.

The first property we consider reflects the fact that the model held by one dialogue participant of the other always reflects the latter's true internal state if it did so previously.

Proposition 1. *If, before move D_i, for all indexes T, $\mathcal{B}'^O[T] \subseteq \mathcal{B}^{\mathcal{M}}[T]$, $\Pi'^O \subseteq \Pi^{\mathcal{M}}$, $\pi'^O[T] \in \{\emptyset\} \cup \{\pi^{\mathcal{M}}[T]\}$, $\tau_e'^O[T] \in \{\emptyset\} \cup$*

Utterance	Follows	Intuition
$why(\neg A, T)$	None	Asks why action A did not take place at time T.
$why(A, T)$	None $why(\neg A, T)$	Asks why action A took place at time T.
$did(A, T)$	$why(\neg A, T)$	Asserts that action A took place at T. Ends the dialogue.
$didnt(A, T)$	$why(A, T)$	Asserts that action A did not occur at time T. Ends the dialogue.
$assert(\pi, T)$	$why(A, T+1)$ $why(B, T+1)$ $why(\neg B, T+1)$ $assert(\pi', T)$	Asserts that plan π was selected for execution at time T in response to a question regarding why an action or belief held, or to counter a claim that another plan was executed at time T.
$not\_in\_library(\pi)$	$assert(\pi, T)$	States that the dialogue participant is not aware of plan π. Ends the dialogue.
$precedence(\pi, \pi')$	$assert(\pi, T)$	Follows a second $assert(\pi', T)$ move. States that plan π takes precedence over π'. Ends the dialogue.
$accept(\pi, T)$	$assert(\pi, T)$	Accepts that plan π was selected for execution at time T.
$why(\pi, T)$	$assert(\pi, T)$	Asks why plan π was selected for execution at time T.
$assert(B, T, T')$	$why(\pi, T'+1)$	Asserts that belief B exists at all times between T and T' (inclusive).
$assert(\neg B, T, T')$	$assert(B, T'', T')$	Asserts that B did not exist at all times between T and T'.
$accept(B, T, T')$	$assert(B, T, T')$	Accepts that B holds between T and T'.
$accept(\neg B, T, T')$	$assert(\neg B, T, T')$	Accepts that B does not hold between T and T'.
$why(B, T)$	assert(B, T, T')	Asks why belief B exists at time T.
$why(\neg B, T)$	assert$(\neg B, T, T')$	Asks why belief B does not exist at time T.
$percept(+B, T)$	$why(B, T+1)$	Explains that B was perceived at time T in response to asking why it held at the next time point.
$percept(-B, T)$	$why(\neg B, T+1)$	Explains that B was perceived being removed at time T.

Table 1: Legal dialogue utterances and what moves they follow, as well as their intuitive meaning.

$\{\tau_e^{\mathcal{M}}[T]\}$, *and* ${a'_{ex}}^{O}[T] \in \{\emptyset\} \cup \{a_{ex}^{\mathcal{M}}[T]\}$, *then this will also be the case following the move.*

Proof. We note that *why* moves do not affect the traces, and therefore only consider the remaining move types.

Those moves which update an element of O' do so in a way consistent with $\mathcal{M}$, giving us the desired result. □

Note that the update procedure described above also updates the constraints in $\overline{O}'$ in a manner consistent with $\mathcal{M}$. Therefore, $O', \overline{O}'$ are consistent with the program execution trace $\mathcal{M}$.

Corollary 1. *Given two dialogue participants* $\langle \mathcal{M}, O, \overline{O} \rangle, \langle \mathcal{M}', O', \overline{O'} \rangle$, *if* $\mathcal{M}$ *and* O' *contain no contradictions prior to move* D_i, *then they will contain no contradictions following it.*

Next, we demonstrate that our dialogues always terminate.

Theorem 4.1. *Given a finite set of plans* Π *with a finite set of propositions in their guards, G, then any dialogue starting with a why question on an action will terminate.*

Proof. We show that any move at time T either immediately closes the dialogue, or is closed when move referring to a time $T' < T$ is made, closing it. Since the lowest possible time is 0 and no moves may refer to time intervals below this, any dialogue must terminate. We consider each possible move individually.

