Chapter 9
Argument-Based Plan Explanation

Nir Oren, Kees van Deemter, and Wamberto W. Vasconcelos

Abstract We describe a tool for providing explanation of plans to non-technical users, built on formal argumentation and dialogue theory, and supported by natural language generation and visualisation technologies. We describe how arguments can be generated from domain rules, and how justified arguments can be identified through dialogue, allowing the system to use such a dialogue to explain a plan. We provide information about our prototype system implementation, discussing its current limitations, and identifying potential avenues for future research.

1 Introduction

Automated planners have, together with other technologies, enabled autonomous systems to generate and then execute plans in pursuit of a set of goals with little or no human intervention. While such plans are often better than those a human planner can create, there is a reliance on the correct specifying the initial and goal states, as well as the effects of actions, making such plans brittle in the presence of exceptional (and unexpected) situations.

There is, therefore, a clear need to be able to verify or validate the correctness of the plan specification with regards to the current environmental state. Furthermore, autonomous systems do not operate in isolation, but often form part of a *human-agent* team. In such cases, joint plans dictate both human and autonomous system actions, and mechanisms are required to ensure that humans execute their portion of the plan correctly. If the human actors trust the correctness of a plan, they are

N. Oren (✉) · W. W. Vasconcelos
Computing Science, University of Aberdeen, Aberdeen, UK
e-mail: n.oren@abdn.ac.uk; w.w.vasconcelos@abdn.ac.uk

K. van Deemter
Information & Computing Sciences, University of Utrecht, Utrecht, The Netherlands
e-mail: c.j.vandeemter@uu.nl

M. Vallati, D. Kitchin (eds.), *Knowledge Engineering Tools and Techniques for AI Planning*, https://doi.org/10.1007/978-3-030-38561-3_9

more likely to follow it. One way to engender such trust, which also addresses the validation and verification problem, is to provide an explanation of the generated plan.

We argue that *plan explanation* can serve to improve human trust in a plan. Such plan explanation can take on several forms. *Visual plan explanation* [25] presents the user with a graphical representation of a plan (e.g., with nodes representing actions, edges providing temporal links between actions, and paths representing different plans), and allows for different filters to be applied in order to mitigate cognitive overload. We briefly discuss one instantiation of such techniques in Sect. 4.1.

The second approach for plan explanation that we consider here involves a textual presentation of the plan in natural language, which is created through interaction with the user. This *dialogue based* approach allows a user to ask questions about the plan or about alternative plans, and understand the reasons why specific planning steps were selected. By allowing users to guide the dialogue, the information most relevant to them can be presented, reducing the time needed for them to understand the selected plan, and militating against information overload. An appropriate choice of dialogue will also allow a user to provide new information to the system, allowing re-planning to take place in a natural manner. Our focus in this paper is on how argument and dialogue can be employed to provide plan explanation. This second approach builds on argumentation and formal dialogue theory to select what information to convey. The information is then presented using natural language through the application of Natural Language Generation (NLG), the area of Language Technology where algorithms are developed that can automatically convert "data" into text [11, 23].

The remainder of the chapter is structured as follows. In the next section we provide a brief overview of formal argumentation and dialogue theory, a branch of knowledge representation on which our dialogue based approach is built. Following this, we describe some proof dialogues which can be applied to plan explanation, before describing a plan explanation application we created as part of the "Scrutable Autonomous Systems" project.[1] In Sect. 5 we discuss related work, before identifying current and future avenues of research in Sect. 6, and concluding.

2 Argumentation and Dialogue

The process of explanation can be viewed as the provision of a justification for some conclusion, or equivalently, as advancing some set of arguments which justify the conclusion. Research in formal argumentation theory has described the nature which such justification can take, and we build our textual explanations on this theory. We therefore begin by providing a high-level overview of argumentation, which underpins our approach to plan explanation.

[1]Funded by the Engineering and Physical Sciences Research Council (EPSRC, UK), Grant ref. EP/J012084/1, 2012–2015.

2.1 Abstract Argumentation

Dung's seminal 1995 paper [9] described how, given a set of arguments and attacks between them, one could identify which arguments remain justified. Dung did not consider how arguments were formed, and his approach therefore treats arguments as atomic entities which are part of an abstract *argumentation framework*.

Definition 1 (Argumentation Framework [9]) An argumentation framework is a pair $(\mathcal{A}, \mathcal{D})$ where $\mathcal{A}$ is a set of arguments, and $\mathcal{D} : \mathcal{A} \times \mathcal{A}$ is a binary *defeat* relation over arguments.

