Argument & Computation 10 (2019) 149–189
DOI 10.3233/AAC-190447
IOS Press

Acquiring knowledge from expert agents in a structured argumentation setting

Ramiro Andres Agis *, Sebastian Gottifredi and Alejandro Javier García
Institute for Computer Science and Engineering (UNS-CONICET), Department of Computer Science and Engineering, Universidad Nacional del Sur, Bahía Blanca, Argentina
E-mails: ramiro.agis@cs.uns.edu.ar, sg@cs.uns.edu.ar, ajg@cs.uns.edu.ar

Abstract. Information-seeking interactions in multi-agent systems are required for situations in which there exists an expert agent that has vast knowledge about some topic, and there are other agents (questioners or clients) that lack and need information regarding that topic. In this work, we propose a strategy for automatic knowledge acquisition in an information-seeking setting in which agents use a structured argumentation formalism for knowledge representation and reasoning. In our approach, the client conceives the other agent as an expert in a particular domain and is committed to believe in the expert's qualified opinion about a given query. The client's goal is to ask questions and acquire knowledge until it is able to conclude the same as the expert about the initial query. On the other hand, the expert's goal is to provide just the necessary information to help the client understand its opinion. Since the client could have previous knowledge in conflict with the information acquired from the expert agent, and given that its goal is to accept the expert's position, the client may need to adapt its previous knowledge. The operational semantics for the client-expert interaction will be defined in terms of a transition system. This semantics will be used to formally prove that, once the client-expert interaction finishes, the client will have the same assessment the expert has about the performed query.

Keywords: Information-seeking, argumentation, defeasible logic programming

1. Introduction

In multi-agent systems, agents can have different aims and goals, and it is normal to assume that there is no central control over their behaviour. One of the advantages of these systems is that the information is decentralised. Hence, the agents have to interact in order to obtain the information they need, or to share part of their knowledge.

In this work, we propose a strategy for automatic knowledge acquisition which involves two different kinds of agents: one agent that has expertise in a particular domain or field of knowledge and a client agent that lacks that quality. In our approach, the client agent will initially make a query to the expert agent in order to acquire some knowledge about a topic it does not know about or partially knows about. Since the client conceives the other agent as an expert, it will be committed to believe in the answer for its query. Unlike other approaches in the literature, we consider that the client may have previous strict knowledge that is in contradiction with the information the expert knows about the consulted topic. Hence, the client may require to ask further questions and adapt its previous knowledge in order to be aligned with what the expert says.

*Corresponding author. E-mail: ramiro.agis@cs.uns.edu.ar.

A naive solution to the proposed problem would be for the expert to send its whole knowledge base to the client. However, this is not a sensible nor feasible solution for several reasons. First, depending on the application domain, the expert could have private information that is sensitive and should not be shared. Second, its knowledge base could be very extensive and merging it with the client's could be computationally impracticable in a real-time and dynamic environment. Finally, the merged knowledge bases would probably have many contradictions whose relevance is outside of the domain of the query. Ignoring these inconsistencies would lead to undesired results and conclusions, but solving them would be time-consuming and irrelevant for the query. Another solution would be for the client to revise its initial knowledge base to believe in the expert's opinion in a single step. However, as will be shown in the following sections, this may imply the unnecessary removal of pieces of information that, from the expert's perspective, are valid.

In [47], the concept of *information-seeking dialogues* was introduced, in which one participant is an expert in a particular domain or field of knowledge and the other is not. By asking questions, the non-expert participant elicits the expert's opinion (advice) on a matter in which the questioner itself lacks direct knowledge. The questioner's goal is to accept the expert's opinion while the expert's goal is to provide just the necessary information to help the questioner understand its opinion about the consulted topic. In this particular type of dialogue, the questioner can arrive at a presumptive conclusion which gives a plausible expert-based answer to its question. Information-seeking has already been addressed in literature when defining dialogue frameworks for agents. However, some of these approaches do not consider that the questioner may have previous strict knowledge in contradiction to the expert's [31], while others, which consider such a possibility, simply disregard conflicting interactions [17–19].

Differently from existing approaches, our proposal not only considers that agents may have previous strict knowledge in contradiction, but also focuses on a strategy which guarantees that the information-seeking goals are always achieved. That is, once a client-expert interaction finishes, the client agent will believe the same as the expert agent about the initial query. Since the client conceives the other agent as an expert, whenever a contradiction between their knowledge arises the client will always prefer the expert's opinion. However, in order to avoid the unnecessary removal of pieces of information that – from the expert's perspective – are valid, the client will keep asking questions to the expert until the goal is achieved.

In order to provide a dialogue protocol specification that satisfies the aforementioned goals, one of the main contributions of our proposal is a definition of the operational semantics in terms of a transition system. Although we will formalise a two-agent interaction, this strategy can be applied in a multi-agent environment in which an expert agent could have several simultaneous interactions with different clients – each one in a separate *session*.

The research on the use of argumentation to model agent interactions is a very active field, including argumentation-based negotiation approaches [3,26,27,36], persuasion [6,12,32,33], general dialogue formalizations [17], strategic argumentation [12,43,44], among others. In our proposal, agents will be equipped with the structured argumentation reasoning mechanism of DeLP (Defeasible Logic Programming) [22]. DeLP allows the involved agents to represent tentative or weak information in a declarative manner. Such information is used to build arguments, which are sets of coherent information supporting claims. The acceptance of a claim will depend on an exhaustive dialectical analysis (formalised through a proof procedure) of the arguments in favour of and against it. This procedure provides agents with an inference mechanism for warranting their entailed conclusions. We will use the structured argumentation formalism DeLP for knowledge representation and reasoning since the purpose of this paper is to show how to solve the problems associated to the agents' argument structures in an information-seeking

setting. In particular, DeLP has been used to successfully implement *inquiries* [1,42,45], another type of dialogue defined by [47]. In contrast to other approaches that use argumentation as a tool for deciding among dialogue moves, similarly to [4,8] we use structured argumentation just as the underlying representation formalism for the involved agents.

There is plenty of work on revision of argumentation frameworks (AFs) [11,13,14,30,38] which, regardless of their individual goals, end up adding or removing arguments or attacks and returning a new AF or set of AFs as output. Our proposal differs from all those approaches in that the client agent will not just revise its "initial framework" in order to warrant the expert's opinion. Instead, the client will keep asking questions and acquiring knowledge from the expert agent (that is relevant to the initial query) in order to avoid removing from its knowledge base pieces of information that, from the expert's perspective, are valid. As will be explained in detail in the following sections, in order to be able to believe in the expert's qualified opinion, the client will only revise its previous knowledge if it is in contradiction with the expert's. In other words, our proposal differs from other approaches in that unnecessary modifications are avoided by maintaining the communication with the expert, with the additional benefit of acquiring more relevant knowledge and making informed changes considering a qualified opinion.

It has been recognised in the literature [22,37] that the argumentation mechanism provides a natural way of reasoning with conflicting information while retaining much of the process a human being would apply in such situations. Thus, defeasible argumentation provides an attractive paradigm for conceptualising common-sense reasoning, and its importance has been shown in different areas of Artificial Intelligence such as multi-agent systems [10], recommender systems [5], decision support systems [23], legal systems [34], agent internal reasoning [2], multi-agent argumentation [42,45], agent dialogues [7,8], among others (see [37]). In particular, DeLP has been used to equip agents with a qualitative reasoning mechanism to infer recommendations in expert and recommender systems [9,24,41]. Below, we introduce an example to motivate the main ideas of our proposal.

Example 1 (Motivating example)**.** Consider an agent called `M` that is an expert in the stock market domain, and a client agent called `B` (the client) that consults `M` for advice. Suppose that `B` asks `M` whether to buy stocks from the company Acme. The agent `M` is in favour of buying Acme's stocks and answers with the following argument: "Acme has announced a new product, then there are reasons to believe that Acme's stocks will rise; based on that, there are reasons to buy Acme's stocks". The client `B` has to integrate the argument shared by `M` into its own knowledge in order to be able to infer the same conclusion drawn by the expert. In the particular case that the client has no information at all about the topic – or at least no information in conflict with the expert's argument – it will simply add all the provided information to its own knowledge. However, it could occur that the client has previous knowledge about the query's topic. Consider that `B` can build the following argument: "Acme is in fusion with the company Steel and generally being in fusion makes a company risky; usually, I would not buy stocks from a risky company." Clearly, this argument built by `B` is in conflict with the conclusion of the one received from `M`. In order to solve the conflict and believe in the expert's opinion, a naive solution for `B` would be to delete from its knowledge all the conflictive pieces of information without further analysis. Nevertheless, following that solution, valuable information could be unnecessarily lost. However, `B` can follow a different approach: continue with the interaction and send the conflictive argument to `M` to give the expert the opportunity to return its opinion. Now consider that `M` already knows `B`'s argument but has information that defeats it. Then, `M` sends `B` a new argument that defeats `B`'s: "Steel is a strong company, and being in fusion with a strong company gives reasons to believe that Acme is not risky." Finally, `B` can adopt both arguments sent by `M` and then, after the interaction and without losing information, `B` can infer exactly what `M` has advised.

It is important to note that, in the example above, the expert M could have much more knowledge (related or not to the topic in question) that will not be sent to B. As will be explained in more detail below, in our proposal the expert will just send the necessary information that the client needs to infer the answer to its query. Examples of different situations that can arise during the client-expert interaction will be introduced along the presentation of the paper. The contributions of this paper are:

- A strategy for information-seeking in an argumentative setting – defined in terms of a transition system – in which agents use DeLP for knowledge representation and reasoning.
- Results that formally prove that the agents always achieve the information-seeking goals.
- Two different approaches which the expert can take to minimise – under some assumption – or reduce – using the client's previous knowledge – the information exchange.
- An extension to the operational semantics, which allows the client to reject the expert's qualified opinion, hence relaxing the assumption that the client is committed to believe in the expert.

The rest of this work is organised as follows. In Section 2 we introduce the background related to the agents' knowledge representation and reasoning formalism. Then, in Section 3, we explain the client-expert interaction in detail, and we define the operational semantics of the proposed strategy using transition rules. Next, in Section 4, we define some operators that the expert can use to minimise or reduce the information exchange with the client. Section 5 follows with an extension to the operational semantics that allows the client to reject the expert's opinion, relaxing the assumption of commitment. Then, in Section 6 we discuss on some design choices of our formalism. Next, in Section 7, related work is included. Finally, in Section 8, we present conclusions and comment on future lines of work. At the end of the paper we include an Appendix with the proofs for the formal results of our approach.

2. Knowledge representation and reasoning

In this section, the background related to the agents' knowledge representation and reasoning is included. In our approach, both the expert and the client represent their knowledge using DeLP, a language that combines results of Logic Programming and Defeasible Argumentation [21]. As in Logic Programming, DeLP allows to represent information declaratively using facts and rules. A DeLP program consists of a finite set of facts and defeasible rules. *Facts* are used for the representation of irrefutable evidence, and are denoted with ground literals: that is, either atomic information (*e.g.*, *has_new_product*(*acme*)), or the negation of atomic information using the symbol "$\sim$" of strong negation (*e.g.*, $\sim$*in_fusion*(*magma*)). In turn, *defeasible rules* are used for the representation of tentative information, and are denoted $L_0 \prec L_1, \ldots, L_n$, where L_0 (the head of the rule) is a ground literal and $\{L_i\}_{i>0}$ (the body) is a set of ground literals. A defeasible rule *"Head* $\prec$ *Body"* establishes a weak connection between "*Body*" and "*Head*" and can be read as *"reasons to believe in the antecedent Body give reasons to believe in the consequent Head"*. When required, a DeLP program $\mathcal{P}$ will be noted as (Π, Δ), distinguishing the subset Π of facts and the subset Δ of defeasible rules. Although defeasible rules are ground, following the usual convention proposed in [28] we will use "schematic rules" with variables denoted with an upper-case letter. The other language elements (literals and constants) will be denoted with an initial lower-case letter. Given a literal q, the complement of q with respect to "$\sim$" will be denoted $\overline{q}$, *i.e.*, $\overline{q} = \sim q$ and $\overline{\sim q} = q$. An example of a DeLP program follows:

Example 2. Consider the motivating example introduced above. The DeLP program $(\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}})$ represents the knowledge of the expert agent M.

$$\Pi_{\mathtt{M}} = \left\{ \begin{array}{lll} \mathit{in\_fusion}(\mathit{acme}, \mathit{steel}) & \sim\mathit{in\_fusion}(\mathit{magma}) & \mathit{strong\_co}(\mathit{steel}) \\ \mathit{new\_co}(\mathit{starter}) & \mathit{has\_new\_product}(\mathit{acme}) & \mathit{has\_new\_owner}(\mathit{acme}) \\ \mathit{stock\_was\_dropping}(\mathit{acme}) & \sim\mathit{stock\_stable}(\mathit{magma}) & \end{array} \right\}$$

$$\Delta_{\mathtt{M}} = \left\{ \begin{array}{l} \mathit{buy\_stocks}(X) \prec \mathit{stock\_will\_rise}(X) \\ \mathit{stock\_will\_rise}(X) \prec \mathit{has\_new\_product}(X) \\ \sim\mathit{buy\_stocks}(X) \prec \mathit{risky\_co}(X) \\ \mathit{risky\_co}(X) \prec \mathit{in\_fusion}(X, Y) \\ \sim\mathit{risky\_co}(X) \prec \mathit{in\_fusion}(X, Y), \mathit{strong\_co}(Y) \\ \sim\mathit{stock\_will\_rise}(X) \prec \mathit{stock\_was\_dropping}(X) \\ \mathit{sell\_stocks}(X) \prec \mathit{closing}(X) \\ \sim\mathit{sell\_stocks}(X) \prec \mathit{new\_co}(X) \\ \mathit{be\_cautious}(X) \prec \mathit{tech\_co}(X), \mathit{new\_co}(X) \end{array} \right\}$$

In the set $\Pi_{\mathtt{M}}$ there are eight facts that represent evidence that the expert M has about the stock market domain (for instance: "Acme and Steel are in fusion", "Magma is not in fusion", "Steel is a strong company", "Starter is a new company", "Acme has announced a new product"). The set $\Delta_{\mathtt{M}}$ has nine (schematic) defeasible rules that M can use to infer tentative conclusions. Note that the first two rules will allow to infer *buy_stocks*(*acme*), the third and fourth to infer $\sim$*buy_stocks*(*acme*), and the fifth to infer $\sim$*risky_co*(*acme*). The last four defeasible rules represent knowledge that M has, but were not sent to B in the motivating example because they were not relevant with respect to the queries that B made.

We will briefly include below some concepts related to the argumentation inference mechanism of DeLP. We refer to [21] for the details. In a valid DeLP program the set Π must be non-contradictory. Since in this proposal Π is a set of facts, this means that Π cannot have a pair of contradictory literals, and this should be considered every time the client adopts new knowledge from the expert. In DeLP a ground literal L will have a *defeasible derivation* from a program (Π, Δ) if there exists a finite sequence of ground literals $L_1, L_2, \ldots, L_n = L$, where L_i is a fact in Π, or there exists a rule R_i with head L_i and body $B_1, B_2, \ldots, B_m$ such that every literal of the body is an element L_j of the sequence appearing before L_i $(j < i)$. Since strong negation can appear in the head of defeasible rules, complementary literals can be defeasibly derived. For the treatment of contradictory knowledge, DeLP uses an argumentative inference mechanism which identifies the pieces of knowledge that are in contradiction, and then uses a *dialectical process* for deciding which conclusions prevail as warranted. This process involves the construction and evaluation of arguments that are either for or against the query under analysis.

Definition 1 (Argument)**.** Let h be a literal, and $\mathcal{P} = (\Pi, \Delta)$ a DeLP program. We say that $\mathcal{A} = \langle \mathsf{R}, h \rangle$ is an argument for h if:

(1) $\mathsf{R} \subseteq \Delta$, and
(2) there exists a defeasible derivation for h from $\Pi \cup \mathsf{R}$, and
(3) $\Pi \cup \mathsf{R}$ is non-contradictory (*i.e.*, no pair of contradictory literals can be defeasibly derived from $\Pi \cup \mathsf{R}$), and
(4) R is minimal: there is no proper subset R' of R such that R' satisfies conditions (2) and (3).

Given $\mathcal{A} = \langle \mathsf{R}, h \rangle$ we say that h is the conclusion of $\mathcal{A}$, denoted $\mathsf{claim}(\mathcal{A}) = h$. For instance, in $(\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}})$ from Example 2, $\mathcal{M}_1$ is an argument for the literal *buy_stocks*(*acme*), and $\mathcal{M}_2$ is an argument for the literal *stock_will_rise*(*acme*). Both $\mathcal{M}_1$ and $\mathcal{M}_2$ have rules that are instances of the schematic rules of $\Delta_{\mathtt{M}}$.

$$\mathcal{M}_1 = \left\langle \left\{ \begin{array}{l} buy\_stocks(acme) \prec stock\_will\_rise(acme) \\ stock\_will\_rise(acme) \prec has\_new\_product(acme) \end{array} \right\}, buy\_stocks(acme) \right\rangle$$

$$\mathcal{M}_2 = \langle \{ stock\_will\_rise(acme) \prec has\_new\_product(acme) \}, stock\_will\_rise(acme) \rangle$$

The set of literals used to build an argument is called *evidence* and is defined as follows:

Definition 2 (Evidence)**.** Let $\mathcal{A} = \langle \mathsf{R}, h \rangle$ be an argument, $\mathsf{heads}(\mathcal{A})$ be the set of literals that appear at the head of rules in R, and $\mathsf{bodies}(\mathcal{A})$ be the set of all literals that appear at the body of rules in R. The *evidence* used to build $\mathcal{A}$ is defined as $\mathsf{evidence}(\mathcal{A}) = (\mathsf{bodies}(\mathcal{A}) \cup \{\mathsf{claim}(\mathcal{A})\}) \setminus \mathsf{heads}(\mathcal{A})$.

The evidence of $\mathcal{A}$ is $\mathsf{bodies}(\mathcal{A}) \setminus \mathsf{heads}(\mathcal{A})$ when the set of rules is not empty, and is $\{\mathsf{claim}(\mathcal{A})\}$ otherwise. For instance, $\mathsf{bodies}(\mathcal{M}_1) = \{has\_new\_product(acme), stock\_will\_rise(acme)\}$, $\mathsf{heads}(\mathcal{M}_1) = \{stock\_will\_rise(acme), buy\_stocks(acme)\}$, and $\mathsf{evidence}(\mathcal{M}_1) = \{has\_new\_product(acme)\}$. Note that $\langle \emptyset, strong\_co(steel) \rangle$ is an argument for *strong_co*(*steel*), and $\mathsf{evidence}(\langle \emptyset, strong\_co(steel) \rangle) = \{strong\_co(steel)\}$. Given $\mathcal{S} = \langle \mathsf{R}_1, h_1 \rangle$ and $\mathcal{A} = \langle \mathsf{R}_2, h_2 \rangle$, $\mathcal{S}$ is a *sub-argument* of $\mathcal{A}$ when $\emptyset \subset \mathsf{R}_1 \subseteq \mathsf{R}_2$ or when $\mathsf{R}_1 = \emptyset$ and $\mathsf{claim}(\mathcal{S}) \in \mathsf{evidence}(\mathcal{A})$. In particular, every argument is a sub-argument of itself. For instance, $\mathcal{M}_2$ is a sub-argument of $\mathcal{M}_1$, and $\langle \emptyset, has\_new\_product(acme) \rangle$ is a sub-argument of $\mathcal{M}_2$. If an argument $\mathcal{A}$ has more than one rule, then the head of each rule represents an intermediate reasoning step and it corresponds to the conclusion of a sub-argument of $\mathcal{A}$. The following four arguments can also be built from $(\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}})$:

$$\mathcal{M}_3 = \left\langle \left\{ \begin{array}{l} \sim buy\_stocks(acme) \prec risky\_co(acme) \\ risky\_co(acme) \prec in\_fusion(acme, steel) \end{array} \right\}, \sim buy\_stocks(acme) \right\rangle$$

$$\mathcal{M}_4 = \langle \{ risky\_co(acme) \prec in\_fusion(acme, steel) \}, risky\_co(acme) \rangle$$

$$\mathcal{M}_5 = \langle \{ \sim risky\_co(acme) \prec in\_fusion(acme, steel), strong\_co(steel) \}, \sim risky\_co(acme) \rangle$$

$$\mathcal{M}_6 = \langle \{ \sim stock\_will\_rise(acme) \prec stock\_was\_dropping(acme) \}, \sim stock\_will\_rise(acme) \rangle$$

$\mathcal{M}_3$ and $\mathcal{M}_1$ are in conflict since they have contradictory conclusions, and so are $\mathcal{M}_5$ and $\mathcal{M}_4$, and $\mathcal{M}_6$ and $\mathcal{M}_2$. In DeLP, in order to solve conflicts, preferences among arguments can be provided in a modular way and hence, the most appropriate criterion for the application domain can be used. Since the argument comparison criterion is not the focus of our proposal, in the examples below we will assume that it is given as a relation denoted "$>$". That is, considering that $\text{ARGS}_{\mathcal{P}}$ is the set of all the arguments that can be obtained from a DeLP program $\mathcal{P}$, the relation $> \subseteq (\text{ARGS}_{\mathcal{P}} \times \text{ARGS}_{\mathcal{P}})$ establishes the preference among the arguments of $\text{ARGS}_{\mathcal{P}}$. We refer the interested reader to [20,21,41] where different argument preference criteria are described.