- $why(\neg A, T)$: There are two possible responses: $did(A, T)$, which closes the dialogue, or $why(A, T)$.
- $why(A, T)$: Possible responses are $didnt(A, T)$, which closes the dialogue, or $assert(\pi, T-1)$.
- $assert(\pi, T)$: is closed immediately in case of *accept*, *not_in_library* and *precedence* moves, and we need only consider $why(\pi, T)$ and $assert(\pi', T)$. Given the move's conditions, a third *assert* of a plan cannot occur, meaning either that $why(\pi', T)$ will be asked, or one of the closure moves mentioned previously must be played. Therefore, we must show that $why(\pi, T)$ will be closed.
- $why(\pi, T)$: The only response is a set of $assert(B, T', T-1)$ where $T' \leq T-1$. Each of these considers a time before T.
- $assert(B, T', T)$: An $accept(B, T', T)$ closes the move, and we must therefore consider $assert(\neg B, T'', T)$ and $why(B, T)$ moves, which as shown below, consider a time less than T.
- $assert(\neg B, T', T)$: An $accept(\neg B, T', T)$ closes the move. We must therefore show $why(\neg B, T)$ for a time before T.
- $why(B, T)/why(\neg B, T)$: This move is closed by a $percept(+B, T-1)/percept(-B, T-1)$ move. The only other response is $assert(\pi, T-1)$, which considers a time $T-1$.

It is also clear, given the requirement that $stage[0] = \mathsf{p}$, that no dialogue will refer to a time $T < 0$. □

Turning to the question of dialogue complexity, we demonstrate that the worst case length of a dialogue depends on the number of plans in the plan library and the size of plan guards for each plan.

Corollary 2. *The complexity of creating a dialogue is polynomial in the size of the plan library and plan's guard.*

Proof. Let k be the total number of plans in the plan library. Since moves cannot be repeated the maximum number of asserts which can take place in any branch of the tree is k. Furthermore, the dialogue can branch whenever a $why(\pi, T)$ is asked, with a factor equal to the number of beliefs in the guard of the plan (call these g). Finally, from the previous theorem, we know that a dialogue can take place for at most T time points, meaning that the upper bound for the number of moves is $O(g^{Tk})$. □

The following proposition states that if a disagreement within the dialogue participant's views (M) exists, its root cause — the