An abstract argumentation framework can be represented visually as a graph, with nodes denoting arguments, and edges denoting defeats between them.

An *extension* is a subset of arguments from within $\mathcal{A}$ that is in some sense justified. Perhaps the simplest requirement for a set of arguments to be justified is that they do not contradict, or conflict with each other, as modelled via the defeat relation.

Definition 2 (Conflict Free) Given an argumentation framework $(\mathcal{A}, \mathcal{D})$, a set of arguments $A \subseteq \mathcal{A}$ is *conflict free* if there is no $a, b \in A$ such that $(a, b) \in \mathcal{D}$.

A slightly stronger criteria for an argument to be justified is that no defeat against it should succeed. For this to occur, the argument should either not be defeated, or should be defended from the defeat by some other arguments.

Definition 3 (Defence and Admissibility) Given an argumentation framework $(\mathcal{A}, \mathcal{D})$, an argument $a \in \mathcal{A}$ is defended by a set of arguments $S \subseteq \mathcal{A}$ if, for any defeat $(b, a) \in \mathcal{D}$, it is the case that there is a $s \in S$ such that $(s, b) \in \mathcal{D}$. A set of arguments S is then said to be *admissible* if it is conflict free and if each argument in S is defended by S.

Building on the notion of admissible arguments, we may define *extensions*, which identify discrete groups of arguments that can be considered justified together.

Definition 4 (Extensions) Given an argumentation framework $(\mathcal{A}, \mathcal{D})$, a set of arguments $S \subseteq \mathcal{A}$ is a

- **complete extension** if and only if it is admissible, and every argument which it defends is within S.
- **preferred extension** if and only if it is a maximal (with respect to set inclusion) complete extension.
- **grounded extension** if and only if it is the minimal (with respect to set inclusion) complete extension.
- **stable extension** if it is conflict free and defeats any argument not within it.

While other extensions have been defined (see [2]) for details, these four extensions capture many of the intuitions regarding what it means for a set of arguments to be justified.

It should be noted that for a given argumentation framework, there will be only a single unique grounded extension. However, multiple complete and preferred extensions may exist, as can zero or more stable extensions. An argument is said to be sceptically accepted under a semantics X if it appears in all X extensions; it is credulously preferred if it appears in at least one, but not all such extensions.

2.2 Labellings

Labellings [1] provide another approach to computing extensions. A labelling $\mathcal{L} : \mathcal{A} \rightarrow \{\mathbf{IN}, \mathbf{OUT}, \mathbf{UNDEC}\}$ is a total function mapping each argument to a single label. Informally, **IN** denotes that an argument is justified; **OUT** that it is not, and **UNDEC** that its status is uncertain.

Wu et al. [28] among others demonstrated an equivalence between such labellings and different argumentation semantics. For example, the following constraints are those required for the arguments labelled **IN** to be equivalent to those within a complete extension:

- An argument is labelled **IN** if and only if all its defeaters are labelled **OUT**.
- An argument is labelled **OUT** if and only if at least one of its defeaters is labelled **IN**.
- It is labelled **UNDEC** otherwise.

Maximising the number of **UNDEC** arguments will result in a labelling which is equivalent to the grounded semantics while minimising these arguments will yield a preferred extension. While some argue that labellings are more intuitive (especially to non-technical audiences) than the standard argumentation semantics, the question of how a legal labelling can be identified still remains. Several algorithms for identifying legal labellings have been proposed [17] whose complexity mirrors the complexity of computing the relevant argumentation semantics.

Proof dialogues, which we discuss in Sect. 3 next, are another technique for computing an argumentation semantics. As their name implies, such proof dialogues seek to mirror some form of discussion, building up the elements of an extension as the dialogue progresses. Before considering proof dialogues, we consider how arguments are generated.

2.3 From Knowledge to Arguments

While abstract argumentation allows us to identify which arguments are justified, we must also consider how arguments are generated. In this section we introduce a simple *structured* argumentation framework which allows for the construction of arguments from a knowledge base. The system we consider here is a slight

simplification of ASPIC− [4] which in itself is a variant of ASPIC+ which includes several simplifications and enables *unrestricted rebut*, as explained below.