In this paper, agents will be characterised by three elements: $(\Pi, \Delta, >)$. The first element (Π) is a set of facts which represents the evidence that the agent has, whereas the second element (Δ) is a set of defeasible rules. Thus, from (Π, Δ) the agent will be able to build a set of arguments denoted $\text{ARGS}_{(\Pi,\Delta)}$. The third element $(>)$ is an argument comparison criterion, *i.e.*, $> \subseteq (\text{ARGS}_{(\Pi,\Delta)} \times \text{ARGS}_{(\Pi,\Delta)})$. For

instance, in Example 2 we assume that the expert M has the following preferences: $\mathcal{M}_5 >_{\mathtt{M}} \mathcal{M}_4$ and $\mathcal{M}_2 >_{\mathtt{M}} \mathcal{M}_6$. Preferences allow to decide which argument prevails between two conflicting arguments:

Definition 3 (Proper Defeater, Blocking Defeater)**.** Let $\mathcal{D} = \langle \mathsf{R}_1, h_1 \rangle$ and $\mathcal{A} = \langle \mathsf{R}_2, h_2 \rangle$ be two arguments. $\mathcal{D}$ is a proper defeater for $\mathcal{A}$ at literal h if and only if there exist a sub-argument $\mathcal{S} = \langle \mathsf{R}, h \rangle$ of $\mathcal{A}$ such that $\mathcal{D}$ is in conflict with $\mathcal{A}$ at h, and $\mathcal{D} > \mathcal{S}$ (*i.e.* $\mathcal{D}$ is preferred to $\mathcal{S}$). If $\mathcal{D} \not> \mathcal{S}$ and $\mathcal{S} \not> \mathcal{D}$ (*i.e.* $\mathcal{D}$ is unrelated by the preference order to $\mathcal{S}$) then $\mathcal{D}$ is a blocking defeater for $\mathcal{A}$.

Continuing with Example 2, for the agent M the argument $\mathcal{M}_5$ is a proper defeater for $\mathcal{M}_4$ because $\mathcal{M}_5 >_{\mathtt{M}} \mathcal{M}_4$, and $\mathcal{M}_3$ is a blocking defeater for $\mathcal{M}_1$ because $\mathcal{M}_3$ and $\mathcal{M}_1$ are in conflict and there is no preference between $\mathcal{M}_3$ and $\mathcal{M}_1$. On the contrary, $\mathcal{M}_6$ is not a defeater for $\mathcal{M}_1$ because $\mathcal{M}_2 >_{\mathtt{M}} \mathcal{M}_6$.

In DeLP, in order to determine whether a query h can be accepted as warranted from a program $\mathcal{P}$, first it is necessary to find out if at least one argument $\mathcal{A}$ for h can be constructed from $\mathcal{P}$. If such argument exists, then all the defeaters for $\mathcal{A}$ that can be constructed from $\mathcal{P}$ are considered as potential reasons against h. If there is one argument $\mathcal{A}$ for h and there are no defeaters for $\mathcal{A}$, then $\mathcal{A}$ and its conclusion h are warranted from $\mathcal{P}$. However, since defeaters are arguments, if one or more defeaters for $\mathcal{A}$ exists, then there may be defeaters for them, defeaters for those defeaters, and so on. Thus, for each defeater $\mathcal{D}$, all the defeaters for $\mathcal{D}$ that can be constructed from $\mathcal{P}$ must be considered. This leads to the generation of a tree structure called *dialectical tree* with root $\mathcal{A}$ in which every node – except for the root – is a defeater for its parent. Figure 1 further below depicts different examples of dialectical trees. We will distinguish proper and blocking defeaters with different types of arrows. An unidirectional arrow from an argument $\mathcal{D}$ to another argument $\mathcal{A}$ denotes that $\mathcal{D}$ is a proper defeater for $\mathcal{A}$. On the contrary, a bidirectional arrow from $\mathcal{D}$ to $\mathcal{A}$ denotes that $\mathcal{D}$ is a blocking defeater for $\mathcal{A}$.

Given a dialectical tree $\mathcal{T}$ for $\mathcal{A} = \langle \mathsf{R}, h \rangle$, every path from the root $\mathcal{A}$ to one leaf is called *argumentation line*, and each argumentation line represents a different sequence that alternates arguments in favour of h (called *pro* arguments) and arguments against h (called *con* arguments). That is, given an argumentation line $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \ldots]$ the set of pro arguments is $\Lambda_P = \{\mathcal{A}_1, \mathcal{A}_3, \mathcal{A}_5, \ldots\}$ and the set of con arguments is $\Lambda_C = \{\mathcal{A}_2, \mathcal{A}_4, \ldots\}$. A dialectical tree is considered acceptable when all its argumentation lines are acceptable (see Definition 4). Informally, that occurs when the argumentation lines satisfy certain constraints imposed on them to avoid infinite or circular argumentation and other undesirable situations. For instance, an argument cannot appear twice in the same argumentation line, and if an argument is defeated by a blocking defeater, that blocking defeater can only be defeated by a proper defeater. In addition, both the set of pro arguments and the set of con arguments must be *concordant*, that is, no contradictions must be derived when joining Π with all the defeasible rules from the arguments in Λ_P or when joining Π with all the defeasible rules from the arguments in Λ_C. We refer the interested reader to [21] and [22] for a complete explanation of those constraints.

Definition 4 (Acceptable Argumentation Line)**.** Let $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_i, \ldots, \mathcal{A}_n]$ be an argumentation line. Λ is an acceptable argumentation line iff:

(1) Λ is a finite sequence, and
(2) no argument $\mathcal{A}_k$ in Λ is a sub-argument of an argument $\mathcal{A}_i$ appearing earlier in Λ ($i < k$), and
(3) for every i such that the argument $\mathcal{A}_i$ is a blocking defeater for $\mathcal{A}_{i-1}$, if $\mathcal{A}_{i+1}$ exists then $\mathcal{A}_{i+1}$ is a proper defeater for $\mathcal{A}_i$, and
(4) both the set Λ_P of pro arguments and the set Λ_C of con arguments are concordant.

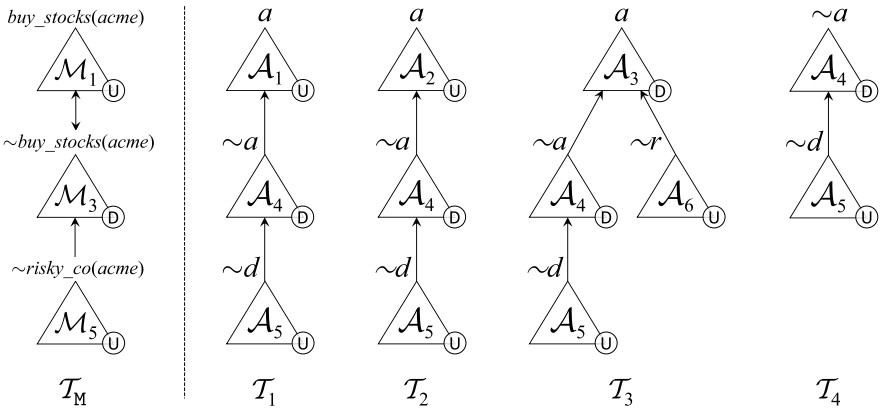

Fig. 1. Marked dialectical trees from Example 2 and Example 3.

Note that, given an acceptable dialectical tree, all the leaves of the tree are undefeated arguments. Therefore, to determine if $\mathcal{A}$ is either defeated or undefeated, the computation process of DeLP performs a marking of every node of the dialectical tree for $\mathcal{A}$ as follows: First, the leaves of the tree are marked as Ⓤ (undefeated). Then the inner nodes – including the root – are marked as Ⓓ (defeated) if they have at least one child marked as Ⓤ, or are marked as Ⓤ if all their children are marked as Ⓓ.[1]

Figure 1 (left) shows the marked dialectical tree $\mathcal{T}_{\mathtt{M}}$ that the expert $\mathtt{M}$ can build for the query *buy_stock*(*acme*). Observe that the root of $\mathcal{T}_{\mathtt{M}}$ is marked as Ⓤ. Given a query h, there can be more than one argument for h, and for each argument a different marked dialectical tree can be constructed. The next example shows how the expert agent proceeds in a scenario in which more than one dialectical tree for the same query can be constructed.

Example 3. Consider the DeLP program $(\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}})$:

$$\Pi_{\mathtt{E}} = \begin{Bmatrix} c & w & z \\ e & f & x \end{Bmatrix} \qquad \Delta_{\mathtt{E}} = \begin{Bmatrix} a \prec b & b \prec c & b \prec w, z & a \prec r & r \prec c, d & d \prec e \\ \sim a \prec d & \sim r \prec w & \sim d \prec p & p \prec f & \sim p \prec x & \end{Bmatrix}$$

Observe that from $(\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}})$ three different arguments for the conclusion "a" can be constructed: $\mathcal{A}_1 = \langle\{a \prec b; b \prec c\}, a\rangle$, $\mathcal{A}_2 = \langle\{a \prec b; b \prec w, z\}, a\rangle$ and $\mathcal{A}_3 = \langle\{a \prec r; r \prec c, d\}, a\rangle$. The arguments $\mathcal{A}_4 = \langle\{\sim a \prec d; d \prec e\}, \sim a\rangle$, $\mathcal{A}_5 = \langle\{\sim d \prec p; p \prec f\}, \sim d\rangle$, $\mathcal{A}_6 = \langle\{\sim r \prec w\}, \sim r\rangle$ and $\mathcal{A}_7 = \langle\{\sim p \prec x\}, \sim p\rangle$ can also be constructed from $(\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}})$. Consider the following preferences among these arguments: $\mathcal{A}_4 >_{\mathtt{E}} \mathcal{A}_1$, $\mathcal{A}_4 >_{\mathtt{E}} \mathcal{A}_2$, $\mathcal{A}_4 >_{\mathtt{E}} \mathcal{A}_3$, $\mathcal{A}_5 >_{\mathtt{E}} \mathcal{A}_4$, $\mathcal{A}_5 >_{\mathtt{E}} \mathcal{A}_7$, $\mathcal{A}_6 >_{\mathtt{E}} \mathcal{A}_3$.

In Fig. 1 we show four marked dialectical trees that can be constructed from $(\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$. In figures, nodes of the dialectical trees will be depicted with triangles with the argument's name inside, the argument's conclusion at the top vertex, and the mark (Ⓤ or Ⓓ) at the rightmost vertex. Given a marked dialectical tree $\mathcal{T}$ and a node $\mathcal{A}$ of $\mathcal{T}$, $\mathsf{root}(\mathcal{T})$ denotes the argument that is the root of $\mathcal{T}$, $\mathsf{rootmark}(\mathcal{T})$ denotes the mark assigned to $\mathsf{root}(\mathcal{T})$ and $\mathsf{mark}(\mathcal{A}, \mathcal{T})$ denotes the mark assigned to $\mathcal{A}$ in $\mathcal{T}$. For instance, in Fig. 1, $\mathsf{root}(\mathcal{T}_1) = \mathcal{A}_1$, $\mathsf{rootmark}(\mathcal{T}_1) =$ Ⓤ and $\mathsf{mark}(\mathcal{A}_4, \mathcal{T}_1) =$ Ⓓ. Given an argument $\mathcal{A}$ marked as

[1] This may resemble the argumentation game of Dung's [15] grounded semantics. However, they differ since DeLP does not allow arguments to be repeated within an argumentation line, whereas the grounded semantics only prevents pro arguments from being repeated. In this sense, the grounded semantics are more skeptical than DeLP's dialectical process.

Ⓓ in a dialectical tree $\mathcal{T}$, a *denier* for $\mathcal{A}$ is a defeater $\mathcal{D}$ for $\mathcal{A}$ marked as Ⓤ in $\mathcal{T}$. That is, $\mathcal{D}$ precludes $\mathcal{A}$ from being marked as Ⓤ in $\mathcal{T}$. For instance, in Fig. 1, $\mathcal{A}_6$ is a denier for $\mathcal{A}_3$ in the dialectical tree $\mathcal{T}_3$.

We say that a literal q (respectively an argument $\mathcal{A} = \langle \mathsf{R}, q \rangle$) is *warranted* by the agent $(\Pi, \Delta, >)$ if there exists at least one dialectical tree $\mathcal{T}$ for $\mathcal{A}$ constructed from $(\Pi, \Delta, >)$ such that $\mathsf{rootmark}(\mathcal{T}) = \text{Ⓤ}$ (*i.e.* $\mathcal{A}$ results undefeated). For instance, considering $(\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}})$ and $>_{\mathtt{E}}$ from Example 3, the agent $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$ warrants the literal a and the arguments $\mathcal{A}_1$ and $\mathcal{A}_2$, and does not warrant $\sim a$. Considering $(\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}})$ and $>_{\mathtt{M}}$ from Example 2, the agent $\mathtt{M} = (\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}}, >_{\mathtt{M}})$ warrants the literal *buy_stocks*(*acme*) and the argument $\mathcal{M}_1$.

Given a DeLP program $\mathcal{P}$ and a literal q, $\mathsf{dtrees}(\mathcal{P}, q)$ will denote the set of all dialectical trees that can be constructed from $\mathcal{P}$ such that the root's claim is the literal q. The set $\mathsf{U\text{-}dtrees}(\mathcal{P}, q) \subseteq \mathsf{dtrees}(\mathcal{P}, q)$ is defined as $\mathsf{U\text{-}dtrees}(\mathcal{P}, q) = \{\mathcal{T} \in \mathsf{dtrees}(\mathcal{P}, q) \mid \mathsf{rootmark}(\mathcal{T}) = \text{Ⓤ}\}$. That is, $\mathsf{U\text{-}dtrees}(\mathcal{P}, q)$ contains all the dialectical trees from $\mathcal{P}$ that provide a warrant for q, and it will be empty if no warrant for q exists. Observe that $\mathsf{U\text{-}dtrees}(\mathcal{P}, \overline{q})$ includes all the dialectical trees from $\mathcal{P}$ that provide a warrant for $\overline{q}$. For instance, in Fig. 1, $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), a) = \{\mathcal{T}_1, \mathcal{T}_2\}$ and $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), \sim a) = \{\}$.

3. Client-expert interaction

The whole interaction between both agents is called a *session*, which starts with a query and finishes when the client believes in the expert's position. We assume that the expert's knowledge does not change during a session and that the client does not simultaneously maintain more than one session. If the client has specific personal information that represents a context for its query and would affect the expert's answer, then this contextual information is sent before the session starts.

Once a new session has started, its current state will be determined by a *session state*. Intuitively, a session state is a structure that keeps track of all the elements that both agents had already communicated, and also other elements that the client needs to ask about and remain pending. As will be explained in detail below, those pending elements will allow the client to acquire the knowledge that it needs to understand the expert's position while avoiding unnecessary loss of information. In the rest of the paper, we will distinguish the *client agent* with $\mathtt{C} = (\Pi_{\mathtt{C}}, \Delta_{\mathtt{C}}, >_{\mathtt{C}})$ and *expert agent* with $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$. As stated above, the expert should know in advance all the relevant specific personal information from the client in order to give a proper answer. Since our approach focuses on the changes made on the client's knowledge base, we assume that this specific contextual information is already considered in $(\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$. Session states will be used to formalise the elements that actually change during the client-expert interaction, and thus the client agent $\mathtt{C}$ will be part of the 5-tuple of the session states whereas the expert agent $\mathtt{E}$ will not.

Definition 5 (Session State)**.** Let $\mathtt{C} = (\Pi_{\mathtt{C}}, \Delta_{\mathtt{C}}, >_{\mathtt{C}})$ be a client agent and "q" a query. A session state for "q" is a 5-tuple $(\mathtt{C}, \mathbb{I}, \mathbb{F}, \mathbb{P}, \mathbb{D})_q$ where $\mathbb{I}$ is a sequence of interaction elements, $\mathbb{F}$ is a set of arguments, $\mathbb{P}$ is a set of preference questions, and $\mathbb{D}$ is a set of denier questions.

When there is no ambiguity, session states will be simply referred to as "states", and the query subscript q will be omitted. The first component in a state's 5-tuple is the client agent $\mathtt{C}$ that made the query q. $\mathtt{C}$ is characterised by its knowledge and preferences among arguments $(\Pi_{\mathtt{C}}, \Delta_{\mathtt{C}}, >_{\mathtt{C}})$, which can change from state to state during the client-expert interaction. Although the other four components will be formalised and explained in detail further below, their intuitive meaning are introduced next: $\mathbb{F}$ is a set of arguments received by $\mathtt{C}$ which are all in favour of the claim of the *justification* sent by

the expert. The goal of the client is to leave all the arguments in $\mathbb{F}$ marked as ⓤ. $\mathbb{P}$ is a set of pending *preference questions*, which are pairs of arguments $(\mathcal{A}, \mathcal{B})$ representing the question "What do you think about the preference between $\mathcal{A}$ and $\mathcal{B}$?". $\mathbb{D}$ is a set of pending *denier questions*, which are arguments for which the client needs defeaters from the expert. Finally, $\mathbb{I}$ is a sequence that gathers all the interaction elements, composed by preference questions and denier questions that were previously asked by C, and by defeaters and preferences that were previously sent by the expert.

Note that once a query q is posed to an expert agent, one of the following three alternatives could occur: (1) the expert believes in q (*i.e.* q is warranted by E), (2) the expert believes in the opposite (*i.e.* $\overline{q}$ is warranted by E), or (3) the expert does not believe in q nor $\overline{q}$ (*i.e.* q and $\overline{q}$ are not warranted by E). In our approach, a session will only exist if one of the first two alternatives hold, that is, the expert has enough knowledge for a warrant for either q or $\overline{q}$. When this occurs, the expert's answer for the client's query will be either an argument for q or an argument for $\overline{q}$. This argument is called *justification* and will be denoted $\mathcal{J}$. As will be proved further below, when the session which started by the query q finishes, $\mathcal{J}$ will be warranted by the client agent, and hence, the client will believe in $\mathsf{claim}(\mathcal{J})$.

Figure 2 shows a directed graph that outlines how session states could evolve during a client-expert interaction. In that graph, nodes – identified with Greek letters – represent different session states of the interaction, while the directed arcs represent transitions among session states. All these transitions will be formally specified with transition rules during the rest of this section. For instance, the transition rule t_1 will specify how the session evolves from the initial state (α) into a new state (β) in which the

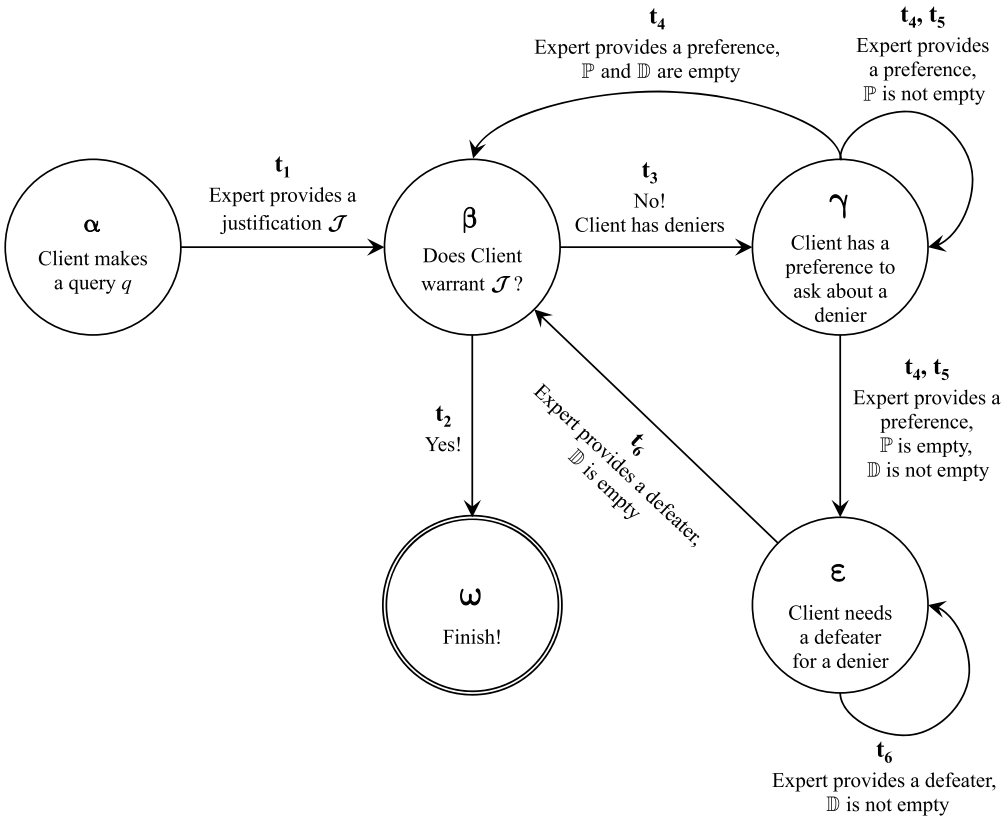

Fig. 2. Session evolution outline.

client has adopted information sent by the expert, whereas the transition rule t_2 will specify the necessary conditions for the session to reach the final state. We will refer to Fig. 2 along the rest of this section in order to explain all the details of our approach.