Utterance	Move Condition	Closure Condition	Effect
$why(\neg A, T)$	$stage^{\mathcal{M}}[T] = \mathsf{e}$	$did(A, T)$ or $why(A, T)$ in the dialogue	None
$why(A, T)$	$stage^{\mathcal{M}}[T] = \mathsf{e}$	$didnt(A, T)$ or $assert(\pi, T-1)$ s.t. $\pi = \_ \rightarrow [\_, do(A)]$ in the dialogue.	None
$why(\pi, T)$	$\pi = \pi^{O}[T]$	If π is of the form $[b_1, \ldots, b_n] \rightarrow [U, \_]$ then there is a move $assert(b_i, \_, T-1)$ for all $i = 1, \ldots, n$.	None
$why(B, T)$	$B \in \mathcal{B}^{O}[T]$	When there is a move $percept(+B, T-1)$ or a move $assert(\pi, T-1)$ such that π is of the form $\_ \rightarrow [U, \_]$ and $+B \in U$.	None
$why(\neg B, T)$	$B \notin \mathcal{B}^{O}[T]$	When there is a move $percept(-B, T-1)$ or a move $assert(\pi, T-1)$ such that π is of the form $\_ \rightarrow [U, \_]$ and $-B \in U$.	None
$assert(\pi, T)$	$stage^{\mathcal{M}}[T] = \mathsf{s}$ and $\pi = \pi^{\mathcal{M}}[T]$. If following $why(A, T+1)$ (equivalently $why(B, T+1)$ or $why(\neg B, T+1)$) then π is of the form $\_ \rightarrow [\_, do(A)]$ (equivalently $\_ \rightarrow [[\ldots, +B, \ldots], \_]$ or $\_ \rightarrow [[\ldots, -B, \ldots], \_]$).	Dialogue contains one of the following: $why(\pi, T)$ $assert(\pi', T)$ $accept(\pi, T)$ $not\_in\_library(\pi)$ $precedence(\pi, \pi')$	${\pi'}^{O}[T] = \pi$. If following an $assert(\pi', T)$ then π is added to $\overline{\pi'}[T]$.
$assert(B, T, T')$	$B \in \mathcal{B}^{\mathcal{M}}[i]$ such that $T \leq i \leq T'$	Dialogue contains $why(B, T')$ or $accept(B, T, T')$ or $assert(\neg B, T'', T')$ (where $T'' \leq T'$).	B is added to all ${\mathcal{B}'}^{O}[i]$ for all $T \leq i \leq T'$
$assert(\neg B, T, T')$	$B \notin \mathcal{B}^{\mathcal{M}}[i]$ such that $T \leq i \leq T'$	Dialogue contains $why(\neg B, T')$ or $accept(\neg B, T, T')$	B is added to $\overline{\mathcal{B}'}[i]$ for all $T \leq i \leq T'$
$did(A, T)$	$A = a_{ex}^{\mathcal{M}}[T]$	Always closed	${a'}_{ex}^{O}[T] = A$
$didnt(A, T)$	$A \neq a_{ex}^{\mathcal{M}}[T]$	Always closed	$\overline{a'_{ex}}[T] = A$
$not\_in\_library(\pi)$	$\pi \notin \Pi^{\mathcal{M}}$	Always closed	π is added to $\overline{\Pi'}$
$precedence(\pi, \pi')$	The moves $assert(\pi, T)$, $assert(\pi', T)$ are in the dialogue. $\pi, \pi' \in \Pi^{\mathcal{M}}$ and $\pi > \pi'$ there.	Always closed	$\pi > \pi' \in {\Pi'}^{O}$
$accept(\pi, T)$	$\pi = \pi^{\mathcal{M}}[T]$	Always closed	${\pi'}^{O}[T] = \pi$
$accept(B, T, T')$	$B \in \mathcal{B}^{\mathcal{M}}[i]$ for all $T \leq i \leq T'$	Always closed	B is added to ${\mathcal{B}'}^{O}[i]$ for all $T \leq i \leq T'$
$accept(\neg B, T, T')$	$B \notin \mathcal{B}^{\mathcal{M}}[i]$ for all $T \leq i \leq T'$	Always closed	B is added to $\overline{\mathcal{B}'}[i]$ for all $T \leq i \leq T'$
$percept(+B, T)$	$stage^{\mathcal{M}}[T] = \mathsf{p}$ and $+B$ is contained within the set at the head of $\tau_e^{\mathcal{M}}[T]$	Always closed	B is added to ${\mathcal{B}'}^{O}[T+1]$
$percept(-B, T)$	$stage^{\mathcal{M}}[T] = \mathsf{p}$ and $-B$ is contained within the set at the head of $\tau_e^{\mathcal{M}}[T]$	Always closed	B is added to $\overline{\mathcal{B}'}[T+1]$

Table 2: Preconditions for an utterance; requirements to label the move closed; and utterance effects on dialogue participants.

difference in plans, perceptions or beliefs which led to it — can be detected by the dialogue, assuming that some aspect of the disagreement was already known to the dialogue participants (e.g., a difference in perceived action).

Proposition 2. *Given two agents for which $\mathcal{M} \neq \mathcal{M}'$ and for which $a_{ex}^{\mathcal{M}}[T] \neq a_{ex}^{\mathcal{M}'}[T]$, there is a dialogue which terminates with a not_in_library, precedence or percept move.*

Proof. We know from Theorem 4.1 that all dialogues terminate. We show that there is at least one belief or plan that is not *accept*ed.