The knowledge base of ASPIC− consists of a set of *strict* and *defeasible* rules. The former encode standard modus ponens, while the latter represent rules whose conclusions hold by default. We write $P \rightarrow c$ where P is a set of literals and c is a literal to denote a strict rule; $P \Rightarrow c$ encodes a defeasible rule. In both cases, P are the rule's premises, and c is the rule's conclusion. We also assume a preference ordering $\prec$ over defeasible rules, and that—given the standard negation operator $\neg$—the set of strict rules is closed under contraposition. In other words, given a strict rule $a \rightarrow b$ in the knowledge base, the rule $\neg b \rightarrow \neg a$ must also be present.

Arguments are constructed by nesting rules. An argument is made up of a set of *sub-arguments* and a single *top rule*. We can formalise this as follows.

Definition 5 Given a set of rules KB and a set of arguments S, we can construct an argument $A = \langle tr, sa \rangle$ where $tr \in KB$ is a rule and $sa \subseteq S$ is a set of arguments such that if tr is of the form $P \rightarrow c$ or $P \Rightarrow c$, then for every $p \in P$ there is an argument $a_p \in sa$ whose top rule has conclusion p, and $sa = \bigcup\{a_p\} \bigcup$ sub-arguments of a_p.

An argument is said to be strict if its top rule is strict, and all of its sub-arguments are strict. The final conclusion of an argument is the conclusion of its top rule, while an argument's conclusions consist of its final conclusion and the final conclusion of all its sub-arguments.

An argument ($A1$) for a simple strict or defeasible fact can be introduced through a rule with no premises, e.g., $A1 :\rightarrow rw1$. If a second rule $R2 : rw1 \Rightarrow rf$ exists, then a second argument $A2 : A1 \Rightarrow rf$ can be obtained. The top rule of this latter argument is the $R2$, while its sub-argument is $A1$. We illustrate the argument generation process with a running example.

Example 1 Consider a UAV which has two choices regarding where to land, namely runway 1 (*rw1*), or runway 2 (*rw2*). While it believes it has sufficient fuel to reach both runways, *rw1* is further, meaning it will have to utilise its reserve fuel. However, given the runway length and weight of surveillance equipment it is carrying, landing at *rw2* is also considered dangerous. The UAV is programmed to prefer dipping into its fuel reserves over landing on a short runway. This can formally be encoded through the set of rules shown in Table 9.1. In turn, these rules result in the arguments shown in Table 9.2.

As in abstract argumentation, arguments interact with each other via attacks. While ASPIC− considers undercutting attacks (where one rule makes another inapplicable) as well as rebutting attacks (where the conclusions of a rule are in conflict with another rule's premises or conclusions), in this chapter we consider only the latter type of attack.

Definition 6 Given two arguments A and B, argument A attacks B (via *rebut*) if the conclusion of A's top rule is either

Table 9.1 Rules for the UAV example

	Rule	Description
R1	$\Rightarrow \neg ow$	(By default) we are not overweight
R2	$\Rightarrow \neg rf$	(By default) we are not using reserve fuel
R3	$\Rightarrow rw1$	(By default) we will land at rw1
R4	$\Rightarrow rw2$	(By default) we will land at rw2
R5	$rw1 \rightarrow rf$	Landing at rw1 will use reserve fuel
R6	$rw2 \rightarrow ow$	Landing at rw2 will cause us to be overweight

Table 9.2 Arguments obtained from the rules of Table 9.1

A0	$A13 \rightarrow \neg rw1$	A1	$A7 \rightarrow \neg rw2$
A2	$A12 \rightarrow ow$	A3	$A11 \rightarrow rw1$
A4	$A3 \rightarrow rf$	A5	$A13 \rightarrow ow$
A6	$\Rightarrow \neg rf$	A7	$\Rightarrow rw1$
A8	$\Rightarrow \neg ow$	A9	$A6 \rightarrow \neg rw1$
A10	$A7 \rightarrow rf$	A11	$A8 \rightarrow \neg rw2$
A12	$A9 \rightarrow rw2$	A13	$\Rightarrow rw2$

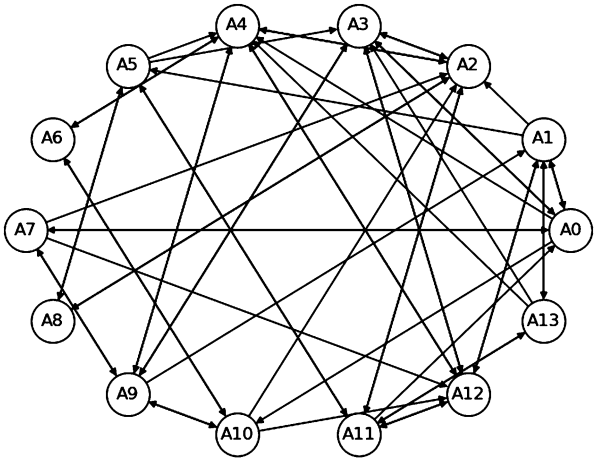

Fig. 9.1 The attacks obtained from the UAV example

– the negation of the conclusion of B's top rule; or
– the negation of the conclusion of B''s top rule where B' is a sub-argument of B.