First, two special session states will be distinguished: the initial state and the final state. Observe that a session starts (see α in Fig. 2) when a client agent $\mathtt{C}$ makes a query q to the expert agent – and, if necessary, sends specific contextual information from $\Pi_{\mathtt{C}} \cup \Delta_{\mathtt{C}}$ (see Section 6). Hence, an initial state contains the elements that characterise $\mathtt{C}$ before receiving anything from the expert, *i.e.* $(\Pi_{\mathtt{C}}, \Delta_{\mathtt{C}}, >_{\mathtt{C}})$, and the other four components are empty as follows: $(\mathtt{C}, [\,], \emptyset, \emptyset, \emptyset)_q$

A final session state (see ω in Fig. 2) will have at least one interaction element in the tuple's second component, and the last three components will be empty (*i.e.* all the pending issues will have been solved). The final state's first component ($\mathtt{C}'$) will characterise the client agent with all the knowledge that it will have adopted from the expert by the end of the session: $(\mathtt{C}', [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)_q$

Once a session has started, the expert has to consider the initial query q made by the client and send one argument $\mathcal{J}$ which justifies its answer. As was shown in Section 2, the expert may have more than one dialectical tree whose root is suitable to be sent to the client as the justification $\mathcal{J}$ for the submitted query. Formally, the set $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), q)$ (or either $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), \overline{q})$) can have more than one element. For instance, consider the example depicted in Fig. 1 and suppose that the client has sent the query a. Then, $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), a) = \{\mathcal{T}_1, \mathcal{T}_2\}$, and thus the expert has two dialectical trees' roots as possible candidates to be the justification $\mathcal{J}$. Since there can exist different strategies for selecting one element, our approach is defined in a modular way and we assume the existence of an operator select-dtree that selects one dialectical tree from a set of dialectical trees. Therefore, the most appropriate implementation for the dialectical tree selection could be used depending on the represented application domain. For instance, the expert could select – among the roots of the dialectical trees – the most relevant argument in that domain, or the strongest argument with respect to the used comparison criterion. The analysis and comparison of different implementations for select-dtree is out of the scope of this paper and is left as future work (see Section 8). Hence, the justification $\mathcal{J}$ sent by the expert agent for a query q is determined by the operator expert-justification introduced next:

Definition 6 (Expert Justification)**.** Let $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$ be the expert agent, and "q" be a query. $\mathtt{E}$'s justification for q is defined as:

$$\mathsf{expert\text{-}justification}(q) = \begin{cases} \mathsf{root}(\mathsf{select\text{-}dtree}(\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), q))), & \text{if } q \text{ is warranted by } \mathtt{E} \\ \mathsf{root}(\mathsf{select\text{-}dtree}(\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), \overline{q}))), & \text{if } \overline{q} \text{ is warranted by } \mathtt{E} \end{cases}$$

Example 4. Consider the expert agent $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$ introduced above in Example 3. Let us suppose that $\mathtt{E}$ receives the query "a". The expert warrants the literal "a" since there exist two dialectical trees whose root arguments claim "a" and are marked as Ⓤ: $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}), a) = \{\mathcal{T}_1, \mathcal{T}_2\}$, and both $\mathcal{A}_1$ and $\mathcal{A}_2$ are suitable as justification. In this case, we will assume that the selected argument is $\mathsf{expert\text{-}justification}(q) = \mathcal{A}_1$, and hence, $\mathcal{A}_1$ is sent to the client agent. Note that if the expert had received the query "$\sim a$", the justification would also have been $\mathcal{A}_1$ or $\mathcal{A}_2$ because "$\sim a$" would not have been warranted by the expert.

As was explained above, the client's goal is to accept the expert's opinion, and hence, when the interaction is finished the client should believe the same as the expert about the submitted query. Therefore, when the client $\mathtt{C}$ receives the expert's justification $\mathcal{J}$, its goal is to be able to warrant $\mathsf{claim}(\mathcal{J})$ from its

knowledge base (Π_C, Δ_C). In order to achieve this, the client will first *adopt* the received argument $\mathcal{J}$. Adopting an argument will consist in making the minimum necessary changes to (Π_C, Δ_C) in order to be able to construct $\mathcal{J}$. The client will then add rules and facts to its knowledge base and, in some cases, it will have to withdraw from Π_C the elements which are inaccurate from the expert's point of view. The adoption of an argument is determined by the operator argument-adoption introduced next:

Definition 7 (Argument Adoption)**.** Given a client agent $C = (\Pi_C, \Delta_C, >_C)$ and an argument $\mathcal{A} = \langle R, h\rangle$. The adoption of $\mathcal{A}$ by C is defined as argument-adoption$(C, \mathcal{A}) = (\Pi'_C, \Delta'_C, >_C)$, where:

$\Pi'_C = (\Pi_C \cup \text{evidence}(\mathcal{A})) \setminus (X \cup Y \cup Z)$, where

$X = \{x : \overline{x} \in \text{evidence}(\mathcal{A})\}$,
$Y = \{y : \overline{y} \prec b_1, \ldots, b_n \in R\}$ and
$Z = \{z : z \prec b_1, \ldots, b_n \in R\}$.

$\Delta'_C = \Delta_C \cup R$.

Consider a client agent $(\Pi_C, \Delta_C, >_C)$ that adopts an argument $\mathcal{A} = \langle R, h\rangle$. On the one hand, all the defeasible rules of R will be added to Δ_C and the evidence of $\mathcal{A}$ will be added to Π_C. This added knowledge will allow the client to infer h. On the other hand, all the facts from Π_C that contradict the evidence from $\mathcal{A}$ or contradict the head of a rule in R will be erased from Π_C. Those erased elements would prevent the argument $\mathcal{A}$ from being built from the client's knowledge base (see Section 2). Also, due to the minimality constrain that an argument should satisfy, all the facts from Π_C that are a head of a rule in R are also removed. The proof of Proposition 1 included in the appendix shows in detail why the literals in the sets X, Y, Z have to be erased upon an argument adoption. The following two examples show how the adoption operator behaves in two different scenarios. In the first one, the client (*Kyle*) has nothing to erase. In the second one, the client (*Randy*) has to erase elements in order to adopt the received argument. Note that, given an argument that can be constructed by both the client and the expert, for convenience the same argument name will be given to both.

Example 5. Consider an agent Kyle, characterised by $(\Pi_{K_1}, \Delta_{K_1}, >_{K_1})$ where $>_{K_1} = \emptyset$ (see Fig. 3), who makes the query "a" to the expert agent E from Example 4. In response to this query, E sends the argument $\mathcal{A}_1 = \langle\{a \prec b, b \prec c\}, a\rangle$ as justification. In order to adopt $\mathcal{A}_1$, Kyle has to add the fact "c" (the evidence of $\mathcal{A}_1$) to Π_{K_1} and the defeasible rules "$a \prec b$" and "$b \prec c$" to Δ_{K_1} without the need to delete anything. Figure 3 shows the result of the argument adoption: argument-adoption$((\Pi_{K_1}, \Delta_{K_1}, >_{K_1}), \mathcal{A}_1) = (\Pi_{K_2}, \Delta_{K_2}, >_{K_1})$. In this case, the sets X, Y and Z from Definition 7 are all empty.

Example 6. Consider an agent Randy, characterised by $(\Pi_{R_1}, \Delta_{R_1}, >_{R_1})$ where $>_{R_1} = \emptyset$ (see Fig. 4), who makes the query "$\sim a$" to the expert agent E from Example 4. Randy receives $\mathcal{A}_1$ in response and, in order to adopt $\mathcal{A}_1$, adds both the fact "c" to Π_{R_1} and the defeasible rules "$a \prec b$" and "$b \prec c$" to Δ_{R_1}. However, Randy also needs to erase two facts that disallow $\mathcal{A}_1$ to be a valid argument: "$\sim b$", given that it is the complement of the head of the rule "$b \prec c$", and "$\sim c'$", given that it is the complement of the evidence "c". Figure 4 shows the result of the argument adoption: argument-adoption$((\Pi_{R_1}, \Delta_{R_1}, >_{R_1}), \mathcal{A}_1) = (\Pi_{R_2}, \Delta_{R_2}, >_{R_1})$. In this case, the sets X and Y from Definition 7 have the following elements: $X = \{\sim c\}$ and $Y = \{\sim b\}$.

The following proposition shows that, whenever the client agent adopts an argument from the expert agent, it is guaranteed that the client will be able to construct that argument. The proofs of all the formal results are in the Appendix section at the end of the paper.

$$\Pi_{\mathrm{K}_1} = \{e \quad x\} \qquad \Delta_{\mathrm{K}_1} = \{\sim a \prec d \quad d \prec e \quad \sim p \prec x\}$$
$$\Pi_{\mathrm{K}_2} = \{e \quad x \quad c\} \qquad \Delta_{\mathrm{K}_2} = \{\sim a \prec d \quad d \prec e \quad \sim p \prec x \quad a \prec b \quad b \prec c\}$$

Fig. 3. Client agent Kyle before and after adopting $\mathcal{A}_1$.

$$\Pi_{\mathrm{R}_1} = \{\sim b \quad \sim c \quad f\} \qquad \Delta_{\mathrm{R}_1} = \{g \prec f\}$$
$$\Pi_{\mathrm{R}_2} = \{f \quad c\} \qquad \Delta_{\mathrm{R}_2} = \{g \prec f \quad a \prec b \quad b \prec c\}$$

Fig. 4. Client agent Randy before and after adopting $\mathcal{A}_1$.

Proposition 1. *Let* $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$ *be the expert agent, and* $\mathcal{A}$ *an argument constructed by* E*, if* argument-adoption$(\mathtt{C}_1, \mathcal{A}) = \mathtt{C}_2$ *then* $\mathcal{A}$ *can be constructed by* $\mathtt{C}_2$.

As it will be explained below, the justification $\mathcal{J}$ sent by the expert may not be the only argument that the client will need to adopt from the expert in order to achieve its goal. Thus, the operator argument-adoption will be used whenever the client receives an argument from the expert. Another important property that holds in our approach is that, whenever the client adopts a new argument, it is still able to construct all the arguments that it adopted previously from the expert:

Proposition 2. *Let* $\mathcal{A}$ *be an argument constructed by the expert agent* E*, and* $\mathcal{B}$ *an argument constructed by both a client agent* $\mathtt{C}_1$ *and* E*, if* argument-adoption$(\mathtt{C}_1, \mathcal{A}) = \mathtt{C}_2$ *then* $\mathcal{B}$ *can be constructed by* $\mathtt{C}_2$.

In order to formally define our strategy and then prove its properties, the operational semantics of the interaction will be defined in terms of a transition system. A transition system is a set of transition rules for deriving transitions. In addition, a transition is a transformation of one session state into another. A transition rule has the form

$$t : \frac{\langle cond \rangle}{\langle current\_state \rangle \;\rightarrow\; \langle new\_state \rangle}$$

and can be read as "*if the session is in the state* $\langle current\_state \rangle$ *and condition* $\langle cond \rangle$ *holds, then the session evolves into the state* $\langle new\_state \rangle$". Next, the transition rule t_1 is introduced, which describes how the session evolves from the initial state (see α in Fig. 2) into a new state in which the client has adopted the expert's justification (see β in Fig. 2).

$$t_1 : \frac{\text{expert-justification}(q) = \mathcal{J} \;\wedge\; \text{argument-adoption}(\mathtt{C}, \mathcal{J}) = \mathtt{C}'}{(\mathtt{C}, [\,], \emptyset, \emptyset, \emptyset) \;\rightarrow\; (\mathtt{C}', [\mathcal{J}], \{\mathcal{J}\}, \emptyset, \emptyset)}$$

When the transition rule t_1 is executed, the justification $\mathcal{J}$ is added to the current session state's second component and, therefore, the first interaction element in the sequence will always be that argument. $\mathcal{J}$ is also added to the current state's third component (the set of received arguments in favour of the claim of $\mathcal{J}$) because it will need to be analysed by the client. For instance, in Example 5, the transition rule t_1 makes the session between *Kyle* and the expert evolve from the initial state $(\mathtt{K}_1, [\,], \emptyset, \emptyset, \emptyset)_a$ into the state $(\mathtt{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \emptyset)_a$. Consider also Example 6: since the expert sends *Randy* and *Kyle*

the same justification, then *Randy*'s session evolves from the state $(\mathtt{R}_1, [\ ], \emptyset, \emptyset, \emptyset)_{\sim a}$ into the state $(\mathtt{R}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \emptyset)_{\sim a}$. Nevertheless, as we will show below, given that both agents have different previous knowledge their sessions will evolve differently.

Recall that, in our approach, the client conceives the other agent as an expert and its goal is to believe in the expert's qualified opinion. Since the client may have previous knowledge that can be in conflict with the information acquired from the expert agent (*e.g.* it could build a defeater for $\mathcal{J}$), and its goal is to adopt all the expert knowledge, then the client may need to adapt its previous knowledge losing as little information as possible in order to be aligned with the expert's position. As we will show next, the involved agents will exchange arguments until the client has a warrant for $\mathcal{J}$ from its own knowledge. From the client's point of view, $\mathcal{J}$ will be the root of a dialectical tree $\mathcal{T}$ constructed from $(\Pi_\mathtt{C}, \Delta_\mathtt{C}, >_\mathtt{C})$ and, in order to warrant $\mathcal{J}$, the result of $\mathsf{rootmark}(\mathcal{T})$ has to be Ⓤ. Hence, whenever this mark is Ⓓ, the client should ask the expert for more information. This situation is captured by a session state $(\mathtt{C}, \mathbb{I}, \mathbb{F}, \emptyset, \emptyset)$ with $\mathbb{F} \neq \emptyset$ (see β in Fig. 2) in which the client agent will check what it thinks about the previously adopted arguments in $\mathbb{F}$ in order to determine the course of the session. That could be: either it can warrant $\mathcal{J}$ and then the session finishes (see transition t_2 to ω), or it still does not have a warrant for $\mathcal{J}$ and then it is necessary to ask subsequent questions to the expert agent (see transition t_3 to γ). In order to determine this, the client will *introspect* into its own knowledge to check the mark (Ⓤ or Ⓓ) of the previously adopted arguments.

Definition 8 (Introspection)**.** Given a client agent $\mathtt{C} = (\Pi_\mathtt{C}, \Delta_\mathtt{C}, >_\mathtt{C})$, and two arguments $\mathcal{R}$ and $\mathcal{A}$, let $\mathcal{T}$ be the dialectical tree constructed from $(\Pi_\mathtt{C}, \Delta_\mathtt{C}, >_\mathtt{C})$ such that $\mathsf{root}(\mathcal{T}) = \mathcal{R}$. C's introspection for $\mathcal{A}$ is defined as $\mathsf{introspection}(\mathtt{C}, \mathcal{R}, \mathcal{A}) = \mathsf{mark}(\mathcal{A}, \mathcal{T})$, if $\mathcal{A}$ is in $\mathcal{T}$. Otherwise, the introspection returns Ⓓ.

Note that introspections check the mark of an argument in a particular dialectical tree. As we mentioned above, the client is only interested in the mark of that argument in the dialectical tree whose root is the justification $\mathcal{J}$ sent by the expert. When no confusion arises, when we talk about the mark of an argument, we implicitly refer to the mark of that argument in the dialectical tree (of the client or the expert, accordingly) whose root is $\mathcal{J}$.

If the client makes an introspection to check the mark of the justification $\mathcal{J}$ (see β in Fig. 2) and results in Ⓤ, this implies that it has a warrant for the claim of $\mathcal{J}$ and, therefore, the session can end. This behaviour is reflected in the following transition rule:

$$t_2 : \frac{\mathsf{introspection}(\mathtt{C}, I_1, I_1) = \text{Ⓤ}}{(\mathtt{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \emptyset) \ \rightarrow \ (\mathtt{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)} \ n \geqslant 1$$

Recall that the first element in a session state's second component is always the expert's justification. If the transition rule t_2 is executed the current state's third component becomes empty, making the session evolve into the final state (see ω in Fig. 2). Example 7 illustrates a scenario in which the client does not have knowledge opposing the expert's justification. That argument is adopted and no other changes are needed in the client's knowledge in order to warrant it. In consequence, the session finishes.

Example 7. Consider again the client agent Randy $(\mathtt{R}_2)$ from Example 6. After adopting the expert's justification $\mathcal{A}_1$ the current session state is $(\mathtt{R}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \emptyset)_{\sim a}$. Then, Randy makes an introspection to check the mark of the justification. The only argument constructed by Randy is $\mathcal{A}_1 = \langle \{a \prec b, b \prec c\}, a \rangle$, and thus the only dialectical tree for "a" is the one depicted in Fig. 5

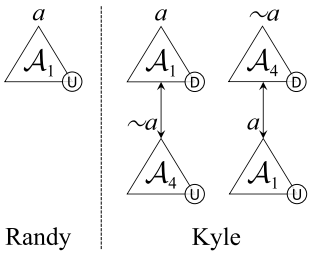

Fig. 5. Dialectical trees from Example 7 (left) and Example 8 (right).

(left). The result of introspection($\mathtt{R}_2, \mathcal{A}_1, \mathcal{A}_1$) is Ⓤ because $\mathcal{A}_1$ does not have any deniers. In consequence, all the arguments in the session state's third component are removed, and the session finishes after reaching the final state $(\mathtt{R}_2, [\mathcal{A}_1], \emptyset, \emptyset, \emptyset)_{\sim a}$.

In contrast to Example 7, the following example shows a scenario in which the client has a denier for the received justification. Hence, the result of the introspection would be Ⓓ and the session would not be able to end immediately. The client-expert interaction will continue with the client's goal of having a warrant for the received justification.

Example 8. Consider again the client agent Kyle ($\mathtt{K}_2$) from Example 5. After adopting the expert's justification $\mathcal{A}_1$ the current session state is $(\mathtt{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \emptyset)_a$. Then, Kyle makes an introspection to check the mark of the justification. The arguments constructed by agent Kyle are $\mathcal{A}_1 = \langle\{a \prec b, b \prec c\},\ a\rangle$ and $\mathcal{A}_4 = \langle\{\sim a \prec d, d \prec e\}, \sim a\rangle$. Now, let us suppose that it has no preference between them. In this case, Kyle's dialectical trees for "a" are the ones depicted in Fig. 5 (right). The result of introspection($\mathtt{K}_2, \mathcal{A}_1, \mathcal{A}_1$) is Ⓓ because $\mathcal{A}_1$ has a denier ($\mathcal{A}_4$). Therefore, unlike Randy in Example 7, Kyle's session does not reach the final state immediately after adopting the justification $\mathcal{A}_1$, because the transition rule t_2 is not applicable.

A denier (as $\mathcal{A}_4$ in Example 8) is an argument defeating another argument which, from the expert's position, is undefeated (as $\mathcal{A}_1$ in Example 8). Clearly the expert will not have deniers for the arguments that it sends to the client. However, since the client could have different knowledge, the client may construct a defeater that the expert cannot. In addition, the comparison criteria of both agents could differ. Finally, the expert could have a defeater for the denier that the client does not. Hence, in order to deal with its deniers, the client will first ask *preference questions* to the expert in order to adjust its preferences among its arguments. In DeLP the argument comparison criterion is modular and the most appropriate for the application domain being modelled can be used. Hence, we opted to formalize the preference between arguments as the relations $>_{\mathtt{C}}$ and $>_{\mathtt{E}}$ in order to abstract away from the details of the agents'comparison criteria. A discussion on how a particular comparison criterion can be applied to our approach can be found in Section 6. The operator preference-questions is defined as follows:

Definition 9 (Preference Questions)**.** Given a client agent $\mathtt{C} = (\Pi_{\mathtt{C}}, \Delta_{\mathtt{C}}, >_{\mathtt{C}})$, and two arguments $\mathcal{R}$ and $\mathcal{A}$, let $\mathcal{T}$ be the dialectical tree constructed from $(\Pi_{\mathtt{C}}, \Delta_{\mathtt{C}}, >_{\mathtt{C}})$ such that root($\mathcal{T}$) $= \mathcal{R}$. C's set of preference questions for $\mathcal{A}$ is defined as preference-questions($\mathtt{C}, \mathcal{R}, \mathcal{A}$) $= \{(\mathcal{B}, \mathcal{S}) \mid \mathcal{B}$ is a defeater for $\mathcal{A}$, and $\mathcal{S}$ is the sub-argument of $\mathcal{A}$ which $\mathcal{B}$ is in conflict with, and mark($\mathcal{B}, \mathcal{T}$) $=$ Ⓤ$\}$.

For instance, since in Example 8 $\mathcal{A}_4$ prevents $\mathcal{A}_1$ from being marked as Ⓤ, the client will obtain a set of preference questions preference-questions($\mathtt{K}, \mathcal{A}_1, \mathcal{A}_1$) $= \{(\mathcal{A}_4, \mathcal{A}_1)\}$ and then will ask to the expert

about its preference between both arguments. Similarly to introspections, the set of preference questions for a given argument is obtained from the dialectical tree whose root is the expert's justification. The following proposition states that, if an argument is marked as Ⓓ, the set of preference questions that can be generated for that argument is not empty.

Proposition 3. *Let* $\mathtt{C}$ *be a client agent,* $\mathcal{A}$ *an argument in a dialectical tree* $\mathcal{T}$*, and* $\mathcal{R} = \mathsf{root}(\mathcal{T})$*, if* $\mathsf{introspection}(\mathtt{C}, \mathcal{R}, \mathcal{A}) = \text{Ⓓ}$ *then* $\mathsf{preference\text{-}questions}(\mathtt{C}, \mathcal{R}, \mathcal{A}) \neq \emptyset$.

We mentioned above that the justification for the initial query may not be the only argument that the client will need to adopt during the session. As we will explain in detail below, other arguments – also in favour of the claim of the justification (called "pro arguments") – can be received. Then, if the client makes an introspection to check the mark of the justification (see β in Fig. 2) and does not result in Ⓤ, it means that some of these pro arguments are marked as Ⓓ. In this case, the client agent will proceed to ask preference questions for one of the furthest deniers from the root (see γ in Fig. 2). When this pro argument becomes marked as Ⓤ later in the session, the pro argument marked as Ⓓ above in the same argumentation line may also become marked as Ⓤ and so on, like a cascade effect. This behaviour is reflected in the following transition rule:

$$t_3 : \frac{\begin{array}{l} I_j \in \mathbb{F} \ \wedge\ \mathsf{introspection}(\mathtt{C}, I_1, I_j) = \text{Ⓓ} \ \wedge\ \mathsf{preference\text{-}questions}(\mathtt{C}, I_1, I_j) = \mathbb{P} \ \wedge \\ \textit{not exists } k > j \textit{ such that } I_k \in \mathbb{F} \ \wedge\ \mathsf{introspection}(\mathtt{C}, I_1, I_k) = \text{Ⓓ} \end{array}}{(\mathtt{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \emptyset) \ \rightarrow\ (\mathtt{C}, [I_1, \ldots, I_n], \mathbb{F}, \mathbb{P}, \emptyset)} \quad n \geqslant j \geqslant 1$$

The transition rule t_3 specifies how the set of preference questions $\mathbb{P}$ is generated in order to deal with those deniers. The rightmost (last) argument marked as Ⓓ in $[I_1, \ldots, I_n]$ is selected, and the corresponding set of preference questions is added to the session state's fourth component. For instance, consider again Example 8 in which the current session state is $(\mathtt{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \emptyset)_a$ after Kyle adopted the expert's justification $\mathcal{A}_1$. It was shown that $\mathsf{introspection}(\mathtt{K}_2, \mathcal{A}_1, \mathcal{A}_1) = \text{Ⓓ}$ and, since $\mathsf{preference\text{-}questions}(\mathtt{K}, \mathcal{A}_1, \mathcal{A}_1) = \{(\mathcal{A}_4, \mathcal{A}_1)\}$, the transition rule t_3 can be executed, making the session evolve into $(\mathtt{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \{(\mathcal{A}_4, \mathcal{A}_1)\}, \emptyset)_a$. That is, the new session state's fourth component contains a preference question that will be sent to the expert. In the following example the client has more than one denier to deal with.