Note that since there is a disagreement in actions, the dialogue can initiate by asking why the disagreed upon action was executed, meaning that we can ignore $did/didnt$ moves.

Assume the proposition is false. This would mean that there is agreement on which plan was executed. But there is a disagreement in actions, which means that $\mathcal{M}$ believes a plan with head $a_{ex}^{\mathcal{M}}[T]$ was executed, while $\mathcal{M}'$ believes a plan with head $a_{ex}^{\mathcal{M}}[T]$. Therefore this is a contradiction. The only way to close the dialogue is either with a *not_in_library*, or *precedence* move, asking *why* the plan was selected, or for another plan to be asserted leading to the same arguments as above. Therefore only the question regarding *why* a plan was chosen does not (eventually) close the move. In turn, this leads to a disagreement about beliefs, which can only be resolved via a *percept* disagreement, or by asking about further plans. over which there must (as above) be a disagreement. □

It follows trivially that at least one dialogue participant will be able to identify the disagreement by examining their $\mathcal{M}$ and O.

The results above suggests a simple strategy for identifying the root cause of a disagreement, namely to never *accept* when a disagreement exists, and always ask *why* about such disagreements. Such *disagreement* dialogues can be contrasted from *confirmatory* dialogues, where one participant may wish to confirm that the other's internal trace matches their own. A simple strategy for such

confirmatory dialogues involves always asking *why* where possible, *accepting* only when no other move exists.

Finally it can be easily shown that if both $\mathcal{M}$ and $\mathcal{M}'$ are identical, all dialogues will terminate with *did*/*didnt* moves (if the initial *why* asks about a move that did/didn't occur, or *accept* moves. In other words, no disagreement will be identified.

5 IMPLEMENTATION

We have implemented SimpleBDI and our dialogue explanation system in Python[3]. In our system two agents execute the program. Perceptions are supplied to each agent individually at certain time steps – allowing for differences in execution to occur because of differing perceptions. Once execution is completed, a trace of the actions performed by the two agents is used to detect points where their behaviour diverged and these points can be used to start a dialogue. For convenience the first agent in the dialogue is referred to as the human, though it should be noted that our dialogues are in fact generated by two software agents conversing.

Figure 2 shows a sample dialogue generated by our system for Example 1. In this example when the robot reached the waypoint it perceived that the terrain was no longer safe and so did not move to the final location. The human asks why it did not make this move. The robot and human agree that it was at the waypoint, and that it had a goal to move to the final location, but they realise they diagree that the terrain was safe and the robot explains that it no longer believed the terrain to be safe from time point 15.

Figure 3 shows a dialogue for a different example. In this example the robot is charged with performing routine remote inspections of some site (e.g., a nuclear waste storage facility). When it performs its daily inspection it should inspect the walls of the facility (if they are scheduled for inspection) and the stored barrels (if they are stored for inspection). In the situation where both inspections are scheduled then inspecting the barrels takes precedence (indicated implicitly by the ordering of the plans in the robot's plan library). This simple program is shown in Listing 2.

Code Listing 2

```
daily_inspection, barrels_scheduled ->          1
                    do(inspect_barrels)         2
daily_inspection, wall_scheduled ->             3
                    do(inspect_wall)            4
```

Figure 3 shows a dialogue generated for an instance where the human believes that the wall inspection should have priority over barrel inspection. The human asks why the robot did not inspect the wall, the robot counters by asking why the human thought it should inspect the wall. They both explain the plan they thought applicable at that point and the human asserts that they thought the wall inspection plan had priority.

6 DISCUSSION AND FUTURE WORK

A dialogue participant can make multiple utterances in some stages of the dialogue. For example, a possible response to a $why(\pi, T)$ move, which could be a single $assert(b_1, \_, T-1)$ move followed by a sub-dialogue to close this assertion, after which a second $assert(b_2, \_, T-2)$ move can be made followed by another sub-dialogue. Alternatively, a response consisting of multiple moves of the form $assert(b_1, \_, T-1), \ldots, assert(b_n, \_, T-1)$ could be made, closing the original *why* move, but leaving all the assertions open until dealt with. Rules covering turn taking (for example) would then instantiate specific dialogues, but this would not affect the dialogue properties described previously.