Example 2 Definition 6 applied to the arguments in Table 9.2 results in the argument graph shown in Fig. 9.1. This argument graph has multiple preferred extensions, but no grounded extension.

Attacks between arguments reflect inter-argument inconsistencies, but do not take preferences or priorities between rules into account. Attacks are transformed into *defeats* when these priorities are considered. For an argument A, if we denote all rules used within it (including the rules used within its sub-arguments) as $Rules(A)$, then such defeats are defined as follows

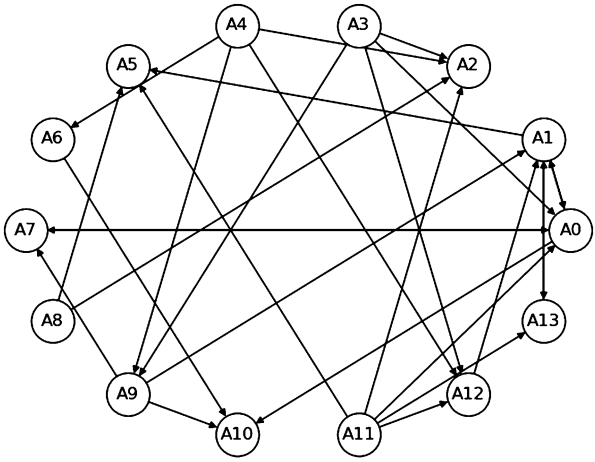

Fig. 9.2 The argument framework obtained when defeats are computed for the UAV example

Definition 7 Argument A defeats an argument B if A attacks B for all $r_A \in Rules(A), r_B \in Rules(B)$, it is the case that $r_A > r_B$.

In the above definition, it is assumed that a strict rule is preferred to all defeasible rules within an argument.

This condition for defeat captures the *weakest link* principle, computing the strength of an argument by considering its weakest rule. Furthermore, it complies with the *democratic* ordering principle, requiring that a single rule within the stronger argument be preferred over all rules within the weaker argument. A discussion of other principles can be found in [18].

Example 3 Figure 9.2 illustrates the abstract argument framework obtained when defeats are computed for the UAV example. The grounded extension for this argument framework is $\{A1, A3, A4, A7, A8, A10, A11\}$. One can therefore conclude that the UAV should land on $rw1$ rather than $rw2$ while making use of reserve fuel and not being overweight.

3 Proof Dialogues

While argumentation can be used to identify appropriate arguments which justify why some plan should be executed, we have not yet considered how such arguments should be presented to a user. In this section, we describe *proof dialogues*, which provide a dialogical approach to justifying arguments. Such a dialogical approach then naturally provides an explanation as to why an argument (and in turn a plan) is justified.

Proof dialogues seek to determine the status of a single argument, i.e., whether it does, or does not appear within an extension according to some semantics (or

alternatively what its labelling is). In the remainder of this chapter, we refer to this single argument as the focal argument. In the process of determining the status of the focal argument, the status of other arguments may also become apparent.

Since our focus lies in explaining why a plan was executed, we do not care about what could have been, or could be, but rather what was or is. Within the Scrutable Autonomous Systems project, we therefore concentrated on single extension semantics, namely the grounded and—to a lesser extent—sceptically preferred semantics. Less attention was paid to the latter due to the computational complexity involved in computing the status of arguments under this semantics [2, 24].

In this section we revisit the proof dialogue described in [17] for the grounded semantics, in order to illustrate how such dialogues operate. In the next section, we then consider more advanced proof dialogues which (we argue) are better able to provide explanation than this dialogue. We discuss this point further below.

Proof dialogues are usually represented as a discussion between two players, **Pro**, who wishes to demonstrate that the focal argument appears within the extension under the given semantics, and **Con**, who wishes to demonstrate otherwise.

For an argument to appear within a grounded extension, it must be defended by other arguments within the extension, but cannot (directly or indirectly) defend itself. This suggests the following structure for a proof dialogue where **Pro** and **Con** alternate in advancing arguments.

Opening move: **Pro** introduces the focal argument.