Example 9. Consider a new scenario in which a client agent $\mathtt{C}_1$ is in the state $(\mathtt{C}_1, [\mathcal{B}_1], \{\mathcal{B}_1\}, \emptyset, \emptyset)$ after adopting the justification $\mathcal{B}_1$. Let us assume that $\mathtt{C}_1$ can also construct the arguments $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$, $\mathcal{B}_5$, $\mathcal{B}_6$, $\mathcal{B}_7$, where the first five arguments are defeaters for $\mathcal{B}_1$, and $\mathcal{B}_7$ is a defeater for $\mathcal{B}_6$. In addition, let us assume that the client's preferences are $\mathcal{B}_2 >_{\mathtt{C}_1} \mathcal{B}_1$, $\mathcal{B}_3 >_{\mathtt{C}_1} \mathcal{B}_1$, $\mathcal{B}_4 >_{\mathtt{C}_1} \mathcal{B}_1$ and $\mathcal{B}_6 >_{\mathtt{C}_1} \mathcal{B}_1$. Hence, when the client $\mathtt{C}_1$ makes an introspection to check the mark of $\mathcal{B}_1$, it constructs the dialectical tree depicted in Fig. 6 in which there are four deniers: $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$ (proper defeaters), and $\mathcal{B}_5$ (blocking defeater). Given that $\mathcal{B}_1$ is marked Ⓓ and is the last argument sent by the expert, the client will generate the set of preference questions for $\mathcal{B}_1$. The result of $\mathsf{preference\text{-}questions}(\mathtt{C}_1, \mathcal{B}_1, \mathcal{B}_1)$ is the set $\mathbb{P} = \{(\mathcal{B}_2, \mathcal{B}_1), (\mathcal{B}_3, \mathcal{B}_1), (\mathcal{B}_4, \mathcal{B}_1), (\mathcal{B}_5, \mathcal{B}_1)\}$ and, by the transition rule t_3, the session reaches the state $(\mathtt{C}_1, [\mathcal{B}_1], \{\mathcal{B}_1\}, \mathbb{P}, \emptyset)$.

Whenever a set of preference questions $\mathbb{P} = \{(\mathcal{A}_2, \mathcal{A}_1), \ldots, (\mathcal{A}_n, \mathcal{A}_1)\}$ is generated, the expert will answer them one by one by sending to the client its own preference between the corresponding pair

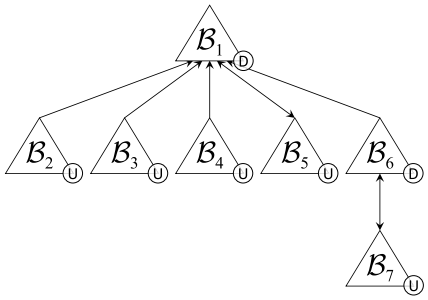

Fig. 6. Dialectical tree from Example 9.

of arguments. Then, the client will adopt such preference in order to deal with the deniers. Given a pair of arguments, the expert can either prefer the first over the second, prefer the second over the first, or believe that they are unrelated by the preference order, in which cases the expert will answer GRT (greater), LES (less), or UNR (unrelated), respectively. Given a set of preference questions $\mathbb{P} = \{(\mathcal{A}_2, \mathcal{A}_1), \ldots, (\mathcal{A}_n, \mathcal{A}_1)\}$ the expert agent always knows and is able to construct the second element of each pair because it corresponds to an argument that the client has previously received from the expert. However, the first argument in a preference question may not be considered as valid by the expert, either because it contains defeasible rules that are not in Δ_{E}, uses evidence that is not in Π_{E}, or simply because it cannot be constructed (it is non-minimal or uses evidence that is contradictory to Π_{E}). In this case, considering the assumptions we have made for the expert, it will always prefer an argument that it can construct over an argument that it does not consider as valid; and the answer will be LES (less). Then, the preference sent by the expert for a pair of arguments is determined by the operator expert-preference:

Definition 10 (Expert Preference)**.** Let $\mathrm{E} = (\Pi_{\mathrm{E}}, \Delta_{\mathrm{E}}, >_{\mathrm{E}})$ be the expert agent. E's preference between two arguments $\mathcal{A}$ and $\mathcal{B}$ is defined as:

$$\text{expert-preference}(\mathcal{A}, \mathcal{B}) = \begin{cases} \text{GRT}, & \text{if } (\mathcal{A}, \mathcal{B}) \in >_{\mathrm{E}} \\ \text{LES}, & \text{if } (\mathcal{B}, \mathcal{A}) \in >_{\mathrm{E}} \text{ or } \mathcal{A} \notin \text{ARGS}_{(\Pi_{\mathrm{E}}, \Delta_{\mathrm{E}})} \\ \text{UNR}, & \text{if } (\mathcal{A}, \mathcal{B}) \notin >_{\mathrm{E}}, (\mathcal{B}, \mathcal{A}) \notin >_{\mathrm{E}}, \text{ and } \mathcal{A} \in \text{ARGS}_{(\Pi_{\mathrm{E}}, \Delta_{\mathrm{E}})}. \end{cases}$$

When the client receives from the expert a preference between two arguments, it will adjust its own preferences accordingly. Similarly to an argument adoption, in a *preference adoption* the client respects the expert's opinion. The adoption of a preference is determined by the operator preference-adoption:

Definition 11 (Preference Adoption)**.** Given a client agent $\mathrm{C} = (\Pi_{\mathrm{C}}, \Delta_{\mathrm{C}}, >_{\mathrm{C}})$, a preference question $(\mathcal{A}, \mathcal{B})$, and a preference answer $P \in \{\text{GRT}, \text{LES}, \text{UNR}\}$. The preference adoption of P for $(\mathcal{A}, \mathcal{B})$ by C is defined as preference-adoption$(\mathrm{C}, (\mathcal{A}, \mathcal{B}), P) = (\Pi_{\mathrm{C}}, \Delta_{\mathrm{C}}, >'_{\mathrm{C}})$, where:

$$>'_{\mathrm{C}} = \begin{cases} >_{\mathrm{C}} \cup \{(\mathcal{A}, \mathcal{B})\}, & \text{if } P = \text{GRT} \\ (>_{\mathrm{C}} \setminus \{(\mathcal{A}, \mathcal{B})\}) \cup \{(\mathcal{B}, \mathcal{A})\}, & \text{if } P = \text{LES} \\ >_{\mathrm{C}} \setminus \{(\mathcal{A}, \mathcal{B})\}, & \text{if } P = \text{UNR} \end{cases}$$

Note that the adoption of new preferences may require to withdraw from $>_{\mathrm{C}}$ a preference between a pair of arguments which, from the expert's position, is inaccurate.

The transition rules t_4 and t_5 will specify how the session evolves once a received preference is adopted. On the one hand, t_4 is applicable when the adopted preference directly solves the problem with the denier. In this case, the corresponding preference question $(\mathcal{A}, \mathcal{B})$ is removed from $\mathbb{P}$. On the other hand, t_5 is applicable when the received preference is not enough to solve the problem with the denier. In this case, as with t_4 the pair $(\mathcal{A}, \mathcal{B})$ is removed from $\mathbb{P}$, but in addition $\mathcal{A}$ is added to the set of *denier questions* $\mathbb{D}$. Every denier question $\mathcal{A}$ is an argument for which the client needs an undefeated argument $\mathcal{D}$ from the expert such that $\mathcal{D}$ defeats $\mathcal{A}$.

$$t_4: \frac{\begin{array}{c}(\mathcal{A},\mathcal{B}) \in \mathbb{P} \ \wedge\ \textsf{expert-preference}(\mathcal{A},\mathcal{B}) = P \ \wedge \\ \textsf{preference-adoption}(\mathtt{C},\mathcal{A},\mathcal{B},P) = \mathtt{C}' \ \wedge\ \textsf{introspection}(\mathtt{C}', I_1, \mathcal{A}) \neq \textcircled{U}\end{array}}{(\mathtt{C},[I_1,\ldots,I_n],\mathbb{F},\mathbb{P},\mathbb{D}) \ \rightarrow\ (\mathtt{C}',[I_1,\ldots,I_n,(\mathcal{A},\mathcal{B}),P],\mathbb{F},\mathbb{P}\setminus\{(\mathcal{A},\mathcal{B})\},\mathbb{D})}\ n \geqslant 1$$

$$t_5: \frac{\begin{array}{c}(\mathcal{A},\mathcal{B}) \in \mathbb{P} \ \wedge\ \textsf{expert-preference}(\mathcal{A},\mathcal{B}) = P \ \wedge \\ \textsf{preference-adoption}(\mathtt{C},\mathcal{A},\mathcal{B},P) = \mathtt{C}' \ \wedge\ \textsf{introspection}(\mathtt{C}', I_1, \mathcal{A}) = \textcircled{U}\end{array}}{(\mathtt{C},[I_1,\ldots,I_n],\mathbb{F},\mathbb{P},\mathbb{D}) \ \rightarrow\ (\mathtt{C}',[I_1,\ldots,I_n,(\mathcal{A},\mathcal{B}),P],\mathbb{F},\mathbb{P}\setminus\{(\mathcal{A},\mathcal{B})\},\mathbb{D}\cup\{\mathcal{A}\})}\ n \geqslant 1$$

Recall that, given a preference question $(\mathcal{A}, \mathcal{B})$ asked by the client, $\mathcal{B}$ is always undefeated from the expert's point of view and, consequently, either $\mathcal{A}$ is marked as Ⓓ or $\mathcal{A}$ is an invalid argument that is not part of the dialectical tree. After adopting the corresponding preference for $(\mathcal{A}, \mathcal{B})$ from the expert, the client will make an introspection to check the mark of $\mathcal{A}$. If $\mathcal{A}$ is not marked as Ⓤ then $\mathcal{A}$ is no longer a denier (transition rule t_4). On the contrary, if $\mathcal{A}$ is marked as Ⓤ (*i.e.*, $\mathcal{A}$ is still a denier for $\mathcal{B}$) the client will need from the expert an undefeated defeater $\mathcal{C}$ for $\mathcal{A}$ (transition rule t_5). $\mathcal{C}$ must exist because otherwise – from the expert's point of view – $\mathcal{B}$ would not be an undefeated argument in favour of the justification. Note that the transition rules t_4 and t_5 will be executed until $\mathbb{P}$ is empty (see γ in Fig. 2).

Consider again the Example 8 in which the transition rule t_3 was executed and the session reached the state $(\mathtt{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \{(\mathcal{A}_4, \mathcal{A}_1)\}, \emptyset)_a$. Here, although $\mathbb{P}$ is not empty and $\textsf{expert-preference}((\mathcal{A}_4, \mathcal{A}_1)) = \textsc{grt}$, the transition rule t_4 cannot be executed because $\textsf{introspection}(\mathtt{K}_2, \mathcal{A}_1, \mathcal{A}_4) = \textcircled{U}$. Nevertheless, t_5 is applicable and the session evolves into $(\mathtt{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \{\mathcal{A}_4\})_a$.

Example 10. Let us suppose that the client agent $\mathtt{C}_1$ from Example 9 is in the session state $(\mathtt{C}_1, [\mathcal{B}_1], \{\mathcal{B}_1\}, \{(\mathcal{B}_2, \mathcal{B}_1), (\mathcal{B}_3, \mathcal{B}_1), (\mathcal{B}_4, \mathcal{B}_1), (\mathcal{B}_5, \mathcal{B}_1)\}, \emptyset)$ with the expert agent after generating the preference questions for $\mathcal{B}_1$. Recall that the client's preferences are $\mathcal{B}_2 >_{\mathtt{C}_1} \mathcal{B}_1$, $\mathcal{B}_3 >_{\mathtt{C}_1} \mathcal{B}_1$, $\mathcal{B}_4 >_{\mathtt{C}_1} \mathcal{B}_1$ and $\mathcal{B}_6 >_{\mathtt{C}_1} \mathcal{B}_1$, and also let us assume that the expert's preferences are $\mathcal{B}_5 >_{\mathtt{E}} \mathcal{B}_1$, $\mathcal{B}_8 >_{\mathtt{E}} \mathcal{B}_2$, $\mathcal{B}_9 >_{\mathtt{E}} \mathcal{B}_5$, $\mathcal{B}_{10} >_{\mathtt{E}} \mathcal{B}_5$, $\mathcal{B}_{11} >_{\mathtt{E}} \mathcal{B}_{10}$ and $\mathcal{B}_1 >_{\mathtt{E}} \mathcal{B}_4$. Both agents' (relevant) dialectical trees during this part of the session are depicted in Fig. 7. First, the client asks the preference question $(\mathcal{B}_2, \mathcal{B}_1)$ and the expert answers UNR. Then, the client removes $\mathcal{B}_2 >_{\mathtt{C}_1} \mathcal{B}_1$ (see 1 in Fig. 7) so $\mathcal{B}_2$ is now a blocking defeater instead of a proper defeater. Since the introspection for $\mathcal{B}_2$ results in Ⓤ, the transition rule t_5 is executed causing $\mathcal{B}_2$ to be added to the set of denier questions because the client (now $\mathtt{C}_2$) needs a defeater for $\mathcal{B}_2$ from the expert. Then, the session reaches the state $(\mathtt{C}_2, [\mathcal{B}_1, (\mathcal{B}_2, \mathcal{B}_1), \textsc{unr}], \{\mathcal{B}_1\}, \{(\mathcal{B}_3, \mathcal{B}_1), (\mathcal{B}_4, \mathcal{B}_1), (\mathcal{B}_5, \mathcal{B}_1)\}, \{\mathcal{B}_2\})$. Second, the client asks the preference question $(\mathcal{B}_3, \mathcal{B}_1)$ and the expert answers LES. The client removes $\mathcal{B}_3 >_{\mathtt{C}_2} \mathcal{B}_1$ (see 2), adds $\mathcal{B}_1 >_{\mathtt{C}_2} \mathcal{B}_3$, and makes an introspection for $\mathcal{B}_3$ which does not result in Ⓤ. Then, t_4 is executed and the session evolves into $(\mathtt{C}_3, [\mathcal{B}_1, (\mathcal{B}_2, \mathcal{B}_1), \textsc{unr}, (\mathcal{B}_3, \mathcal{B}_1), \textsc{les}], \{\mathcal{B}_1\}, \{(\mathcal{B}_4, \mathcal{B}_1), (\mathcal{B}_5, \mathcal{B}_1)\}, \{\mathcal{B}_2\})$. Third, the client asks the preference question $(\mathcal{B}_4, \mathcal{B}_1)$ and the expert answers LES. The client proceeds to remove $\mathcal{B}_4 >_{\mathtt{C}_3}$

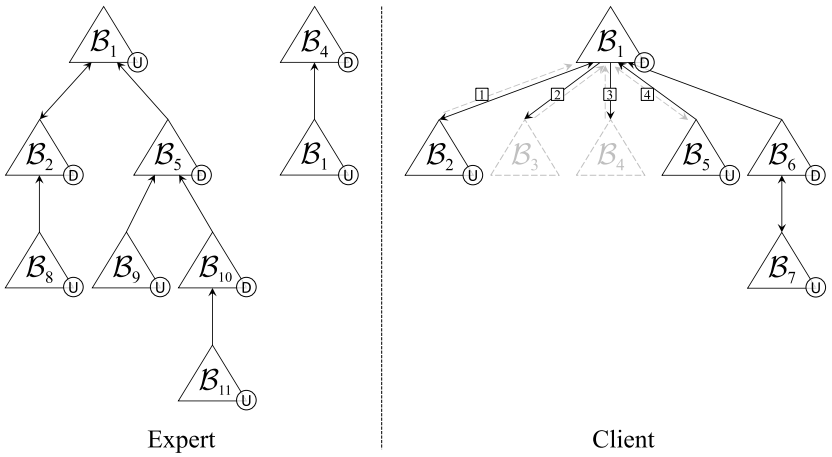

Fig. 7. Dialectical trees from Example 10.

$\mathcal{B}_1$ (see 3), adds $\mathcal{B}_1 >_{C_3} \mathcal{B}_4$, and makes an introspection for $\mathcal{B}_4$ which does not result in Ⓤ. Then, by t_4 the session evolves into $(C_4, [\mathcal{B}_1, (\mathcal{B}_2, \mathcal{B}_1), \text{UNR}, (\mathcal{B}_3, \mathcal{B}_1), \text{LES}, (\mathcal{B}_4, \mathcal{B}_1), \text{LES}], \{\mathcal{B}_1\}, \{(\mathcal{B}_5, \mathcal{B}_1)\}, \{\mathcal{B}_2\})$. Finally, the client asks the last preference question $(\mathcal{B}_5, \mathcal{B}_1)$ and the expert answers GRT. The client proceeds to add $\mathcal{B}_5 >_{C_4} \mathcal{B}_1$ (see 4) so $\mathcal{B}_5$ is now a proper defeater instead of a blocking defeater. Since the introspection for $\mathcal{B}_5$ results in Ⓤ, the transition rule t_5 is executed causing $\mathcal{B}_5$ to be added to the set of denier questions because the client (now C_5) needs a defeater for $\mathcal{B}_5$ from the expert. Then, the session reaches the state $(C_5, [\mathcal{B}_1, (\mathcal{B}_2, \mathcal{B}_1), \text{UNR}, (\mathcal{B}_3, \mathcal{B}_1), \text{LES}, (\mathcal{B}_4, \mathcal{B}_1), \text{LES}, (\mathcal{B}_5, \mathcal{B}_1), \text{GRT}], \{\mathcal{B}_1\}, \emptyset, \{\mathcal{B}_2, \mathcal{B}_5\})$. Note that due to the changes in the client's preferences ($\mathcal{B}_1 >_{C_5} \mathcal{B}_3$ and $\mathcal{B}_1 >_{C_5} \mathcal{B}_4$), the arguments $\mathcal{B}_3$ and $\mathcal{B}_4$ are no longer defeaters for $\mathcal{B}_1$, and thus they are not part of the client's dialectical tree for $\mathcal{B}_1$ any more. Those arguments are greyed out in Fig. 7.

Whenever a set of denier questions $\mathbb{D}$ is generated, the expert will answer them one by one by sending to the client an undefeated defeater for the corresponding denier. Similarly to the selection of the dialectical tree for the justification, the expert may have multiple undefeated defeaters for a denier and there are different strategies to select one of them. We will follow again the same modular approach as we have explained for the operator select-dtree used in Definition 6: we assume the existence of an operator select-defeater that the expert uses for selecting a suitable defeater. Therefore, the most appropriate implementation for select-defeater could be used depending on the represented application domain. For instance, the expert could select – among the undefeated defeaters for the denier – the most relevant argument in that domain, or the strongest argument with respect to the used comparison criterion, etc. As stated above for the operator select-dtree, the analysis and comparison of different implementations for select-defeater is left as future work.