Our work makes an important assumption, namely that both dialogue participants apply the SimpleBDI semantics correctly to their internal version of the BDI program. In other words, the disagreements we identify come about from omissions or differences in the plan library, in the initial set of beliefs held by the dialogue participants, or differences in beliefs regarding the input trace τ_e. Extending the dialogue to deal with fallible participants who may simply forget a belief or to apply a rule is an important strand of future work, as doing so will provide for a dialogue more suited to humans acting as dialogue participants.

At worst, our dialogue identifies only a single disagreement between participants.We assume that between dialogues, participants update their beliefs about the program execution trace and therefore, on rerunning the dialogue would identify different disagreements. Determining how such belief updates should take place is outside the scope of the paper, but serves as another important avenue of future work. Related to this, allowing the participants to update their $\mathcal{M}$ models during the trace (with concomitant effects on O' and $\overline{O}'$) would enable more disagreements to be discovered during single instance of the dialogue. Such work would require, at the very least, the addition of moves to retract beliefs [**?**].

Explanation has become an important area of AI research. Much of the work in the domain focuses on the explainability of machine learning systems [**?**], but several recent papers consider explanation of BDI and planning systems. For example, Caminada *et al.* introduced a dialogue game to explain the behaviour of an automated planner [**? ?**], building on ideas taken from proof dialogues [**?**]. We note that such work considers how to translate formal utterances (as per our dialogue) into natural language, and believe that this will be an interesting avenue of future work.

Several other argumentation based approaches to explanation have been proposed in the literature (e.g., [**? ?**]). While these approaches could be adapted to explain the behaviour of BDI systems, we are unaware of such adaptations, which would — at least — require instantiating BDI specific concepts related to time, beliefs, goals etc as rules, which could then be combined into arguments, and over which explanation dialogues could then operate. In contrast, our current work does not utilise an argumentation-theoretic semantics to underpin it. Instead, it could be viewed as a dialogue game built using argument schemes and critical questions [**?**] created for the BDI domain, in the tradition of work in informal logic [**?**] and practical reasoning [**?**].

Winikoff [**?**] and Hindriks [**?**] both consider providing explanations for BDI languages in the context of debugging. Hindriks' work was later expanded by Koeman *et al.* [**?**]. These systems all generate explanations using a formal semantics over a trace of program execution. Harbers [**?**] generates explanations for BDI systems using goal hierarchy paired with a behaviour log. Winikoff *et al.* [**?**] uses a concept of preferences to help produce explanations

[3]Source code can be found at https://github.com/jhudsy/BDIexplanation.git.

```
human: Why Not move2 at 17
robot: Why move2 at 17
human: Selected at_waypoint,goal_move_to_location,safe_terrain, ->
           do(move2),-goal_move_to_location,-at_waypoint,+at_location,+goal_take_sample, at 16
robot: Why select at_waypoint,goal_move_to_location,safe_terrain, ->
           do(move2),-goal_move_to_location,-at_waypoint,+at_location,+goal_take_sample, at 16
human: +at_waypoint at time 14 and it remained so until at least 16
robot: I agree +at_waypoint between 14 and 16
human: +goal_move_to_location at time 11 and it remained so until at least 16
robot: I agree +goal_move_to_location between 11 and 16
human: +safe_terrain at time 6 and it remained so until at least 16
robot: -safe_terrain at time 15 and it remained so until at least 16
human: Why -safe_terrain at 15
robot: I perceived -safe_terrain at 15
```

Figure 2: Sample Dialogue for Example 1

```
human: Why Not inspect_wall at 14
robot: Why inspect_wall at 14
human: Selected wall_scheduled,daily_inspection, -> do(inspect_wall), at 13
robot: Selected barrels_scheduled,daily_inspection, -> do(inspect_barrels), at 13
human: wall_scheduled,daily_inspection, -> do(inspect_wall), has precedence in my plan library
```

Figure 3: Sample Dialogue for Plan Priority Example

from BDI program execution traces. While all of these systems — like us — use execution traces to provide explanations for BDI program, none compare conflicting traces through dialogue to guide the generation of the explanation towards the concerns of the user.