Dialogue moves: If the length of the dialogue is odd (i.e., it is **Con**'s move) then **Con** must introduce an argument that attacks the last argument introduced. Otherwise, if the length of the dialogue is even (i.e., it is **Pro**'s move), then **Pro** must introduce an argument that attacks the last introduced argument, but cannot introduce an argument that they have already introduced. If a player cannot make a move, then the dialogue terminates.

Dialogue termination: The last person to be able to make a move is the winner of the dialogue.

It has been shown that if an argument is in the grounded extension, then there is a sequence of arguments that **Pro** can advance (i.e., a strategy) to win the dialogue.

Perhaps the most significant disadvantage of the dialogue game described above is that they do not describe how a winning strategy may be found. If such a dialogue is used for explanation, and a non-winning strategy is used, than the explanation generated will not be appropriate. We therefore describe an alternative dialogue game for computing whether a focal argument is in the grounded extension, together with an appropriate strategy. This dialogue game was originally introduced in [3], and is referred to as the *Grounded Discussion Game*, abbreviated GDG.

Participants within the game can make four different moves, defined as follows.

– *HTB(A)* stating that "*A* has to be the case". This move, made by **Pro**, claims that *A* has to be labelled **IN** within the legal labelling.

– *CB(B)* stands for "B can be the case". This move, made by **Con**, claims that B does not necessarily have to be labelled **OUT**.
– *CONCEDE(A)* allows **Con** to agree that A has to be the case.
– *RETRACT(B)* allows **Con** to agree that B must be labelled **OUT**.

The game starts with the proponent making a HTB statement about the focal argument. In response, **Con** utters one or more CB, CONCEDE, or RETRACT statements. **Pro** makes a further HTB statement in response to a CB move. The precise conditions for each move are as follows:

– HTB(A) is the first move. Alternatively, the previous move was CB(B), and A attacks B.
– CB(A) is moved when A attacks the last HTB(B) move made by **Pro**; A has not been retracted, and no CONCEDE or RETRACT move is applicable.
– CONCEDE(A) is moved if HTB(A) was moved previously, all attackers of A have been retracted, and this move was not yet played.
– RETRACT(A) is moved if **Con** made a CB(A) move in the past which has not yet been retracted, and A has an attacker B for which the move CONCEDE(B) was played.

An additional condition is that HTB and CB moves cannot be repeated (to prevent the dialogue going around in circles), and HTB and CB cannot be played for the same argument.

Pro wins the game if **Con** concedes the focal argument while **Con** wins if they make a CB move to which **Pro** cannot respond.

Caminada [3] demonstrates a strategy for this game which is sound and complete for the grounded semantics. That is, **Pro** will win the game if and only if the focal argument is in the grounded extension, and **Con** will win otherwise.

Example 4 Continuing our running example, a user might question whether the UAV ends up using reserve fuel (i.e., whether $A10$ is in the grounded extension). The dialogue could then proceed as illustrated in Table 9.3.

Comparing Table 9.3 with Fig. 9.2, the primary advantage of proof dialogues over the standard labelling-based approaches becomes apparent. Proof dialogues allow for the incremental presentation of arguments which are relevant to the user's interests, while ignoring arguments which the user accepts (by not having the user query such arguments), or are not central to the explanation. By operating in this way, proof dialogues mitigate against information overload and allow the user to "drill down" to where the explanation is necessary.

The dialogue above illustrates one weakness of many argumentation based explanation dialogues, namely that preferences are (normally) treated as meta features which induce defeats between arguments. The dialogue can therefore not explain these preferences directly. Techniques for overcoming this issue are discussed in Sect. 5.

Table 9.3 Sample dialogue for the UAV example

Pro	HTB(A10)	"The UAV uses reserve fuel as it lands on $rw1$"
	CB(A0)	"We know that by default, we can just land on rw2"
Pro	HTB(A3)	"But not being overweight means we must land on runway 1"
Con	CONCEDE(A3)	"I accept that"
	RETRACT(A0)	"And that for that reason, we can't land on runway 2"
	CB(A9)	"But could it not be the case that no reserve fuel is used as it doesn't land on $rw1$"?
	CB(A6)	"After all, by default, no reserve fuel is used"
Pro	HTB(A4)	"But we know that the UAV is not overweight, and therefore can't land on $rw2$". Not landing on $rw2$ means it lands on $rw1$, and therefore uses reserve fuel"
Con	CONCEDE(A4)	"I accept that line of argument"
	RETRACT(A9)	"And retract what I said"
	RETRACT(A6)	

4 Putting it all Together: The SAsSy Demonstrator

Figure 9.3 shows a screenshot of the prototype plan explanation tool developed as part of the Scrutable Autonomous Systems project. In this section, we provide a brief overview of this tool, its strengths, and its limitations.