Apart from the defeater, the expert will also send the *type* of the defeater (PROPER or BLOCKING) so that the client can adjust the preference between that argument and the corresponding denier in advance. The defeater sent by the expert agent for a denier is determined by the operator expert-defeater:

Definition 12 (Expert Defeater)**.** Given two arguments $\mathcal{R}$ and $\mathcal{A}$, let $E = (\Pi_E, \Delta_E, >_E)$ be the expert agent, and $\mathcal{T}$ be the dialectical tree constructed by E such that $\mathsf{root}(\mathcal{T}) = \mathcal{R}$. E's defeater for $\mathcal{A}$ is defined

as expert-defeater$(\mathcal{R}, \mathcal{A}) = (\mathcal{D}, \mathit{Type})$, where:

$\mathcal{D} =$ select-defeater$(\mathfrak{D})$, where $\mathfrak{D}$ is the set of defeaters for $\mathcal{A}$ marked as Ⓤ in $\mathcal{T}$, and

$$\mathit{Type} = \begin{cases} \text{PROPER}, & \text{if } \mathcal{D} \text{ is a proper defeater for } \mathcal{A} \text{ in } \mathcal{T} \\ \text{BLOCKING}, & \text{if } \mathcal{D} \text{ is a blocking defeater for } \mathcal{A} \text{ in } \mathcal{T} \end{cases}$$

Whenever the client receives from the expert a defeater $\mathcal{D}$ for a denier, it will adopt the argument $\mathcal{D}$ using the operator argument-adoption. In addition, it will adjust its preferences accordingly to capture the same type of defeat relationship. The adoption of a defeater is determined by the operator defeater-adoption introduced next:

Definition 13 (Defeater Adoption)**.** Given a client agent $\mathsf{C} = (\Pi_\mathsf{C}, \Delta_\mathsf{C}, >_\mathsf{C})$, two arguments $\mathcal{A}$ and $\mathcal{D}$ such that $\mathcal{D}$ is a defeater for $\mathcal{A}$, and a type of defeater "Type". Let $\mathcal{S}$ be the sub-argument of $\mathcal{A}$ which $\mathcal{D}$ is in conflict with. The defeater adoption of $\mathcal{D}$ as defeater for $\mathcal{A}$ by C is defined as defeater-adoption$(\mathsf{C}, \mathcal{A}, \mathcal{D}, \mathit{Type}) = (\Pi'_\mathsf{C}, \Delta'_\mathsf{C}, >'_\mathsf{C})$, where:

argument-adoption$(\mathsf{C}, \mathcal{D}) = (\Pi'_\mathsf{C}, \Delta'_\mathsf{C}, >_\mathsf{C})$, and

$$>'_\mathsf{C} = \begin{cases} (>_\mathsf{C} \setminus \{(\mathcal{S}, \mathcal{D})\}) \cup \{(\mathcal{D}, \mathcal{S})\}, & \text{if } \mathit{Type} = \text{PROPER} \\ >_\mathsf{C} \setminus \{(\mathcal{D}, \mathcal{S}), (\mathcal{S}, \mathcal{D})\}, & \text{if } \mathit{Type} = \text{BLOCKING} \end{cases}$$

For each denier question $\mathcal{A}$, the corresponding defeater $\mathcal{D}$ adopted by the client will become a new pro argument since it is in favour of the expert's justification. This behaviour is captured by the last transition rule:

$$t_6 : \frac{\begin{array}{c} \mathcal{A} \in \mathbb{D} \ \wedge \ \text{expert-defeater}(I_1, \mathcal{A}) = (\mathcal{D}, \mathit{Type}) \ \wedge \\ \text{defeater-adoption}(\mathsf{C}, \mathcal{A}, \mathcal{D}, \mathit{Type}) = \mathsf{C}' \end{array}}{(\mathsf{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \mathbb{D}) \ \rightarrow \ (\mathsf{C}', [I_1, \ldots, I_n, \mathcal{A}, \mathcal{D}], \mathbb{F} \cup \{\mathcal{D}\}, \emptyset, \mathbb{D} \setminus \{\mathcal{A}\})} \ n \geqslant 1$$

The transition rule t_6 is executed whenever $\mathbb{D}$ has elements, until it is empty (see ϵ in Fig. 2) and the session reaches the state $(\mathsf{C}, \mathbb{I}, \mathbb{F}, \emptyset, \emptyset)$ (back to β in Fig. 2). For each denier question $\mathcal{A}$, the corresponding received defeater $\mathcal{D}$ is added to session state's third component. This means that the next time the session evolves into the state γ (see Fig. 2), if $\mathcal{D}$ is marked as Ⓓ the client will proceed to ask the preference questions for $\mathcal{D}$ (the last received argument).

Example 11. Consider the last session state shown in Example 10 $(\mathsf{C}_5, [\mathcal{B}_1, \ldots, \text{GRT}], \{\mathcal{B}_1\}, \emptyset, \{\mathcal{B}_2, \mathcal{B}_5\})$ in which there are two denier questions for $\mathcal{B}_1$. Recall the expert's dialectical tree depicted in Fig. 7 (left). The answer for the denier question $\mathcal{B}_2$ from the expert will be $(\mathcal{B}_8, \text{PROPER})$, $\mathcal{B}_8$ being the only defeater for $\mathcal{B}_2$ marked as Ⓤ. Then, the transition rule t_6 is executed: the client adopts the defeater $\mathcal{B}_8$, adds $\mathcal{B}_8 > \mathcal{B}_2$ to its preferences, and the session evolves into $(\mathsf{C}_6, [\mathcal{B}_1, \ldots, \text{GRT}, \mathcal{B}_2, \mathcal{B}_8], \{\mathcal{B}_1, \mathcal{B}_8\}, \emptyset, \{\mathcal{B}_5\})$. Note that $\mathcal{B}_8$ is added to the set of arguments in favour of the claim of the justification. Next, the client asks the last denier question $\mathcal{B}_5$ and the expert answers $(\mathcal{B}_9, \text{PROPER})$, being $\mathcal{B}_9$ the only defeater for $\mathcal{B}_5$ marked as Ⓤ. The transition rule t_6 is executed again: the client adopts $\mathcal{B}_9$, adds $\mathcal{B}_9 > \mathcal{B}_5$ to its preferences, $\mathcal{B}_9$ is added to the set of arguments in favour of the claim of the justification, and then the session reaches the

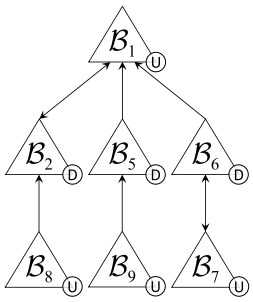

Fig. 8. Dialectical tree from Example 11.

state $(\mathrm{C}_7, [\mathcal{B}_1, \ldots, \mathrm{GRT}, \mathcal{B}_2, \mathcal{B}_8, \mathcal{B}_9, \mathcal{B}_5], \{\mathcal{B}_1, \mathcal{B}_8, \mathcal{B}_9\}, \emptyset, \emptyset)$ (see β in Fig. 2). The client's current dialectical tree for $\mathcal{B}_1$ (the received justification for its original query) is depicted in Fig. 8. Since there are no pending preference or denier questions, the client makes an introspection to check the mark of the justification, which results in Ⓤ. Therefore the transition rule t_2 is executed and the session evolves into the final state $(\mathrm{C}_7, [\mathcal{B}_1, (\mathcal{B}_2, \mathcal{B}_1), \mathrm{UNR}, (\mathcal{B}_3, \mathcal{B}_1), \mathrm{LES}, (\mathcal{B}_4, \mathcal{B}_1), \mathrm{LES}, (\mathcal{B}_5, \mathcal{B}_1), \mathrm{GRT}, \mathcal{B}_2, \mathcal{B}_8, \mathcal{B}_9, \mathcal{B}_5], \emptyset, \emptyset, \emptyset)$. Observe that the second component has the trace of the whole interaction.

The previous example illustrates a scenario in which the justification is marked as Ⓤ after the client adopted the necessary defeaters (new pro arguments) for two deniers. However, as will be shown in Example 12, the client may have undefeated counter-arguments against any of the recently adopted defeaters. These counter-arguments are new deniers, which means that there is still at least one argument marked as Ⓓ in $\mathbb{F}$ (the set of arguments in favour of the claim of the expert's justification). Note that if the client has a denier for an argument received from the expert, that denier is also precluding the justification from being marked as Ⓤ. In this case, the client will proceed to ask the corresponding preference and denier questions for the last received argument in $\mathbb{F}$ marked as Ⓓ. When this argument becomes marked as Ⓤ later in the session, the above pro argument marked as Ⓓ in the same argumentation line will also become marked as Ⓤ (unless it has another denier) and so on, like a cascade effect.

Example 12. Consider again the session between the client Kyle $(\Pi_{\mathrm{K}_1}, \Delta_{\mathrm{K}_1}, >_{\mathrm{K}_1})$ introduced in Example 5 and the expert agent E from Example 4. The session starts with the initial state $(\mathrm{K}_1, [\ ], \emptyset, \emptyset, \emptyset)_a$, by t_1 evolves into $(\mathrm{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \emptyset, \emptyset)_a$, by t_3 evolves into $(\mathrm{K}_2, [\mathcal{A}_1], \{\mathcal{A}_1\}, \{(\mathcal{A}_4, \mathcal{A}_1)\}, \emptyset)_a$, by t_5 evolves into $(\mathrm{K}_2, [\mathcal{A}_1, (\mathcal{A}_4, \mathcal{A}_1), \mathrm{GRT}], \{\mathcal{A}_1\}, \emptyset, \{\mathcal{A}_4\})_a$, and by t_6 evolves into $(\mathrm{K}_3, [\mathcal{A}_1, (\mathcal{A}_4, \mathcal{A}_1), \mathrm{GRT}, \mathcal{A}_4, \mathcal{A}_5], \{\mathcal{A}_1, \mathcal{A}_5\}, \emptyset, \emptyset)_a$. At this point, the client has adopted two arguments: $\mathcal{A}_1$ and the defeater $\mathcal{A}_5$, so its knowledge has changed to $(\Pi_{\mathrm{K}_3}, \Delta_{\mathrm{K}_3}, >_{\mathrm{K}_3})$ where $\Pi_{\mathrm{K}_3} = \{e, x, c, f\}$, $\Delta_{\mathrm{K}_3} = \{\sim a \prec d; d \prec e; \sim p \prec x; a \prec b; b \prec c; \sim d \prec p; p \prec f\}$, and $>_{\mathrm{K}_3} = \{(\mathcal{A}_4, \mathcal{A}_1), (\mathcal{A}_5, \mathcal{A}_4)\}$. Although Kyle has received from E the argument $\mathcal{A}_5$ that defeats the denier $\mathcal{A}_4$, Kyle has the argument $\mathcal{A}_7 = \{\sim p \prec x\}$ that is a blocking defeater for $\mathcal{A}_5$. The argument $\mathcal{A}_7$ is also a denier since it is marked as Ⓤ and is causing $\mathcal{A}_5$ and $\mathcal{A}_1$ (pro arguments) to be marked as Ⓓ. Therefore, the transition rule t_3 is executed again generating a new set of preference questions $\{(\mathcal{A}_7, \mathcal{A}_5)\}$, and the session reaches the state $(\mathrm{K}_3, [\mathcal{A}_1, (\mathcal{A}_4, \mathcal{A}_1), \mathrm{GRT}, \mathcal{A}_4, \mathcal{A}_5], \{\mathcal{A}_1, \mathcal{A}_5\}, \{(\mathcal{A}_7, \mathcal{A}_5)\}, \emptyset)_a$. Then, the transition rule t_4 is executed and, since the expert believes that $\mathcal{A}_5 >_{\mathrm{E}} \mathcal{A}_7$, the client adopts such preference and the session evolves into $(\mathrm{K}_4, [\mathcal{A}_1, (\mathcal{A}_4, \mathcal{A}_1), \mathrm{GRT}, \mathcal{A}_4, \mathcal{A}_5, (\mathcal{A}_7, \mathcal{A}_5), \mathrm{LES}], \{\mathcal{A}_1, \mathcal{A}_5\}, \emptyset, \emptyset)_a$ with $\Pi_{\mathrm{K}_4} = \Pi_{\mathrm{K}_3}$, $\Delta_{\mathrm{K}_4} = \Delta_{\mathrm{K}_3}$, and $>_{\mathrm{K}_4} = \{(\mathcal{A}_4, \mathcal{A}_1), (\mathcal{A}_5, \mathcal{A}_4), (\mathcal{A}_5, \mathcal{A}_7)\}$. Now, since $\mathcal{A}_7$ and $\mathcal{A}_4$ are no longer deniers, introspection$(\mathrm{K}_4, \mathcal{A}_1, \mathcal{A}_1)$ results in Ⓤ and t_2 is applicable, leading to the final state

$(\mathtt{K}_4, [\mathcal{A}_1, (\mathcal{A}_4, \mathcal{A}_1), \text{GRT}, \mathcal{A}_4, \mathcal{A}_5, (\mathcal{A}_7, \mathcal{A}_5), \text{LES}], \emptyset, \emptyset, \emptyset)_a$ in which the literal a is warranted by the agent $\mathtt{K}_4 = (\Pi_{\mathtt{K}_4}, \Delta_{\mathtt{K}_4}, >_{\mathtt{K}_4})$.

Recall from the transition rule t_2 that the arguments in $\mathbb{F}$ are only removed – all together – when the justification becomes marked as Ⓤ. The reason is that arguments marked as Ⓤ may become marked as Ⓓ again. The client constantly acquires new defeasible rules and facts during the session, which could allow it to construct a new argument (denier) which defeats a pro argument previously marked as Ⓤ.

Before concluding this section, a detailed example will be included to show the complete interaction between the client B and the stock market expert M which were introduced in Example 1 to motivate our proposal. In this example we will explain step by step how B's knowledge changes during the client-expert interaction with M regarding the query about whether to buy stocks from the company Acme.

Example 13. Consider the initial knowledge of the client agent $\mathtt{B} = (\Pi_{\mathtt{B}}, \Delta_{\mathtt{B}}, >_{\mathtt{B}})$ where $>_{\mathtt{B}} = \emptyset$ and

$$\Pi_{\mathtt{B}} = \{in\_fusion(acme, steel)\} \qquad \Delta_{\mathtt{B}} = \begin{Bmatrix} \sim buy\_stocks(X) \prec risky\_co(X) \\ risky\_co(X) \prec in\_fusion(X, Y) \end{Bmatrix}$$

Consider again the expert agent $\mathtt{M} = (\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}}, >_{\mathtt{M}})$ introduced in Example 2, which can build the arguments $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$ and $\mathcal{M}_6$ that were shown in Section 2. In addition, consider that the expert's preferences are $\mathcal{M}_5 >_{\mathtt{M}} \mathcal{M}_4$ and $\mathcal{M}_2 >_{\mathtt{M}} \mathcal{M}_6$. The session for the query "$buy\_stocks(acme)$" will start from the initial state $(\mathtt{B}, [\,], \emptyset, \emptyset, \emptyset)_{buy\_stocks(acme)}$.

Figure 9 (left) depicts $\mathcal{T}_{\mathtt{M}}$, the only dialectical tree that the expert can build for the query "$buy\_stocks(acme)$". Since $\mathsf{rootmark}(\mathcal{T}_{\mathtt{M}}) = \text{Ⓤ}$, "$buy\_stocks(acme)$" is warranted by the expert, $\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}}), buy\_stocks(acme)) = \{\mathcal{T}_{\mathtt{M}}\}$, and $\mathsf{root}(\sigma(\mathsf{U\text{-}dtrees}((\Pi_{\mathtt{M}}, \Delta_{\mathtt{M}}), buy\_stocks(acme)))) = \mathcal{M}_1$. Consequently, the expert sends $\mathcal{M}_1$ as justification for the query, and the client B proceeds to adopt the argument $\mathcal{M}_1$. The agent B becomes $\mathsf{argument\text{-}adoption}(\mathtt{B}, \mathcal{M}_1) = \mathtt{B}_2 = (\Pi_{\mathtt{B}_2}, \Delta_{\mathtt{B}_2}, \emptyset)$ where

$$\Pi_{\mathtt{B}_2} = \begin{Bmatrix} in\_fusion(acme, steel) \\ has\_new\_product(acme) \end{Bmatrix} \qquad \Delta_{\mathtt{B}_2} = \begin{Bmatrix} \sim buy\_stocks(X) \prec risky\_co(X) \\ risky\_co(X) \prec in\_fusion(X, Y) \\ buy\_stocks(X) \prec stock\_will\_rise(X) \\ stock\_will\_rise(X) \prec has\_new\_product(X) \end{Bmatrix}$$

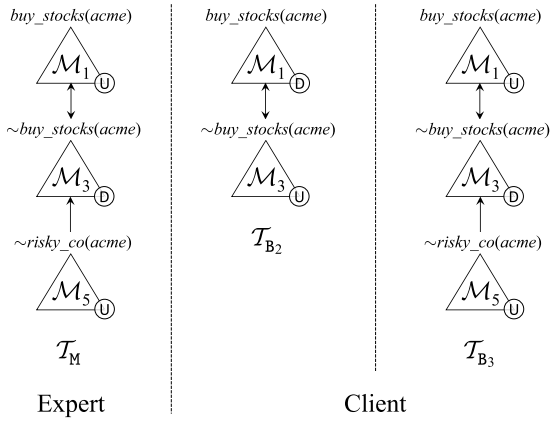

Fig. 9. Dialectical trees from Example 13.

The client ($\mathtt{B}_2$) is currently able to construct the arguments $\mathcal{M}_1$ and $\mathcal{M}_3$. Therefore, by the transition rule t_1, the session evolves into the state $(\mathtt{B}_2, [\mathcal{M}_1], \{\mathcal{M}_1\}, \emptyset, \emptyset)_{buy\_stocks(acme)}$. Next, the client makes an introspection to check the mark of the justification. The client's resulting dialectical tree for $\mathcal{M}_1$ is $\mathcal{T}_{\mathtt{B}_2}$, depicted in Fig. 9 (centre). Given that $\mathcal{M}_3$ is a denier for $\mathcal{M}_1$, $\mathsf{introspection}(\mathtt{B}_2, \mathcal{M}_1, \mathcal{M}_1) =$ Ⓓ. Afterwards, the client selects $\mathcal{M}_1$ from the session state's third component (the last received argument marked as Ⓓ) and generates the corresponding preference questions for that argument: $\mathsf{preference\text{-}questions}(\mathtt{B}_2, \mathcal{M}_1, \mathcal{M}_1) = \{(\mathcal{M}_3, \mathcal{M}_1)\}$. Consequently, by the transition rule t_3, the session evolves into the state: $(\mathtt{B}_2, [\mathcal{M}_1], \{\mathcal{M}_1\}, \{(\mathcal{M}_3, \mathcal{M}_1)\}, \emptyset)_{buy\_stocks(acme)}$.

Then, the client asks the preference question $(\mathcal{M}_3, \mathcal{M}_1)$, and the expert proceeds to answer with $\mathsf{expert\text{-}preference}((\mathcal{M}_3, \mathcal{M}_1)) = \mathtt{UNR}$. Note that $\mathsf{preference\text{-}adoption}(\mathtt{B}_2, (\mathcal{M}_3, \mathcal{M}_1), \mathtt{UNR}) = \mathtt{B}_2$ since the client's set of preferences $>_{\mathtt{B}_2} = \emptyset$ does not change. Afterwards, the client makes an introspection to check the mark of $\mathcal{M}_3$ which results in $\mathsf{introspection}(\mathtt{B}_2, \mathcal{M}_1, \mathcal{M}_3) =$ Ⓤ. Accordingly, $\mathcal{M}_3$ is added to the set of denier questions since the client now needs from the expert a defeater for $\mathcal{M}_3$. Therefore, by t_5, the session evolves into the state $(\mathtt{B}_2, [\mathcal{M}_1, (\mathcal{M}_3, \mathcal{M}_1), \mathtt{UNR}], \{\mathcal{M}_1\}, \emptyset, \{\mathcal{M}_3\})_{buy\_stocks(acme)}$.

Next, the client asks the denier question $\mathcal{M}_3$, and the expert answer with $\mathsf{expert\text{-}defeater}(\mathcal{M}_1, \mathcal{M}_3) = (\mathcal{M}_5, \mathtt{PROPER})$, being $\mathcal{M}_5$ the only defeater for $\mathcal{M}_3$ marked as Ⓤ. The client proceeds to adopt $\mathcal{M}_5$ and becomes $\mathsf{defeater\text{-}adoption}(\mathtt{B}_2, \mathcal{M}_3, \mathcal{M}_5, \mathtt{PROPER}) = \mathtt{B}_3 = (\Pi_{\mathtt{B}_3}, \Delta_{\mathtt{B}_3}, >_{\mathtt{B}_3})$ where

$$\Pi_{\mathtt{B}_3} = \left\{ \begin{array}{l} in\_fusion(acme, steel) \\ has\_new\_product(acme) \\ strong\_co(steel) \end{array} \right\} \qquad \Delta_{\mathtt{B}_3} = \left\{ \begin{array}{l} \sim buy\_stocks(X) \prec risky\_co(X) \\ risky\_co(X) \prec in\_fusion(X, Y) \\ buy\_stocks(X) \prec stock\_will\_rise(X) \\ stock\_will\_rise(X) \prec has\_new\_product(X) \\ \sim risky\_co(X) \prec in\_fusion(X, Y), strong\_co(Y) \end{array} \right\}$$

and $>_{\mathtt{B}_3} = \{\mathcal{M}_5 >_{\mathtt{B}_3} \mathcal{M}_4\}$. Figure 9 (right) depicts $\mathcal{T}_{\mathtt{B}_3}$, the client's resulting dialectical tree for $\mathcal{M}_1$. Note that the client (now $\mathtt{B}_3$) added the preference $\mathcal{M}_5 >_{\mathtt{B}_3} \mathcal{M}_4$, being $\mathcal{M}_4$ the sub-argument of $\mathcal{M}_3$ which $\mathcal{M}_5$ is in conflict with. Then, $\mathcal{M}_5$ is a new argument in favour of the claim of the justification, and it is added to the session state's third component. Thus, by the transition rule t_6, the session evolves into the state $(\mathtt{B}_3, [\mathcal{M}_1, (\mathcal{M}_3, \mathcal{M}_1), \mathtt{UNR}, \mathcal{M}_3, \mathcal{M}_5], \{\mathcal{M}_1, \mathcal{M}_5\}, \emptyset, \emptyset)_{buy\_stocks(acme)}$.

Finally, given that there are no pending preference or denier questions, the client makes an introspection to check the mark of the justification, which results in $\mathsf{introspection}(\mathtt{B}_3, \mathcal{M}_1, \mathcal{M}_1) =$ Ⓤ since $\mathcal{M}_1$ has no deniers. In consequence, by the transition rule t_2, the session evolves into the final state $(\mathtt{B}_3, [\mathcal{M}_1, (\mathcal{M}_3, \mathcal{M}_1), \mathtt{UNR}, \mathcal{M}_3, \mathcal{M}_5], \emptyset, \emptyset, \emptyset)_{buy\_stocks(acme)}$. The client is now able to warrant the literal "*buy_stocks*(*acme*)".

We will end this section showing some results which prove that our strategy behaves as expected. The following lemma shows that from any reachable session state (except a final one) there is always a single applicable transition rule. Recall that all the proofs are included in the Appendix at the end of the paper.

Lemma 1. *Let* $s \neq (\mathtt{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$ *be a reachable state, there exists one and only one applicable transition rule from* s.

Lemma 1 is important because it is used to prove that, given an initial state in which a client agent has made a query to an expert, there always exists a sequence of transitions that leads to the final state. That is, every session will terminate in a finite amount of time and when that occurs the client will have no open issues about the received knowledge. We will also show that, when a session arrives to a final state,

the client has a warrant for the claim of the expert's justification and, therefore, it will have adopted the expert's position about that query.