Sreedharan *et al.* [**?**] consider the question of explanations in the context of AI Planning and, like us, explicitly identify the need to reconcile the human mental model with execution to generate an explanation. They pre-generate a set of explanations that are intended to reveal specific aspects of the Planning system's model (for instance that a particular location must be visited in particular circumstances) and then use machine learning to determine which explanations are most likely to explain which observable transitions in the system behaviour. These are then presented to users when they label some particular transition as inexplicable. [**?**] describes how hypothetical plans can be generated which can be compared to the original plan, serving a similar function to our two dialogue participants. However, these approaches ignore the dialogical aspects of our solution and are grounded in planning, reducing the importance of concepts such as percepts.

Apart from the research mentioned above, there are several additional strands of future work we intend to explore. First, as noted by Caminada *et al.* and others, there are strong links between dialogues and formal argumentation theory. Move sequences such as $assert(\pi, T)$, $assert(\pi', T)$ imply a contradiction in the dialogue participant's views which — through the dialogue — are instantiated into attacking arguments. We therefore intend to investigate an argument-theoretic semantics for the dialogue presented in this paper, potentially allowing for stronger links with other explainable AI approaches underpinned by argumentation [**? ? ?**], potentially allowing for more efficient dialogues through the introduction of concepts such as burden of proof [**?**]. We also intend to investigate the effects of strategy on dialogue properties more deeply. While our results provide worst-case upper bounds for dialogue length, strategies regarding what utterance to make, built on what the dialogue participant wishes to achieve and what they know about the other participant [**?**] may — at least in the average case — significantly reduce the number of moves that need to be made. Finally, extending SimpleBDI may result in more complex dialogues. Allowing, for example, a non-strict ordering over plans could allow participants to argue about the unobserved effects of plans, requiring looking forwards as backwards over time, and such enrichment could be a fruitful direction of future work.

7 CONCLUSIONS

We presented a family of dialogues allowing two dialogue participants to identify if, and where, a divergence of views exists between them with regards to a BDI agent's operation. Our dialogue aims to be general enough to capture two external observers discussing the behaviour of a (third) BDI agent, but we believe that in practice, one of the dialogue participants will be the BDI agent, seeking to explain its actions to the second participant, typically a human. Such explanations then focus on divergences in the views of the participants with regards to the perceptions, plans and underlying beliefs of the BDI system, and we show that when a divergence exists with regards to what action should have taken place, the dialogue enables the root cause of the divergence to be detected.

ACKNOWLEDGMENTS

This work arose out of conversations at a Lorentz Workshop on the Dynamics of Multi-Agent Systems (2018). Thanks are due Koen Hindriks and Vincent Koeman for their input. The work was supported by the UKRI/EPSRC RAIN [EP/R026084], SSPEDI [EP/P011829/1] and FAIR-SPACE [EP/R026092] Robotics and AI Hubs.

REFERENCES

[] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[] Katie Atkinson and Trevor Bench-Capon. 2007. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence* 171, 10 (2007), 855 – 874. https://doi.org/10.1016/j.artint.2007.04.009 Argumentation in Artificial Intelligence.

[] P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. 2018. *Handbook of Formal Argumentation*. College Publications.

[] Martin W Caminada, Roman Kutlak, Nir Oren, and Wamberto Weber Vasconcelos. 2014. Scrutable plan enactment via argumentation and natural language generation. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1625–1626.

[] Kristijonas Cyras, Xiuyi Fan, Claudia Schulz, and Francesca Toni. 2017. Assumption-based Argumentation: Disputes, Explanations, Preferences. *FLAP* 4, 8 (2017). http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf

[] Kristijonas Čyras, Dimitrios Letsios, Ruth Misener, and Francesca Toni. 2019. Argumentation for explainable scheduling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2752–2759.