Underpinning the tool were plans expressed as YAML workflows.[2] Such workflows contain choice points with regards to actions, and decisions as to which action to pursue were made—by the system—through argument-based reasoning. More specifically, a domain model made up of strict and defeasible rules was constructed. From this model, arguments could be generated, and extensions computed. The conclusions of arguments within the extension then identified which actions should be selected when choices existed (c.f., the UAV running example).

As illustrated in the screenshot, the tool's user interface consisted of three main portions. At the top, a visual display of the plan was shown. A textual summary of the plan (or portions of the plan) appears on the bottom left, while an area wherein a user can interact with the system via dialogue appears on the bottom right.

[2] https://www.commonwl.org/user_guide/.

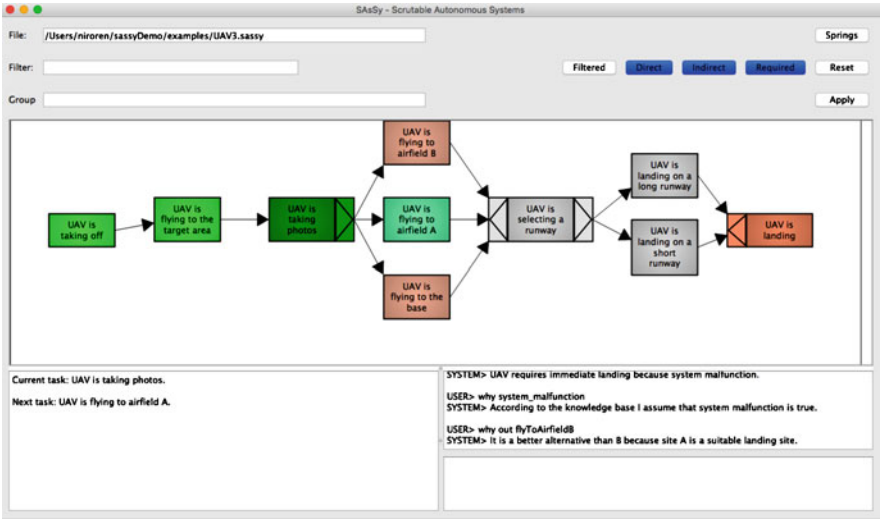

Fig. 9.3 A screenshot from the plan explanation tool

4.1 Plan Visualisation

The plan visualisation window provides a simple view of the plan, showing the ordering between tasks, actions which can be executed in parallel, and the like.

Actions within a plan are executed by different entities, and may affect various resources. Different filtering options were provided to the user, allowing them—for example—to highlight or hide only those actions which affect a specific resource. In [26], the authors show that such highlighting techniques reduce the number of errors and improve response times for users of the system when considering small and medium sized plans. Surprisingly, however, while highlighting relevant portions of the plan led to improved performance, the hiding of unimportant parts of the plan did not lead to improved performance by the user. In addition, questions remain as to whether these results carry through to larger plans than those investigated by [26].

4.2 Natural Language Generation

While plan visualisation using action labels may provide users with important insight, natural language descriptions of these actions can sometimes be easier for such users to understand. Such descriptions may be offered in isolation or—as is often preferable—in addition to graphs, and this is where Natural Language Generation (NLG) comes in.

In some cases, NLG can be accomplished via a simple language realisation toolkit such as SimpleNLG [12], or using template-based techniques as in [8]. This approach works well for the information in our running example (e.g., the bottom left window in Fig. 9.3). As the plan is filtered, the summary changes to reflect only the portions relevant to the user. Similarly, as the plan executes, the natural language summary describes only the current action to be executed, omitting irrelevant information.

Complicated tasks such as planning, however, pose difficult additional challenges, particularly when plans become large (i.e., containing many steps) or structurally complex (e.g., with choices or parallel paths), in which cases NLG needs to find suitable ways to summarise what would otherwise become an unwieldy list of lists. In such cases, the NLG-generated text may start with a high-level summary saying "This plan consist of a large number of actions, which need to be performed in parallel," before going into further detail.

Furthermore, the generator needs to avoid misunderstandings. For example, the text "But could it not be the case that no reserve fuel is used as it doesn't land on rw1"?" contains some syntactic ambiguities that might be misconstrued. Misunderstandings are also known to arise when English expressions such as "if .. then" are employed to express a logical construct such as the material implication (i.e., the standard "arrow" of FOPL), and finding better alternatives automatically is not always easy.