Theorem 1. *Let s_i be an initial state, there exists a sequence of transitions that leads from s_i to $s_f = (\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$.*

The following corollary is a direct consequence of Theorem 1 and states that the sequence of transitions that leads to the final state is unique. Hence, a session will never loop infinitely.

Corollary 1. *Let s_i be an initial state, the sequence of transitions that lead from s_i to the final session state $s_f = (\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$ is unique.*

Finally, the following result shows that in any final session state the client warrants the claim of the justification sent by the expert.

Theorem 2. *Let $s_f = (\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$ be a final state, the client agent C warrants $\mathsf{claim}(I_1)$.*

From the results given above, we can conclude that, for any session, the goal of our proposal will always be achieved: the expert provides just the necessary information to help the client understand its opinion about the query, and the client agent acquires relevant information regarding the topic of the query to be able to warrant the claim of the received justification.

4. Implementing the expert's selection operators

In this section, we will propose two alternatives for implementing the operators select-dtree and select-defeater. Recall that select-dtree is used by the expert to select a dialectical tree $\mathcal{T}$ from a set of dialectical trees $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ whose roots are all suitable as the justification for the client. Once $\mathcal{T}$ is selected and the justification $\mathcal{J}$ is sent, the client starts asking preference questions and denier questions until it is able to mark $\mathcal{J}$ as Ⓤ. Whenever a denier question $\mathcal{A}$ is asked, select-defeater is used by the expert to select an undefeated defeater $\mathcal{D}$ from a set of defeaters for $\mathcal{A}$ in $\mathcal{T}$.

We will define select-dtree and select-defeater with two different focuses. First, assuming a worst-case scenario regarding to the client's knowledge, we will define the selection operators to minimise the size of the client's *resulting dialectical tree*, that is, the dialectical tree that the client will have to construct until it manages to believe in the claim of the justification. Intuitively, reducing the size of that tree also reduces the session's length in terms of the number of preference questions and denier questions that need to be asked until the session finishes. Then, assuming a more realistic scenario, we will define the selection operators to help the expert reduce the size of the client's resulting dialectical tree by considering the query's context and the previous knowledge the client exposed during the session.

The size of the client's resulting dialectical tree is bounded to which *con* arguments from $\mathcal{T}$ (*i.e.*, arguments against the justification) the client can construct. This follows from the fact that, a denier which – from the expert's perspective – is not a con argument will be removed from the client's dialectical tree immediately after the expert sends LES (recall Definition 10) in response to the corresponding preference question. On the contrary, a denier that actually is a con argument will remain as such – in the best case scenario – until the corresponding denier question is asked and the expert sends a defeater $\mathcal{D}$. Then, if

the client can construct a defeater for $\mathcal{D}$, not only the denier will still remain as such but also the defeater will become a new denier that needs to be dealt with.

In the worst-case scenario, the client will have enough knowledge to be able to construct every con argument in $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ before the session starts. In other words, the client will know every con argument that will be used in each of the dialectical trees that the expert will generate for the query. Nevertheless, even if that occurs, the client will not ask a preference question and a denier question for every con argument. Actually, the client will only do so for those con arguments that will defeat the justification that will be sent by the expert, and for the *pro* arguments (*i.e.*, arguments in favour of the justification) that will be posed by the expert to defeat the deniers from the client. Hence, since the expert has to send only one defeater for each denier question from the client, insightfully selecting them is the key to minimise the size of the client's resulting dialectical tree.

Intuitively, if the expert wants to *minimise* the size of the client's resulting dialectical tree, select-defeater has to always select the defeater (pro argument) that will minimise the number of con arguments – known by the expert – that the client will have to deal with while "exploring" the argumentation lines. Analogously, select-dtree has to select the dialectical tree that – considering select-defeater is optimal – will minimise the number of con arguments that the client will have to deal with. Assuming the worst-case scenario, optimizing the selections is feasible since the expert "knows" all the deniers that the client can pose.

Definition 14 details how to assign a *worst-case scenario selection value* (**WCSSV**) to an argument in a dialectical tree. In the case of a pro argument $\mathcal{A}$, this value represents how many con arguments the client will have to deal with – assuming the worst-case scenario – if the expert selects $\mathcal{A}$ and, from then on, always selects the defeaters with the lowest **WCSSV**.

Definition 14 (Worst-Case Scenario Selection Value)**.** Given a dialectical tree $\mathcal{T}$, the worst-case scenario selection value of an argument $\mathcal{A}$ in $\mathcal{T}$ is defined as:

$$\text{wcs-sv}(\mathcal{A}, \mathcal{T}) = \begin{cases} 0, & \text{if } \mathcal{A} \text{ is a pro argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) = \emptyset \\ 1, & \text{if } \mathcal{A} \text{ is a con argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) = \emptyset \\ 1 + \min(\{\text{wcs-sv}(\mathcal{C}, \mathcal{T}) : \mathcal{C} \in \text{children}(\mathcal{A})\}), & \\ \quad \text{if } \mathcal{A} \text{ is a con argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) \neq \emptyset & \\ \sum_{\mathcal{C} \in \text{children}(\mathcal{A})} \text{wcs-sv}(\mathcal{C}, \mathcal{T}), & \text{if } \mathcal{A} \text{is a pro argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) \neq \emptyset \end{cases}$$

The operator's first and second cases assign the corresponding **WCSSV** to all the dialectical tree's leaves. The operator's third case represents the fact that, whenever the client asks a denier question, the expert can choose the defeater that will cause the lowest number of con arguments becoming new deniers for the client. The operator's fourth case represents the fact that – in the worst-case scenario – the client will be able to construct every con argument below that pro argument. Figure 10 depicts two dialectical trees and their corresponding **WCSSV**s.

Next, we define the operators select-dtree and select-defeater that minimise the number of con arguments that the client will have to deal with assuming the worst-case scenario. Consequently, using these operators imply the size of the client's resulting dialectical tree is also minimised.

Definition 15 (Dialectical Tree Selection using Worst-Case Scenario Criterion)**.** Given a non-empty set of dialectical trees $\mathfrak{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, the selected dialectical tree from $\mathfrak{T}$ is defined as $\text{select-dtree}(\mathfrak{T}) = \mathcal{T}$ where $\mathcal{T} \in \mathfrak{T}$ and there does not exist $\mathcal{T}' \in \mathfrak{T}$ such that $\text{wcs-sv}(\text{root}(\mathcal{T}), \mathcal{T}) > \text{wcs-sv}(\text{root}(\mathcal{T}'), \mathcal{T}')$.

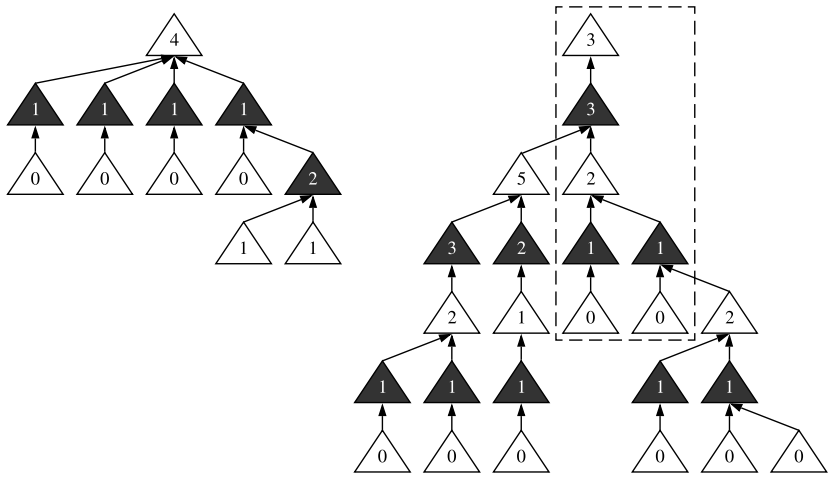

Fig. 10. Two dialectical trees whose arguments contain their **WCSSV**. Undefeated arguments are coloured in white while defeated arguments are coloured in black.

Consider an expert agent that, given a query, constructs the dialectical trees depicted in Fig. 10. In this case, the operator select-dtree selects the one at the right since it has the root with lowest **WCSSV**.

Definition 16 (Defeater Selection using Worst-Case Scenario Criterion)**.** Given a dialectical tree $\mathcal{T}$, and a non-empty set of defeaters $\mathfrak{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ in $\mathcal{T}$, the selected defeater from $\mathfrak{D}$ is defined as select-defeater$(\mathfrak{D}) = \mathcal{D}$ where $\mathcal{D} \in \mathfrak{D}$ and there does not exist $\mathcal{D}' \in \mathfrak{D}$ such that wcs-sv$(\mathcal{D}, \mathcal{T}) >$ wcs-sv$(\mathcal{D}', \mathcal{T})$.

Consider the selected dialectical tree from Fig. 10. The expert will initially send to the client the argument coloured in white with a **WCSSV** of 3 (*i.e.*, [white; 3]) as the justification. Then, if the client can construct the argument [black; 3] and eventually asks the corresponding denier question, the expert has two defeaters to choose from: [white; 5] and [white; 2]. In this case, the operator select-defeater selects [white; 2] – the one with the lowest **WCSSV** – which is sent to the client. Following the same criterion, if the client can construct any of the arguments [black; 1] and eventually asks the corresponding denier question, the expert will select and send the corresponding defeater [white; 0] below. In the worst-case scenario, the client's resulting dialectical tree will be the one inside the dashed box.

Assuming the client has enough knowledge to construct every con argument from the expert may not be realistic depending on the application domain. However, it is clear that unless the client sends in advance all its knowledge, it is impossible for the expert to predict which con arguments the client will be able to build. Hence, given that minimising the size of the client's resulting dialectical tree without making assumptions is not feasible, we will take another approach and define the selection operators to help the expert *reduce* it considering the query's *context* and the *previous knowledge* that the client exposed during the session.

As we mentioned in Section 3, before the session starts the client sends to the expert any specific personal information that represents a context for its query and would affect the expert's answer. As proposed in [22], such contextual information can be temporarily considered by the expert to generate the dialectical trees for the client's query without changing its own knowledge. Once the session has finished, the received context will disappear from the expert's knowledge base and will not be used

for answering queries from other clients. We refer the interested reader to [22] for the details of how different operators for temporarily integrating knowledge can be defined.

Following the aforementioned approach, the query's context will be represented by a DeLP program (Π_P, Δ_P). After the dialectical trees for the query are generated, this program will be referred to as the client's *previous knowledge* and will be reused to store all the facts and defeasible rules *the expert knows the client knows*. Whenever the expert processes a preference question $(\mathcal{A}, \mathcal{B})$ where $\mathcal{A} = \langle \mathsf{R}_a, a \rangle$ and $\mathcal{B} = \langle \mathsf{R}_b, b \rangle$, it will add $\mathsf{evidence}(\mathcal{A}) \cup \mathsf{evidence}(\mathcal{B})$ to Π_P, and $\mathsf{R}_a \cup \mathsf{R}_b$ to Δ_P. In addition, whenever the expert sends a defeater $\mathcal{D} = \langle \mathsf{R}_d, d \rangle$ in response to a denier question, it will add $\mathsf{evidence}(\mathcal{D})$ to Π_P, and R_d to Δ_P. Note that receiving a context, using it to generate the dialectical trees, and keeping the client's previous knowledge updated can be formally introduced into the operational semantics by slightly modifying the transition rule t_1 and Definitions 6, 10 and 12.

Even though it is impossible to predict which con arguments the client will be able to build, the expert can use the client's previous knowledge to select the dialectical tree and the defeaters based on the con arguments that the client is *more likely* to be able to build. In particular, a *constructibility ratio* can be calculated for each con argument, corresponding to the number of elements from that argument (facts and defeasible rules) that are present in the client's previous knowledge over the total number of elements. However, when calculating a constructibility ratio, the expert should not only consider the information the client knows at the moment, but also the information the client will certainly know if it interacts with that particular con argument later in the session. This information consists of all the evidence and defeasible rules of the arguments that are above the con argument in consideration in the corresponding argumentation line, as defined next:

Definition 17 (Ancestral Elements)**.** Given a dialectical tree $\mathcal{T}$ and an argument $\mathcal{A}_i = \langle \mathsf{R}_i, h \rangle$ in $\mathcal{T}$, let $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_i, \ldots, \mathcal{A}_n]$ $(1 \leqslant i \leqslant n)$ be an argumentation line in $\mathcal{T}$. The ancestral elements of $\mathcal{A}_i$ are defined as $\mathsf{ancestral\text{-}elements}(\mathcal{A}_i, \mathcal{T}) = (\mathsf{evidence}(\mathcal{A}_i) \cup \mathsf{R}_i) \cap (\{\bigcup \mathsf{evidence}(\mathcal{A}) \cup \mathsf{R} : \mathcal{A} = \langle \mathsf{R}, h \rangle \in [\mathcal{A}_1, \ldots, \mathcal{A}_{i-1}]\})$.

Then, an argument's constructibility ratio can be calculated by considering both the client's previous knowledge and the argument's ancestral elements, as defined next:

Definition 18 (Constructibility Ratio)**.** Let (Π_P, Δ_P) be the client's previous knowledge. Given a dialectical tree $\mathcal{T}$ and an argument $\mathcal{A} = \langle \mathsf{R}, h \rangle$ in $\mathcal{T}$, the constructibility ratio of $\mathcal{A}$ is defined as:

$$\mathsf{c\text{-}ratio}(\mathcal{A}, \mathcal{T}) = \frac{|(\mathsf{evidence}(\mathcal{A}) \cap \Pi_P) \cup (\mathsf{R} \cap \Delta_P) \cup \mathsf{ancestral\text{-}elements}(\mathcal{A}, \mathcal{T})|}{|\mathsf{evidence}(\mathcal{A}) \cup \mathsf{R}|}$$

Following a similar strategy to the **WCSSV**, the expert can use the constructibility ratio to assign a *constructibility selection value* (**CSV**) to the arguments in a dialectical tree. In the case of a pro argument $\mathcal{A}$, this value represents the sum of the constructibility ratios of the con arguments the client may have to deal with if expert selects $\mathcal{A}$ and, from then on, always selects the defeaters with the lowest **CSV**.

Definition 19 (Constructibility Selection Value)**.** Given a dialectical tree $\mathcal{T}$, the constructibility selection value of an argument $\mathcal{A}$ in $\mathcal{T}$ is defined as:

$$\text{c-sv}(\mathcal{A}, \mathcal{T}) = \begin{cases} 0, & \text{if } \mathcal{A} \text{ is a pro argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) = \emptyset \\ \text{c-ratio}(\mathcal{A}, \mathcal{T}), & \text{if } \mathcal{A} \text{ is a con argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) = \emptyset \\ \text{c-ratio}(\mathcal{A}, \mathcal{T}) + \min(\{\text{c-sv}(\mathcal{C}, \mathcal{T}) : \mathcal{C} \in \text{children}(\mathcal{A})\}), & \\ \quad \text{if } \mathcal{A} \text{ is a con argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) \neq \emptyset & \\ \sum_{\mathcal{C} \in \text{children}(\mathcal{A})} \text{c-sv}(\mathcal{C}, \mathcal{T}), & \text{if } \mathcal{A} \text{ is a pro argument in } \mathcal{T} \text{ and } \text{children}(\mathcal{A}) \neq \emptyset \end{cases}$$

Con arguments increase the total **CSV** by the value corresponding to their constructibility ratio, instead of increasing it by 1 as in the operator wcs-sv. This section concludes with another approach to defining the operators select-dtree and select-dtree in order to help the expert reduce the number of con arguments that the client will have to deal with by just using the client's previous knowledge. Consequently, using these operators imply the number of preference questions and denier questions that need to be asked until the session finishes is also reduced.

Definition 20 (Dialectical Tree Selection using Constructibility Criterion)**.** Given a non-empty set of dialectical trees $\mathfrak{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, the selected dialectical tree from $\mathfrak{T}$ is defined as $\text{select-dtree}(\mathfrak{T}) = \mathcal{T}$ where $\mathcal{T} \in \mathfrak{T}$ and there does not exist $\mathcal{T}' \in \mathfrak{T}$ such that $\text{c-sv}(\text{root}(\mathcal{T}), \mathcal{T}) > \text{c-sv}(\text{root}(\mathcal{T}'), \mathcal{T}')$.

Definition 21 (Defeater Selection using Constructibility Criterion)**.** Given a dialectical tree $\mathcal{T}$, and a non-empty set of defeaters $\mathfrak{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ in $\mathcal{T}$, the selected defeater from $\mathfrak{D}$ is defined as $\text{select-defeater}(\mathfrak{D}) = \mathcal{D}$ where $\mathcal{D} \in \mathfrak{D}$ and there does not exist $\mathcal{D}' \in \mathfrak{D}$ such that $\text{c-sv}(\mathcal{D}, \mathcal{T}) > \text{c-sv}(\mathcal{D}', \mathcal{T})$.

Since Definitions 20 and 21 use the operator c-sv, which relies on the knowledge the expert knows the client knows, there is no guarantee there will be an effective reduction in the client's resulting dialectical tree. Nevertheless, they clearly provide an advantage over using a method based on random selection.

5. Rejecting the expert's opinion

As we have stated in Section 1, we focus on a proposal in which the client's goal is to ask questions to an expert in order to acquire knowledge until it believes in the expert's qualified opinion. As explained, we assume that the client conceives the other agent as an expert on the matter and it will be committed to believe in the answer for its query. Therefore, the client will adopt all the information received from the expert and, in case of a contradiction with its previous knowledge, it will prefer the expert's opinion.

Although our approach was developed for application domains in which committing to the expert's opinion is the best alternative, there can be situations in which it is better not to do so. Clearly, the client is the one that has the responsibility of deciding whether to accept the expert's opinion. For an excellent analysis of this matter see [46], where critical questions are proposed to guide such decision. If the assumption of commitment that we have adopted is relaxed, then the client could argue (with itself or with other agents) about whether to accept the expert's opinion.

Next, we will introduce three additional transition rules which can be included in the operational semantics in order to relax such assumption. With these transition rules, the client can opt to reject an

argument or a preference sent by the expert agent, causing the session to end immediately.

$$t_7 : \frac{\textsf{expert-justification}(q) = \mathcal{J} \ \wedge \ \textsf{reject-expert-opinion}(\mathcal{J})}{(\mathrm{C}, [\,], \emptyset, \emptyset, \emptyset) \ \rightarrow \ (\mathrm{C}, [\mathcal{J}], \emptyset, \emptyset, \emptyset)}$$

With $\textsf{reject-expert-opinion}(\mathcal{J})$ we denote that the client agent rejects the argument $\mathcal{J}$. This may occur, for example, if the client believes in a fact α that is in contradiction with $\mathcal{J}$'s evidence, and refuses to withdraw α from its knowledge base in order to adopt $\mathcal{J}$. This transition rule uses the same current state as the transition rule t_1, allowing the client to not adopt the justification sent when the session begins.

$$t_8 : \frac{(\mathcal{A}, \mathcal{B}) \in \mathbb{P} \ \wedge \ \textsf{expert-preference}((\mathcal{A}, \mathcal{B})) = P \ \wedge \ \textsf{reject-expert-opinion}((\mathcal{A}, \mathcal{B}), P)}{(\mathrm{C}, [I_1, \ldots, I_n], \mathbb{F}, \mathbb{P}, \mathbb{D}) \ \rightarrow \ (\mathrm{C}, [I_1, \ldots, I_n, (\mathcal{A}, \mathcal{B}), P], \emptyset, \emptyset, \emptyset)} \ n \geqslant 1$$

With $\textsf{reject-expert-opinion}((\mathcal{A}, \mathcal{B}), P)$ we denote that the client rejects the preference $P \in \{\textsc{grt}, \textsc{les}, \textsc{unr}\}$ of $\mathcal{B}$ over $\mathcal{A}$. This may occur, for instance, if the client refuses to update its preferences according to what the expert believes. This transition rule uses the same current state as the transition rules t_4 and t_5, allowing the client to not adopt the preference sent by the expert.

$$t_9 : \frac{\mathcal{A} \in \mathbb{D} \ \wedge \ \textsf{expert-defeater}(I_1, \mathcal{A}) = (\mathcal{D}, \mathit{Type}) \ \wedge \ \textsf{reject-expert-opinion}(\mathcal{A}, \mathcal{D}, \mathit{Type})}{(\mathrm{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \mathbb{D}) \ \rightarrow \ (\mathrm{C}, [I_1, \ldots, I_n, \mathcal{A}, \mathcal{D}], \emptyset, \emptyset, \emptyset)} \ n \geqslant 1$$

With $\textsf{reject-expert-opinion}(\mathcal{A}, \mathcal{D}, \mathit{Type})$ we denote that the client rejects the argument $\mathcal{D}$ as a defeater of $\mathit{Type} \in \{\textsc{proper}, \textsc{blocking}\}$ for $\mathcal{A}$. Similar to t_7, this may occur – for instance – if the client believes in a fact α that is in contradiction with $\mathcal{D}$'s evidence, and refuses to withdraw α from its knowledge base in order to adopt $\mathcal{D}$. This transition rule uses the same current state as the transition rules t_6, allowing the client to not adopt the argument $\mathcal{D}$ as a defeater for $\mathcal{A}$.

Note that after any of these transition rules is executed the session evolves into a final state. The reason is that, if the client rejects an expert's argument or preference, it cannot be guaranteed that the client will be able to believe in the claim of the justification any more.

The reader should note that, if any of these transitions rules is added to the system, Theorem 2 will not hold anymore. In addition, t_1, t_4, t_5 and t_6 must be modified as follows:

- $\wedge$ *not*($\textsf{reject-expert-opinion}(\mathcal{J})$) must be added to t_1's condition.
- $\wedge$ *not*($\textsf{reject-expert-opinion}((\mathcal{A}, \mathcal{B}), P)$) must be added to t_4's and t_5's condition.
- $\wedge$ *not*($\textsf{reject-expert-opinion}(\mathcal{A}, \mathcal{D}, \mathit{Type})$) must be added to t_6's condition.