[] L. Dennis, M. Fisher, M. Webster, and R. Bordini. 2012. Model Checking Agent Programming Languages. *Automated Software Engineering* 19, 1 (2012), 5–63.

[] Emilia Garcia, Gareth Tyson, Simon Miles, Michael Luck, Adel Taweel, Tjeerd Van Staa, and Brendan Delaney. 2013. Analysing the Suitability of Multiagent Methodologies for e-Health Systems. In *Agent-Oriented Software Engineering XIII*, Jörg P. Müller and Massimo Cossentino (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 134–150.

[] Maaike Harbers. 2011. *Explaining Agent Behaviour in Virtual Training*. Ph.D. Dissertation. SIKS Dissertation Series. No. 2011-35.

[] Koen V. Hindriks. 2012. Debugging Is Explaining. In *PRIMA 2012: Principles and Practice of Multi-Agent Systems*, Iyad Rahwan, Wayne Wobcke, Sandip Sen, and Toshiharu Sugawara (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 31–45.

[] Vincent Koeman, Louise A. Dennis, Matt Webster, Michael Fisher, and Koen Hindriks. 2019. The "Why did you do that?" Button: Answering Why-questions for end users of Robotic Systems. In *Proceedings of the 7th International Workshop in Engineering Multi-Agent Systems*. Montreal, Canada. http://cgi.csc.liv.ac.uk/~lad/emas2019/accepted/EMAS2019_paper_27.pdf

[] Benjamin Krarup, Michael Cashmore, Daniele Magazzeni, and Tim Miller. 2019. Model-based contrastive explanations for explainable planning. In *ICAPS 2019 Workshop on Explainable AI Planning (XAIP)*.

[] Nir Oren, Kees van Deemter, and Wamberto W. Vasconcelos. 2020. *Argument-Based Plan Explanation*. Springer International Publishing, Cham, 173–188. https://doi.org/10.1007/978-3-030-38561-3_9

[] Henry Prakken, Chris Reed, and Douglas Walton. 2005. Dialogues About the Burden of Proof. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law* (Bologna, Italy) *(ICAIL '05)*. ACM, New York, NY, USA, 115–124. https://doi.org/10.1145/1165485.1165503

[] Tjitze Rienstra, Matthias Thimm, and Nir Oren. 2013. Opponent models with uncertainty for strategic argumentation. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[] Elizabeth I. Sklar and Mohammad Q. Azhar. 2018. Explanation through Argumentation. In *Proceedings of the 6th International Conference on Human-Agent Interaction* (Southampton, United Kingdom) *(HAI '18)*. Association for Computing Machinery, New York, NY, USA, 277–285. https://doi.org/10.1145/3284432.3284470

[] Sarath Sreedharan, Alberto Olmo, Aditya Prasad Mishra, and Subbarao Kambhampati. 2019. Model-free Model Reconciliation. In *IJCAI*.

[] Douglas Walton. 2008. *Informal Logic: A Pragmatic Approach* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511808630

[] Douglas Walton and Erik CW Krabbe. 1995. *Commitment in dialogue: Basic concepts of interpersonal reasoning*. SUNY press.

[] Li Weigang, Bueno Borges de Souza, Antonio Marcio Ferreira Crespo, and Daniela Pereira Alves. 2008. Decision support system in tactical air traffic flow management for air traffic flow controllers. *Journal of Air Transport Management* 14, 6 (2008), 329–336.

[] Michael Winikoff. 2017. Debugging Agent Programs with Why? Questions. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (São Paulo, Brazil) *(AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 251–259.

[] Michael Winikoff, Virginia Dignum, and Frank Dignum. 2016. Why Bad Coffee? Explaining Agent Plans with Valuings. In *SAFECOMP (LNCS, Vol. 9923)*, A. Skavhaug, J. Guiochet, E. Schoitsch, and F. Bitsch (Eds.). Spinger, 521–534.