Finally, plans are often the result of automated theorem proving, where formulas in First-Order Predicate Logic (FOPL), or more complicated logics, are manipulated to find the solution for a planning problem. In these cases, the dialogue needs to inform the user that some action a was chosen (or that some action b, which the user may have suggested, is not feasible) because of some logic proposition p. The problem is that, frequently, the system expresses p in a form that may seem unwieldy to human users, for instance because of background knowledge that they possess. (For example, p may say that a crane is at location loc_1 and *not* at location loc_2, where the latter part is redundant because a crane can only be at one location at a time.) Thus, NLG faces the challenge of having to *"optimise" p* before any standard NLG techniques can be applied.

4.3 Dialogue Based Plan Explanation

The bottom right portion of the prototype allows users to interact with the system through dialogue, asking why the system believes certain facts do, or do not hold. If a query asks why some conclusion holds, the system initiates a proof dialogue taking on the role of **Pro**, while if the user asks why some conclusion does not hold, then the system takes on the role of **Con**. A simple domain specific language allows the user to participate in the dialogue, and arguments are presented to the user as natural language rather than logical formulae.

The dialogue language also allows the user to assert or delete facts within the knowledge base, enabling them to update the system with new knowledge. If the user changes the facts within the knowledge base, the system will determine if any of its actions need to be changed, and will allow the dialogue to restart. The system can change the user's beliefs (by presenting them with justified arguments), and allows the user to change the system's beliefs (via assertions and deletions) through the same natural dialogue based interface.

It should be noted that within the prototype, the grounded persuasion game of [5] was used as the proof dialogue. However, this latter game is not sound, and—unlike the proof dialogue described above—does not allow both participants to introduce arguments, a feature which is undesirable in some instances [3].

5 Discussion and Related Work

This chapter described an argument-based system for explaining plans. The planning domain was encoded using YAML, and argumentation was used to select actions where choices existed. Other researchers have described how classical planning techniques can be recreated using argumentation. For example, [10] provided an algorithm for performing partial order planning in defeasible logic programming, while Pardo [20] and others [19] described how dialogue can be used to perform multi-agent planning. Several researchers [14, 27] have examined how BDI agent programs (which bear strong similarities to HTN planning) can be explained via simple dialogues. The focus of this strand of work involves using argument and dialogue to drive the planning process. The techniques described in this paper can then be used to explain how the plan was generated, and why other plans were not selected, by advancing arguments for the plan and demonstrating attacks against other plans' arguments.

The focus of this work was on dialogue games for the grounded semantics. This semantics is sceptical, selecting arguments which—in some sense—must be justified, and can be contrasted with the preferred semantics, which identify sets of arguments that could be justified together. In the context of explaining planning, selecting a sceptical semantics appears correct, as it is only possible to have selected a single alternative for execution [21]. However, other sceptical semantics do exist, such as the sceptical preferred semantics, which select those arguments lying in the intersection of all preferred extensions. It has been argued that this latter semantic is more natural for humans [22], and dialogue games for identifying the sceptical preferred extension have been proposed [24]. One downside of this semantics is however the computational complexity involved in computing it [2], which may make it unfeasible for large domains.

Argumentation and dialogue appear to be natural techniques for explanation, and is increasingly being used in the context of explainable AI research [13]. Psychologists have claimed that humans innately reason using argument [15], while computer scientists have shown that formal argumentation agrees with

human intuition [6, 22] in most cases. As future work, we intend to evaluate the effectiveness of our approach to explaining plans through human experimentation. In addition, we intend to overcome two of the current limitations of our approach, namely the meta-logical nature of preferences, and the lack of temporal concepts in our argumentation system.

One approach to addressing this first limitation could be to use an extended argumentation system [16] which encodes preferences as arguments. Addressing the second limitation may be possible through the use of timed argumentation frameworks [7], which explicitly include temporal concepts. However, both these systems have been described only in abstract terms, and it will therefore be necessary to create a structured instantiation of them. Doing so will allow us to provide more refined explanations about more aspects of the plans. Finally, we will also consider how explanations can be provided for richer planning languages such as PDDL.