In Section 7 we will discuss on [46] and a possible approach to tackle the implementation of the operator reject-expert-opinion.

6. Discussion

Throughout this paper different design choices were made to tackle the issues that our approach addresses. In this section we will discuss some of those decisions, and for some of them possible alternatives will be commented.

As we explained in the previous sections, the goal of the client agent after adopting the justification $\mathcal{J}$ sent by the expert agent is to believe in such argument, that is, to mark it as Ⓤ. The reader may think that

$$\Pi_{\mathrm{E}} = \begin{Bmatrix} b \\ t \end{Bmatrix} \qquad \Delta_{\mathrm{E}} = \begin{Bmatrix} j \prec a \\ a \prec b, t \\ \sim a \prec t \end{Bmatrix} \qquad \Pi_{\mathrm{C}} = \begin{Bmatrix} t \\ c \end{Bmatrix} \qquad \Delta_{\mathrm{C}} = \begin{Bmatrix} \sim a \prec c \\ j \prec h \\ h \prec b, c \end{Bmatrix}$$

Fig. 11. Client's and expert's knowledge bases from Example 14.

if, instead of doing that, the client aims to find *any* undefeated argument that claims the same as $\mathcal{J}$, the session with the expert would be shorter or the changes in the client's knowledge base would be fewer. Although we could find an example in which that actually happens, this is not necessarily always the case because none of the agents is aware of all the knowledge the other agent has. For instance, consider the following scenario. The expert E sends a justification $\mathcal{J}$ to the client C which, after the adoption, is marked as Ⓓ. By asking just one preference question and one denier question C could mark $\mathcal{J}$ as Ⓤ, but C cannot predict this in advance. Instead, since C has some rules and facts that could potentially construct a new argument $\mathcal{F}$ which claims the same as $\mathcal{J}$, C decides to pose to E these pieces of information in order to find out E's opinion and to receive other pieces of information that could help itself build $\mathcal{F}$. However, after a few interactions, E tells C that one of facts that are required to build $\mathcal{F}$ is invalid, thus making $\mathcal{F}$ an invalid argument and causing the previous interactions to be wasted in some sense.

In addition, consider the expert's justification $\mathcal{J} = \langle \mathsf{R}, j \rangle$. If after adopting $\mathcal{J}$ the client uses only a proper subset of R in order to warrant j instead of warranting $\mathcal{J}$, an undesirable result can arise as shown in the following example:

Example 14. Consider the expert E's and client C's knowledge bases depicted in Fig. 11, and that for E the argument $\mathcal{A} = \langle \{a \prec b, t\}, a \rangle$ is preferred to $\mathcal{B} = \langle \{\sim a \prec t\}, \sim a \rangle$. If C's query is j, then E's justification for j is $\mathcal{J} = \langle \{j \prec a;\ a \prec b, t\},\ j \rangle$. In our approach, in order to warrant $\mathcal{J}$, C would have to deal with the denier $\mathcal{B}$. Instead of that, if C only adds to its knowledge base the fact b that is part of $\mathcal{J}$'s evidence, then C would build the argument $\mathcal{D} = \langle \{j \prec h; h \prec b, c\},\ j \rangle$ which, from its point of view, warrants j since $\mathcal{D}$ has no deniers. However, note that if this alternative is followed, the interaction would end with an undesirable result. Although E's justification contained a sub-argument for a, C will end up ignoring it and believing in $\sim a$. Hence, by proceeding this way, the interaction would finish but the argument $\mathcal{D}$ that C would have for j is something that is not accurate from the expert's point of view. Instead, using our proposal, C would have asked E about the preference between the sub-argument $\mathcal{A}$ and $\mathcal{B}$, and then it would have adopted both this preference and $\mathcal{J}$.

Recall the operator expert-justification introduced in Definition 6. The reader may think that if the expert sends to the client all the justifications it has (instead of selecting one) then the interaction could be shorter. Nevertheless, as shown in the example below, with this alternative both the interaction and the changes to the client's knowledge could increase. The reason is that, as shown in the following example, either the client should explore for each justification the associated dialectical tree, or has to make all the necessary changes to accept them all.

Example 15. Consider a client agent C that, instead of just one, receives two justifications $\mathcal{J}_1$ and $\mathcal{J}_2$ from the expert agent E. Suppose that C has five deniers for $\mathcal{J}_1$ (see Fig. 12(a)) while it has only one denier for $\mathcal{J}_2$ (see Fig. 12(b)) and thus starting the session with $\mathcal{J}_2$ seems to be the option that would imply the shortest interaction with the expert. Then, C asks the preference question ($\mathcal{D}$, $\mathcal{J}_2$) and E replies with GRT since it prefers $\mathcal{D}$ over $\mathcal{J}_2$. Next, since the denier $\mathcal{D}$ is still undefeated, C asks E for a defeater for $\mathcal{D}$ and receives $\mathcal{F}$. After adopting $\mathcal{F}$, C realizes that it has thirty deniers for it (see Fig. 12(c)) and

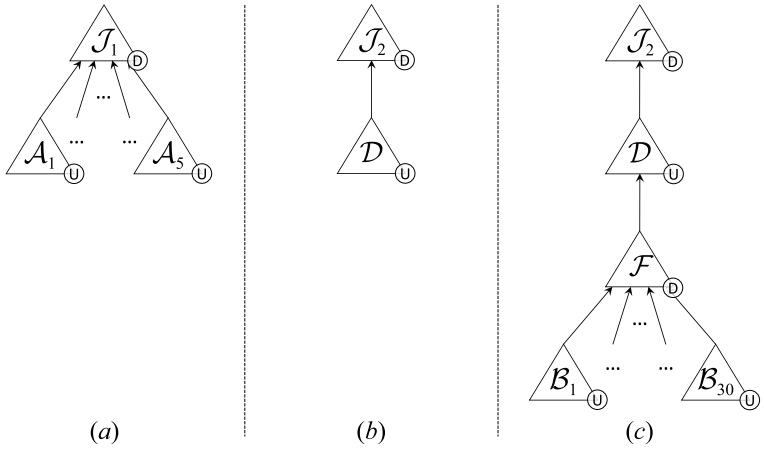

Fig. 12. Dialectical trees from Example 15.

the interaction with the expert will have at least thirty additional preference questions. Note that, if C had started the session with $\mathcal{J}_1$ and the expert had replied with LES to the five preference questions $(\mathcal{A}_1, \mathcal{J}_1), \ldots, (\mathcal{A}_5, \mathcal{J}_1)$, then the interaction would have been shorter.

When Definition 9 was introduced, we mentioned that we opted to use preference questions in order to abstract away from the details of how the client's criterion would be modified. Although we could have formalized our approach with a particular family of preference criteria similarly to [7,8], we aimed for a high level formalization that allows to be instantiated with any argument comparison criterion from any family. As we will show next, any comparison criterion can be applied to our approach by making some changes in the operators expert-preference and preference-adoption. For instance, consider the argument comparison criterion referred to as *rule's priorities* [21,22]. In this criterion an argument $\mathcal{A}$ is preferred to another argument $\mathcal{B}$ if there exists at least one rule r_a in $\mathcal{A}$ and one rule r_b in $\mathcal{B}$ such that $r_a > r_b$ and there is no r'_b in $\mathcal{B}$ and r'_a in $\mathcal{A}$ such that $r'_b > r'_a$, where ">" is a partial order among defeasible rules that is explicitly provided with the program. The intuitive meaning of $r_a > r_b$ is that the rule r_a is preferred over r_b in the application domain being considered. Hence, since the argument comparison criterion is based on the preferences among defeasible rules, both the expert agent E and the client agent C would have the sets $>_{\mathrm{E}}$ and $>_{\mathrm{C}}$ implemented as a partial order among their defeasible rules (Δ_{E} and Δ_{C}, respectively). Given a preference question $(\mathcal{A}, \mathcal{B})$, in addition to the corresponding preference (GRT, LES or UNR) the operator expert-preference would send two sets: an *add-set* and a *delete-set*. The add-set would contain all the pairs of rules (r_1, r_2) such that r_1 is in $\mathcal{A}$ and r_2 is in $\mathcal{B}$ (or vice versa) and (r_1, r_2) is in $>_{\mathrm{E}}$. Furthermore, if E considers that $\mathcal{A}$ is not a valid argument because of any or some of the following reasons, these sets would contain additional elements: if $\mathcal{A}$ uses a defeasible rule r_{inv} that is not in Δ_{E}, the add-set would also contain all the pairs (r_b, r_{inv}) for every rule r_b in $\mathcal{B}$; if $\mathcal{A}$ is non-minimal, the add-set would also contain all the facts from Π_{E} that cause $\mathcal{A}$ to be non-minimal; if $\mathcal{A}$ uses evidence that is not in Π_{E}, the delete-set would contain all that evidence; finally, if $\mathcal{A}$ uses evidence that is contradictory to Π_{E}, the delete-set would also contain all that contradictory evidence. On the other hand, the operator preference-adoption would add to $>_{\mathrm{C}}$ and Π_{C} all the elements in the add-set and would delete from $>_{\mathrm{C}}$ and Π_{C} all the elements in the delete-set. For instance, if C asks the preference question $(\langle\{a \prec b\}, a\rangle, \langle\{\sim a \prec c\}, \sim a\rangle)$, E will answer with GRT, *add-set* $= \{(a \prec b, \sim a \prec c)\}$ and *delete-set* $= \emptyset$, and then C will add $(a \prec b, \sim a \prec c)$ to $>_{\mathrm{C}}$.

7. Related work

In [47], the authors define *dialogue* as a normative framework comprising exchange of arguments between two participants reasoning together to achieve a collective goal. During a dialogue, the participants take turns to make "moves" and have individual goals and strategies in making such moves. A *move* is a sequence of locutions provided by a participant at a particular point in the sequence of dialogue. In addition, the authors define four kinds of rules which characterise different types of dialogue: *locution rules* that define the permissible locutions – like statements, questions, inferences, and so on; *structural rules* that define the order in which moves can be made by each participant; *commitment rules* that define the insertion and deletion of propositions from a participant's commitment store – as a consequence of its moves; and *win-loss rules* that determine the conditions under which a player wins or loses the game.

Regarding to the different types of dialogues, the authors particularly define *information-seeking dialogue* in which one participant has some knowledge and the other party lacks and needs that information. This type of dialogue's goal is to share that knowledge. *Expert consultation* is a subtype of the information-seeking dialogue in which one participant is an expert in a particular domain or field of knowledge, and the other is not. By asking questions, the non-expert participant (in [47], the *layman*) tries to elicit the expert's opinion (advice) on a matter which the questioner itself lacks direct knowledge. In this kind of dialogue, the questioner can arrive at a presumptive conclusion which gives a plausible expert-based answer to its question. Our proposal clearly fits the information-seeking and expert consultation concepts. Furthermore, we could instance every element in the definition of dialogue with elements of our strategy to make a match. For instance, there are strong similarities between the "locution rules" and the operators used by the agents (e.g. expert-defeater and defeater-adoption), between the "structural rules" and the preconditions of the transition rules, between the "commitment rules" and the configurations of the current and resulting states of the transition rules, and between the "win-loss rules" and the final session state $(\mathtt{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)_q$. However, instead of defining a dialogue framework in which some rules and strategies are left to be specified by the agent designer, in this work we focus on defining a specific strategy that guarantees that both agent's goals are always achieved. That is, when the session finishes, the client agent will believe in the claim of the expert's justification.

In [31], the authors present a framework for argumentation-based dialogues between agents. They define a set of locutions by which agents can trade arguments, a set of agent attitudes which relate what arguments an agent can build and what locutions it can make, and a set of protocols by which dialogues can be carried out. Regarding to the *assertion* attitudes, an agent may be either *confident*, if it can assert any proposition p for which it can construct an argument (S, p); or *thoughtful*, if it can assert any proposition p for which it can construct an acceptable argument (S, p). Regarding to the *acceptance* attitudes, an agent may be either *credulous*, if it can accept any proposition p if p is backed by an argument; *cautious*, if it can accept any proposition p if it is unable to construct a stronger argument for $\neg p$; or *skeptical*, if it can accept any proposition p if there is an acceptable argument for p. Although in our approach the agents deal with arguments instead of propositions, we can informally categorise them in their proposed terminology. The expert agent acts with a thoughtful attitude because it only answers a query if it can construct an undefeated argument for either p or $\neg p$. We consider that an expert with a confident attitude – that is able to send arguments without considering their marks (Ⓤ or Ⓓ) – would not behave as an expert according to [47]. On the other hand, the client agent follows a skeptical attitude because only accepts a conclusion p when it has an accepted (undefeated) argument for p.

Based on the typology of [47], the authors of [31] also define information-seeking dialogues in which one agent A seeks the answer to some question from another agent B, which is believed by the first

to know the answer. In the protocol which they provide for an information-seeking dialogue about a proposition p, first A asks a *question*(p) and B replies with either *assert*(p), *assert*$(\neg p)$, or *assert*$(\mathcal{U})$, depending upon the contents of its knowledge base and its assertion attitude. $\mathcal{U}$ indicates that, for whatever reason B cannot give an answer and – like in our approach – the dialogue terminates without the question being resolved. Excluding this case, A either *accepts* B's response if A's acceptance attitude allows it, or *challenges* it to make B explicitly state the argument supporting that proposition. Then, B replies to that challenge with an *assert*(S), where S is the support of an argument for the challenged proposition, and this process is repeated for every proposition in S. In contrast, in our approach it is not optional to receive an argument backing a proposition. Given that our agent's inference mechanism is argumentative, the client agent will require a justification. Finally, unlike our proposal, the authors state that if the agent which starts the information-seeking dialogue by making a question is either skeptical or cautious, it may never accept the proposition whatever the other agent says.

In [17–19], the authors model information-seeking dialogues using Assumption-Based-Argumentation (ABA) [16]. In such proposal, a *questioner* agent α proposes a topic χ and an *answerer* agent β utters information of relevance to χ. They assume that the questioner contributes no information apart from initiating the dialogue and that the answerer is interested in conveying information for χ, but not against. Strategy-move functions are used to help the agents identify suitable utterances that advance the information-seeking dialogue towards its goal while fulfilling the participants' aims. The authors also define two subtypes of dialogues: *IS-Type I*, in which the answerer agent conveys all arguments for χ, and *IS-Type II*, in which the answerer agent conveys only one argument for χ. Unlike that approach, in our work we consider that the client agent may have previous knowledge in conflict with the information acquired from the expert agent. Given that the client agent is committed to believe in the justification for the query sent by the expert, the client will need to adapt its previous knowledge losing as little information as possible. This may either imply the removal of facts contradicting an unchallengeable information acquired from the expert, the addition of new knowledge that allows the construction of arguments provided by the expert, or even further interactions with the expert.

A framework for representing inquiry dialogues that uses Defeasible Logic Programming as the underlying representation formalism is presented in [7,8] and its implementation is reported in [35]. Their approach, unlike us, does not deal with information-seeking dialogues or expert consultations. Instead, their focus is on inquiry dialogues [47], in which the initial situation, the main goal, the participant's aims, and the side benefits are different from the ones in information-seeking. Inquiry dialogues are defined as arising from an initial situation of "general ignorance" and as having the main goal to achieve the "growth of knowledge and agreement", while each individual participating aims to "find a 'proof' or destroy one". As explained in [47], in inquiry dialogues there is a common problem to be solved between the participants while in information-seeking dialogues there in not. In the later, the knowledge is already there, and the problem is to communicate it from one party to the other.

The authors define and give the necessary details to generate two subtypes of inquiry dialogues. *Argument inquiry dialogues* allow two agents to share beliefs to jointly construct arguments for a specific claim that neither of them may construct from their own personal beliefs alone. For instance, an agent that wants to construct an argument for ϕ can open an argument inquiry dialogue with the defeasible rule $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \phi$ as its topic. If the two participants manage to provide arguments for each of the elements α_i $(1 \geqslant i \geqslant n)$, then it would be possible for an argument for ϕ to be constructed. In our approach, after constantly acquiring new arguments during the session, the questioner agent may also be able to build new different arguments that neither of them may construct from their own personal beliefs alone. However, asking for sub-arguments for the antecedents of a defeasible rule is unnecessary: the

questioner knows that if it needs to find a defeater for a certain argument then the expert can certainly provide a complete argument that serves such purpose. On the other hand, *warrant inquiry dialogues* allow two agents that are interested in determining the acceptability of a particular argument to share arguments in order to jointly construct a dialectical tree that none of them may construct from their own personal beliefs alone. In our proposal, both agents do not build a dialectical tree together *per se* but the expert provides just enough arguments to help the questioner mark the justification argument as undefeated in its corresponding dialectical tree. This implies that each agent's dialectical tree for the justification argument could result different by the time the session finishes. Argument inquiry dialogues are often embedded within warrant inquiry dialogues. Without embedded argument inquiry dialogues, the arguments that can be exchanged within a warrant inquiry dialogue potentially miss out on useful arguments that involve unexpressed beliefs of the other agent.

The main contribution of [7,8] is a protocol and strategy sufficient to generate sound and complete warrant inquiry dialogues. To prove this, the authors compare the outcome of their dialogues with the outcome that would be arrived at by a single agent that has as its beliefs the union of both the agents participating in the dialogue's beliefs. This is, in a sense, the "ideal" situation in which there are clearly no constraints on the sharing of beliefs. However, this union of both agents' beliefs is only possible because it is assumed that the agents use *defeasible facts* instead of strict facts. Therefore, in contrast with our approach, no contradictions can arise from both agents' joint beliefs. In addition, the authors assume a global preference ordering across all knowledge, from which a global preference ordering across arguments is derived. Hence, an exchange of opinion on the participants' arguments' preferences is unnecessary, differently from our proposal.

The author of [46] explains *appeal to expert opinion*, a form of argument based on the assumption that the source is alleged to be in a position to know about a subject because he or she has expert knowledge of that subject. In addition, the author mentions that there is a natural tendency to respect experts and treat them as infallible, which is a dangerous approach since they can be wrong. For this reason, an informal argumentation scheme for deciding whether to accept an expert's opinion is introduced. The author mentions that it is vital to see appeal to expert opinion as defeasible, as open to the following critical questions: (1) How credible is E as an expert source? (2) Is E an expert in the field that A (the proposition) is in? (3) What did E assert that implies A? (4) Is E personally reliable as a source? (5) Is A consistent with what other experts assert? (6) Is E's assertion based on evidence? Regarding question 5, the *consistency question*, the author mentions that one can compare A with other known evidence, particularly with what other experts on the field say.

The consistency question from [46] together with the proposal of [25,39,40] could be used in our formalism to add an implementation for the operator reject-expert-opinion defined in Section 5. The authors consider that agents can obtain information from multiple informants, and that the attribution of trust to a particular informant can be higher than the trust attributed to others. Each agent has its own partial order among the different informants, representing the credibility it assigns to them. Then, when information obtained from different informants is in conflict, trust is used in the decision process leading to a prevailing conclusion. In our proposal, whenever the expert sends to the client an argument containing evidence in contradiction to what the client believes, the operator reject-expert-opinion could compare the trust assigned to the different informants to decide which fact prevails. Then, if any of the facts in the expert's argument's evidence does not prevail, the reject-expert-opinion returns TRUE, causing the argument to be rejected. The same comparison can be done whenever the client needs to modify its preferences between arguments.

We will end this section with a comment on the differences between our proposal and borth belief revision and revision of argument frameworks (AFs). Recall that, whenever the client agent receives an argument from the expert agent (either the justification or a defeater for a con argument) the operator argument-adoption (see Definition 7) removes any facts from the client's knowledge base that are in contradiction with the argument's evidence, in a prioritized belief revision fashion. In addition, differently from traditional belief revision, the operator adds all the argument's defeasible rules to the client's knowledge base since DeLP allows the defeasible derivation of contradictory literals. This allows the client not only to construct the received argument, but also to combine its own defeasible rules with the just acquired knowledge to construct new arguments. Given that the client may have deniers for the justification and its goal is to mark the justification as Ⓤ, belief revision techniques are not enough. After adopting the argument, the client may need to change its preferences between arguments to be aligned with the expert's opinion on the matter.

Since changing preferences alters how arguments defeat each other, our proposal may resemble existing work on revision of argumentation frameworks [11,13,14,30,38]. These approaches, regardless of their individual goals, end up adding or removing arguments or attacks and returning a new AF or set of AFs as output. Especially, [11,13] revise AFs by modifying the sets of extensions and then modifying the attack graph accordingly to the newly obtained extensions, while [14] focuses on updating the extensions. However, our proposal differs both conceptually and methodologically: we do not want the client agent to mark the expert's justification as Ⓤ in a single step by changing its preferences or previous knowledge without further information. Instead, we want the client to maintain the communication with the expert in order to ask questions to keep acquiring relevant knowledge (arguments and preferences) and make informed changes considering a qualified opinion.

8. Conclusions and future work

In this work, we have presented a strategy that involves two agents: an expert agent which has expertise in a particular domain or field of knowledge and a client agent which lacks that quality. The inexpert client or questioner initially makes a query to the expert in order to acquire knowledge about a topic in which the client itself lacks expertise. The client agent is committed to believe in the answer of the query and, unlike other approaches, we consider that the client may have previous knowledge in conflict with the information acquired from the expert agent. Thus, the client could need to adapt its previous knowledge without making unnecessary changes in order to accept what the expert says.