6 Conclusions

In this chapter we described how plans can be explained through the use of a dialogue game, where participants take turns to make utterances which are used to establish whether some argument (and therefore its conclusions) is justified. To use our technique, domain rules describing the plan must be transformed to arguments, which we achieved through the use of the ASPIC− formalism. Finally, we described how these plan visualisation and natural language generation, together with dialogue based explanation, can be used to create a tool to explain plans to non-technical users.

While our approach appears promising, a complete evaluation with users is still required. In addition, it suffers from shortcomings with regards to explaining preferences and temporal concepts, suggesting a clear path for future work.

Acknowledgements This work was supported by the Engineering and Physical Sciences Research Council (EPSRC, UK), grant ref. EP/J012084/1 (“Scrutable Autonomous Systems”).

References

1. P. Baroni, M. Caminada, and M. Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011.
2. P. Baroni and M. Giacomin. *Semantics of Abstract Argument Systems*, pages 25–44. Springer US, Boston, MA, 2009.
3. M. Caminada. A discussion game for grounded semantics. In *International Workshop on Theory and Applications of Formal Argumentation*, pages 59–73. Springer, 2015.
4. M. Caminada, S. Modgil, and N. Oren. Preferences and unrestricted rebut. In *Proceedings of the 2014 conference on Computational Models of Argument*, pages 209–220, 2014.

5. M. Caminada and M. Podlaszewski. Grounded semantics as persuasion dialogue. In *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, volume 245, pages 478–485. IOS Press, 2012.
6. F. Cerutti, N. Tintarev, and N. Oren. Formal arguments, preferences and natural language interfaces to humans: an empirical evaluation. In *Proc. ECAI*, pages 207–212, 2014.
7. M. L. Cobo, D. C. Martínez, and G. R. Simari. On admissibility in timed abstract argumentation frameworks. In *ECAI*, volume 215, pages 1007–1008, 2010.
8. K. V. Deemter, M. Theune, and E. Krahmer. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24, 2005.
9. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
10. D. R. García, A. J. García, and G. R. Simari. Defeasible reasoning and partial order planning. In *Proceedings of the 5th International Conference on Foundations of Information and Knowledge Systems*, FoIKS'08, pages 311–328, Berlin, Heidelberg, 2008. Springer-Verlag.
11. A. Gatt and E. Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.
12. A. Gatt and E. Reiter. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 90–93, 2009.
13. D. Gunning, Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency, DARPA/I20*, (DARPA, 2017).
14. V. Koeman, L. A. Dennis, M. Webster, M. Fisher, and K. Hindriks. 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, 2019.
15. H. Mercier and D. Sperber. *The enigma of reason*. Harvard University Press, 2017.
16. S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9–10):901–934, 2009.
17. S. Modgil and M. Caminada. *Proof Theories and Algorithms for Abstract Argumentation Frameworks*, chapter 6. Springer, 2009.
18. S. Modgil and H. Prakken. The ASPIC+ framework for structured argumentation: a tutorial. *Argument and Computation*, 5(1):31–62, 2014.
19. S. Pajares and E. Onaindia. Temporal defeasible argumentation in multi-agent planning. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three*, IJCAI'11, pages 2834–2835. AAAI Press, 2011.
20. P. Pardo, S. Pajares, E. Onaindia, L. Godo, and P. Dellunde. Multiagent argumentation for cooperative planning in DeLP-POP. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 971–978. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
21. H. Prakken. Combining sceptical epistemic reasoning with credulous practical reasoning. *COMMA*, 144:311–322, 2006.
22. I. Rahwan, I. Madakkatel, M., J. Bonnefon, R. N. Awan, and S. Abdallah. Behavioral experiments for assessing the abstract argumentation semantics of reinstatement. *Cognitive Science*, 34(8):1483–1502, 2010.
23. E. Reiter and R. Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997.
24. Z. Shams and N. Oren. A two-phase dialogue game for skeptical preferred semantics. In *JELIA*, volume 10021 of *Lecture Notes in Computer Science*, pages 570–576, 2016.
25. N. Tintarev, R. Kutlak, J. Masthoff, K. Van Deemter, N. Oren, and W. W. Vasconcelos. Adaptive visualization of plans. In *UMAP Workshops*, 2014.
26. N. Tintarev and J. Masthoff. Effects of individual differences in working memory on plan presentational choices. *Frontiers in Psychology*, 7:1793, 2016.

27. M. Winikoff. Debugging agent programs with why? Questions. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 251–259. International Foundation for Autonomous Agents and Multiagent Systems, 2017.
28. Y. Wu, M. Caminada, and M. Podlaszewski. A labelling-based justification status of arguments. *Studies in Logic*, 3(4):12–29, 2010.