A naive solution to the proposed problem would be for the expert to send all of its knowledge to the client, but this is not feasible because, depending on the domain, the expert may have private information or its knowledge could be very extensive. Furthermore, the joined knowledge bases of both agents would probably have contradictions which are completely unnecessary to solve because their relevance is outside of the domain or field of knowledge of the query. For instance, consider a simple scenario in which the expert believes in $\sim exist(aliens)$, and the client believes in $exist(aliens)$ and asks the query $buy\_stocks(acme)$. If we join both knowledge bases this contradiction must be treated, but unnecessarily removing facts from the client's knowledge base that are irrelevant with respect to the client's query just to solve this conflict would not be appropriate. Another naive solution would be for the client to imitate the expert's knowledge about the query by simply adding an unchallengeable fact to its knowledge base, but the client would blindly believe in the answer that was sent. On the contrary, with the strategy we

propose in this work, the client can arrive at a presumptive conclusion which gives a plausible, expert-based answer to its initial query. From a high-level point of view, the client agent learns to think about the topic in question like the expert agent.

All interactions between both agents occur during a session which will start only if the expert has a warrant for the literal of the query or its complement, i.e., it must be certain about the topic of the client agent's query. In this case, the expert will select one of its undefeated arguments to send to the client as justification. Whenever the client acquires new knowledge from the expert, it may have to withdraw some of its previous contradictory knowledge (which is inaccurate from the expert's position).

The goal of this strategy is for the client to be able to mark the justification as Ⓤ (that is, it manages to believe in the claim of the justification) without making unnecessary changes to its previous knowledge. However, after adopting the justification, the client may have denier arguments for it, some of which may be acknowledged or not by the expert. In this case, the client will have to ask preference and denier questions to the expert from which it will acquire new preferences and arguments. The session will continue until the goal is finally achieved. We proved that every session eventually finishes and, when this happens, the client agent will believe in the claim of the expert's justification. This means that the goal of this strategy is always achieved. Another conclusion we can draw is that the fewer pieces of information (facts and rules) the client previously has about the topic of the query, the shorter will be the session. This property holds because the client will have fewer denier arguments against the justification, and it will be easier to mark it as Ⓤ. Ultimately, it could occur that the client knows absolutely nothing, and the goal of the session would be simply achieved by adopting the justification from the expert.

DeLP was used as the underlying knowledge representation and reasoning mechanism in order to show how to solve the problems associated to the agents' argument structures in an information-seeking setting. Conceptually, our proposal can be divided into three different levels: First, a level corresponding to the outline illustrated in Fig. 2, which abstracts away from the argumentative reasoning mechanism used by the agents. Second, a level corresponding to the actual transition rules whose conditions are defined by different operators. Third, a level corresponding to the implementation of those operators. In particular, the transition rules and the operators were defined considering the particular characteristics of DeLP. Even though the separation of these three conceptual levels provides some modularity to our approach, the second level would need some modifications in order to be able to adapt the formalism to another structured argumentation formalism (*e.g.*, ASPIC+ [29]). The transition rules should be more abstractly defined to allow different operators – regardless of how they are implemented – as long as they satisfy certain conditions. The modularization of the transition system is left as future work.

Future work has multiple directions. Although DeLP allows the use of strict rules (besides defeasible rules), we have not considered them in our proposal. We plan to adapt the operational semantics of our formalism to allow agents to have strict rules in their knowledges bases. Hence, in order to guarantee that the client will always be able to believe in the expert's justification, different belief revision operators will be necessary to insightfully adapt the client's knowledge during the session. In addition, we are interested in adapting our strategy to the *didactic dialogues* proposed in [47], in which the purpose of the expert agent is not just to satisfy the doubts of the client agent, but also to turn it into an expert itself. To provide a strategy for didactic dialogues, our expert agent would deliberatively need to decide how to share all its knowledge about its field of expertise to the client. This could also imply an effort from the client to show the expert how deep is its knowledge regarding the topic at hand. Finally, we would like to further analyze the selection operators defined in Section 4 to formally define minimal information exchange in the context of a session between a client and an expert.

Acknowledgements

This work has been partially supported by PGI-UNS (grants 24/ZN32, 24/N046).

Appendix

Proposition 1. *Let* $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$ *be the expert agent, and* $\mathcal{A}$ *an argument constructed by* $\mathtt{E}$*, if* $\mathsf{argument\text{-}adoption}(\mathtt{C}_1, \mathcal{A}) = \mathtt{C}_2$ *then* $\mathcal{A}$ *can be constructed by* $\mathtt{C}_2$.

Proof. Let $\mathtt{C}_1 = (\Pi_{\mathtt{C}_1}, \Delta_{\mathtt{C}_1}, >_{\mathtt{C}_1})$ and $\mathtt{C}_2 = (\Pi_{\mathtt{C}_2}, \Delta_{\mathtt{C}_2}, >_{\mathtt{C}_2})$. In order to prove that $\mathcal{A} = \langle \mathsf{R}, h \rangle$ can be constructed by $\mathtt{C}_2$, by Definition 1, the following conditions must hold:

(1) $\mathsf{R} \subseteq \Delta_{\mathtt{C}_2}$, and
(2) there exists a defeasible derivation for h from $\Pi_{\mathtt{C}_2} \cup \mathsf{R}$, and
(3) the set $\Pi_{\mathtt{C}_2} \cup \mathsf{R}$ is non-contradictory, and
(4) $\mathcal{A}$ is minimal: there is no proper subset R' of R such that R' satisfies conditions (2) and (3).

Next, we prove that all the aforementioned conditions hold:

(1) By Definition 7, every defeasible rule $d \in \mathsf{R}$ was added to $\Delta_{\mathtt{C}_2}$, then it holds $\mathsf{R} \subseteq \Delta_{\mathtt{C}_2}$.
(2) By hypothesis there is a defeasible derivation for h from $\Pi_{\mathtt{E}} \cup \mathsf{R}$. Also, by Definition 7, every literal $e \in \mathsf{evidence}(\mathcal{A})$ (which activates $\mathcal{A}$) was added to $\Pi_{\mathtt{C}_2}$ and every defeasible rule $d \in \mathsf{R}$ was added to $\Delta_{\mathtt{C}_2}$. Then there must exist a defeasible derivation for h from $\Pi_{\mathtt{C}_2} \cup \mathsf{R}$.
(3) By Definition 7, every literal $\overline{r} \in \Pi_{\mathtt{C}_1}$ such that $r \prec b_1, \ldots, b_n \in \mathsf{R}$ and every literal $\overline{e} \in \Pi_{\mathtt{C}_1}$ such that $e \in \mathsf{evidence}(\mathcal{A})$ (which would have made $\Pi_{\mathtt{C}_2}$ contradictory) were removed from $\Pi_{\mathtt{C}_1}$ after the argument adoption. Also, it holds that $\Pi_{\mathtt{C}_1}$ was non contradictory. Then, $\Pi_{\mathtt{C}_2} \cup \mathsf{R}$ cannot be contradictory.
(4) Given that $\mathcal{A}$ is a minimal argument for agent $\mathtt{E}$, the only possible way that $\mathcal{A}$ is not minimal for agent $\mathtt{C}_2$ is that there exists a literal $l \in \Pi_{\mathtt{C}_2}$ such that $l \notin \Pi_{\mathtt{E}}$ and $l \prec b_1, \ldots, b_n \in \mathsf{R}$ and $l \prec b_1, \ldots, b_n \notin \mathsf{R}'$ and $\mathsf{R}' \subset \mathsf{R}$; that is, $\mathtt{C}_2$ has a literal (which $\mathtt{E}$ does not have) that allows it to construct another argument which is a proper subset of R. However, this is not possible because, according to Definition 7, every literal l such that $l \prec b_1, \ldots, b_n \in \mathsf{R}$ was removed from $\Pi_{\mathtt{C}_1}$ after the argument adoption. □

Proposition 2. *Let* $\mathcal{A}$ *be an argument constructed by the expert agent* $\mathtt{E}$*, and* $\mathcal{B}$ *an argument constructed by both a client agent* $\mathtt{C}_1$ *and* $\mathtt{E}$*, if* $\mathsf{argument\text{-}adoption}(\mathtt{C}_1, \mathcal{A}) = \mathtt{C}_2$ *then* $\mathcal{B}$ *can be constructed by* $\mathtt{C}_2$.

Proof. Let $\mathtt{C}_2 = (\Pi_{\mathtt{C}_2}, \Delta_{\mathtt{C}_2}, >_{\mathtt{C}_2})$ and $\mathtt{E} = (\Pi_{\mathtt{E}}, \Delta_{\mathtt{E}}, >_{\mathtt{E}})$. Let us suppose that after adopting $\mathcal{A} = \langle \mathsf{A}, a \rangle$, $\mathtt{C}_2$ cannot construct $\mathcal{B} = \langle \mathsf{B}, b \rangle$ anymore. Then, either:

(1) There is no defeasible derivation for b from $\Pi_{\mathtt{C}_2} \cup \mathsf{B}$, which requires that either:
   (a) a defeasible rule $d \in \mathsf{B}$ is not in $\Delta_{\mathtt{C}_2}$, or
   (b) a literal $e \in \mathsf{evidence}(\mathcal{B})$ is not in $\Pi_{\mathtt{C}_2}$, which, according to Definition 7, implies that either:
      i. $\overline{e} \in \mathsf{evidence}(\mathcal{A})$, or
      ii. $\overline{e} \prec b_1, \ldots, b_n \in \mathsf{A}$, or
      iii. $e \prec b_1, \ldots, b_n \in \mathsf{A}$; or

(2) the set $\Pi_{\mathtt{C}_2} \cup \mathsf{B}$ is contradictory, which requires that for some $\overline{e} \prec b_1, \ldots, b_n \in \mathsf{B}$ the literal $e \in \mathsf{evidence}(\mathcal{A})$ was added to $\Pi_{\mathtt{C}_2}$, or
(3) $\mathcal{B}$ is not minimal any more, which requires that for some $e \prec b_1, \ldots, b_n \in \mathsf{B}$ the literal $e \in \mathsf{evidence}(\mathcal{A})$ was added to $\Pi_{\mathtt{C}_2}$.

Condition (a) is not possible because defeasible rules are never removed during an argument adoption. Condition (i) is not possible because, given that $\mathsf{evidence}(\mathcal{B}) \cup \mathsf{evidence}(\mathcal{A}) \subseteq \Pi_{\mathtt{E}}$, $\Pi_{\mathtt{E}}$ would be contradictory. Condition (ii) is not possible because, given that $e \in \Pi_{\mathtt{E}}$, $\mathsf{A} \cup \Pi_{\mathtt{E}}$ would be contradictory, and thus $\mathcal{A}$ would not be constructed by $\mathtt{E}$. Condition (iii) is not possible because, given that $e \in \Pi_{\mathtt{E}}$, $\mathcal{A}$ would not be minimal for $\mathtt{E}$, and thus $\mathcal{A}$ would not be constructed by $\mathtt{E}$. Condition (2) is not possible because, given that $e \in \Pi_{\mathtt{E}}$, $\mathsf{B} \cup \Pi_{\mathtt{E}}$ would be contradictory, and thus $\mathcal{B}$ would not be constructed by $\mathtt{E}$. Finally, condition (3) is not possible because, given that $e \in \Pi_{\mathtt{E}}$, $\mathcal{B}$ would not be minimal for $\mathtt{E}$, and thus $\mathcal{B}$ would not be constructed by $\mathtt{E}$. Contradiction. □

Proposition 3. *Let* $\mathtt{C}$ *be a client agent,* $\mathcal{A}$ *an argument in a dialectical tree* $\mathcal{T}$*, and* $\mathcal{R} = \mathsf{root}(\mathcal{T})$*, if* $\mathsf{introspection}(\mathtt{C}, \mathcal{R}, \mathcal{A}) = \textcircled{D}$ *then* $\mathsf{preference\text{-}questions}(\mathtt{C}, \mathcal{R}, \mathcal{A}) \neq \emptyset$.

Proof. Let us suppose that $\mathsf{preference\text{-}questions}(\mathtt{C}, \mathcal{R}, \mathcal{A}) = \emptyset$. Then, according to Definition 9, there is no argument $\mathcal{B}$ such that $\mathcal{B}$ is a defeater for $\mathcal{A}$ and $\mathsf{mark}(\mathcal{B}, \mathcal{T}) = \textcircled{U}$, where $\mathsf{root}(\mathcal{T}) = \mathcal{R}$. Therefore, $\mathsf{mark}(\mathcal{A}, \mathcal{T}) = \textcircled{U}$ and, according to Definition 8, $\mathsf{introspection}(\mathtt{C}, \mathcal{R}, \mathcal{A}) = \textcircled{U}$. Contradiction. □

Lemma 1. *Let* $s \neq (\mathtt{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$ *be a reachable state, there exists one and only one applicable transition rule from* s.

Proof. For this proof – starting from the initial state – we will analyse the conditions under which the transition rules are applicable and lead to all the possible session states.

- From the initial state $(\mathtt{C}_0, [\,], \emptyset, \emptyset, \emptyset)$ the transition rule t_1 is always applicable and leads to the state $(\mathtt{C}_1, [\mathcal{J}], \{\mathcal{J}\}, \emptyset, \emptyset)$.
- From the state $(\mathtt{C}_n, [I_1, \ldots, I_n], \mathbb{F}_n, \emptyset, \emptyset)$ with $\mathbb{F}_n \neq \emptyset$ (for example, $(\mathtt{C}_1, [\mathcal{J}], \{\mathcal{J}\}, \emptyset, \emptyset)$), either $\mathsf{introspection}(\mathtt{C}_n, I_1, I_1) = \textcircled{U}$ or $\mathsf{introspection}(\mathtt{C}_n, I_1, I_1) = \textcircled{D}$. If $\mathsf{introspection}(\mathtt{C}_n, I_1, I_1) = \textcircled{U}$ then the only applicable transition rule is t_2 and leads to the final state $(\mathtt{C}_n, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$. On the contrary, if $\mathsf{introspection}(\mathtt{C}_n, I_1, I_1) = \textcircled{D}$, there must exist at least one argument $\mathcal{A} \in \mathbb{F}_n$ such that $\mathsf{introspection}(\mathtt{C}_n, I_1, \mathcal{A}) = \textcircled{D}$ because $I_1 \in \mathbb{F}_n$. In this case, the only applicable transition rule is t_3 and leads to the state $(\mathtt{C}_n, [I_1, \ldots, I_n], \mathbb{F}_n, \mathbb{P}_n, \emptyset)$ with $\mathbb{F}_n \neq \emptyset$ and also $\mathbb{P}_n \neq \emptyset$ as shown in Proposition 3.
- From the state $(\mathtt{C}_m, [I_1, \ldots, I_m], \mathbb{F}_m, \mathbb{P}_m, \mathbb{D}_m)$ with $\mathbb{F}_m \neq \emptyset$ and $(\mathcal{A}, \mathcal{B}) \in \mathbb{P}_m$, either $\mathsf{introspection}(\mathtt{C}_m, I_1, \mathcal{B}) = \textcircled{D}$ or $\mathsf{introspection}(\mathtt{C}_m, I_1, \mathcal{B}) = \textcircled{U}$. If $\mathsf{introspection}(\mathtt{C}_m, I_1, \mathcal{B}) = \textcircled{D}$, the only applicable transition rule is t_4 and leads to the state $(\mathtt{C}'_m, [I_1, \ldots, I_m, (\mathcal{A}, \mathcal{B}), I_{m+2}], \mathbb{F}_m, \mathbb{P}_m \setminus \{(\mathcal{A}, \mathcal{B})\}, \mathbb{D}_m)$. On the contrary, if $\mathsf{introspection}(\mathtt{C}_m, I_1, \mathcal{B}) = \textcircled{U}$, the only applicable transition rule is t_5 and leads to the state $(\mathtt{C}'_m, [I_1, \ldots, I_m, (\mathcal{A}, \mathcal{B}), I_{m+2}], \mathbb{F}_m, \mathbb{P}_m \setminus \{(\mathcal{A}, \mathcal{B})\}, \mathbb{D}_m \cup \{\mathcal{B}\})$. Note that the elements from $\mathbb{P}_m$ are removed one by one by the transition rules t_4 and t_5 until $\mathbb{P}_m$ is empty.
- From the state $(\mathtt{C}_o, [I_1, \ldots, I_o], \mathbb{F}_o, \emptyset, \mathbb{D}_o)$ with $\mathbb{F}_o \neq \emptyset$ and $\mathbb{D}_o \neq \emptyset$, the only applicable transition is t_6 with $\mathcal{A} \in \mathbb{D}_o$ and $\mathsf{expert\text{-}defeater}(I_1, \mathcal{A}) = (\mathcal{D}, T)$, and leads to the state $\mathtt{C}_o, [I_1, \ldots, I_o, \mathcal{A}, \mathcal{D}]$, $\mathbb{F}_o \cup \{\mathcal{D}\}, \emptyset, \mathbb{D}_o \setminus \{\mathcal{A}\}$. Note that the elements from $\mathbb{D}_o$ are removed one by one by the transition rule t_6 until $\mathbb{D}_o$ is empty.

Finally, by the transition rules t_1, t_2, t_3, t_4, t_5 and t_6 it holds that a session cannot evolve into any of the following states:

- $(\mathsf{C}, [\,], \mathbb{F}, \mathbb{P}, \mathbb{D})$ with $\mathbb{F} \neq \emptyset$.
- $(\mathsf{C}, [\,], \emptyset, \mathbb{P}, \mathbb{D})$ with $\mathbb{P} \neq \emptyset$ or $\mathbb{D} \neq \emptyset$.
- $(\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \mathbb{P}, \mathbb{D})$ with $\mathbb{P} \neq \emptyset$ or $\mathbb{D} \neq \emptyset$. □

Theorem 1. *Let s_i be an initial state, there exists a sequence of transitions that leads from s_i to $s_f = (\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$.*

Proof. Let us suppose that there does not exist a sequence of transitions that leads to s_f. Then, either:

(1) The session evolved to a state $s_j \neq s_f$ in which there are no applicable transition rules.
(2) The session can never evolve from the state $s_1 = (\mathsf{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \emptyset)$ with $\mathbb{F} \neq \emptyset$ to s_f, i.e., the justification I_1 can never be marked as Ⓤ. This implies that either:
   (a) Agent C has a denier $\mathcal{D}$ for some argument in $\mathbb{F}$ for which it cannot find an undefeated defeater.
   (b) $\mathbb{F}$ is infinite and the justification is always marked as Ⓓ, which implies that whenever C adopts a new pro argument which defeats a denier, a new denier arises.

By Lemma 1, condition (1) does not hold. Condition (a) is not possible: If the session is in the state $(\mathsf{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \mathbb{D})$ where $\mathcal{D} \in \mathbb{D}$, then $\mathcal{D}$ is defeating an argument $\mathcal{F} \in \mathbb{F}$. If $\mathcal{F} \in \mathbb{F}$, then the expert agent must warrant $\mathcal{F}$. In addition, for condition (a) to hold, the expert has to be able to construct $\mathcal{D}$. Otherwise, the answer for the preference question $(\mathcal{D}, \mathcal{F})$ would have been LES (less), C would have adjusted its preferences so that $\mathcal{F} >_{\mathsf{C}} \mathcal{D}$, and $\mathcal{D}$ would no longer be a denier for $\mathcal{F}$. Nevertheless, since the expert agent has $\mathcal{F}$ marked as Ⓤ, it must have an undefeated defeater for $\mathcal{D}$. Condition (b) is not possible: The expert's knowledge base is a DeLP program, and thus the number of facts and defeasible rules it contains is finite. Then, the number of arguments that the expert can construct is also finite. All the arguments in $\mathbb{F}$ are arguments sent by the expert. Therefore, $\mathbb{F}$ cannot be infinite. Contradiction. □

Corollary 1. *Let s_i be an initial state, the sequence of transitions that lead from s_i to the final session state $s_f = (\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$ is unique.*

Proof. According to Theorem 1 the sequence of transitions that leads from s_i to s_f must exist. In addition, according to Lemma 1 in each intermediate session state from s_i to s_f there is only one applicable transition rule. Therefore, the sequence of transitions from s_i to s_f is unique. □

Theorem 2. *Let $s_f = (\mathsf{C}, [I_1, \ldots, I_n], \emptyset, \emptyset, \emptyset)$ be a final state, the client agent* C *warrants* $\mathsf{claim}(I_1)$.

Proof. A session can only evolve into the state s_f from a state $s_1 = (\mathsf{C}, [I_1, \ldots, I_n], \mathbb{F}, \emptyset, \emptyset)$ with $\mathbb{F} \neq \emptyset$ when the transition rule t_2 is applicable, i.e., $\mathsf{introspection}(\mathsf{C}, I_1, I_1) = \text{Ⓤ}$. Then, from Definition 8, there exists a tree $\mathcal{T}$ such that $\mathsf{root}(\mathcal{T}) = I_1$ and $\mathsf{mark}(I_1, \mathcal{T}) = \text{Ⓤ}$. Therefore $\mathsf{claim}(I_1)$ is warranted by C. □

References

[1] R.A. Agis, S. Gottifredi and A.J. García, An approach for distributed discussion and collaborative knowledge sharing: Theoretical and empirical analysis, *Expert Systems with Applications* **116** (2019), 377–395. doi:10.1016/j.eswa.2018.09.016.

[2] L. Amgoud, C. Devred and M. Lagasquie-Schiex, Generating possible intentions with constrained argumentation systems, *Int. J. Approx. Reasoning* **52**(9) (2011), 1363–1391. doi:10.1016/j.ijar.2011.07.005.
[3] L. Amgoud, Y. Dimopoulos and P. Moraitis, A general framework for argumentation-based negotiation, in: *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007*, Honolulu, HI, USA, May 15, 2007, Revised Selected and Invited Papers, 2007, pp. 1–17.
[4] K. Atkinson, T.J.M. Bench-Capon and P. McBurney, A dialogue game protocol for multi-agent argument over proposals for action, *Autonomous Agents and Multi-Agent Systems* **11**(2) (2005), 153–171. doi:10.1007/s10458-005-1166-x.
[5] P. Bedi and P.B. Vashisth, Empowering recommender systems using trust and argumentation, *Inf. Sci.* **279** (2014), 569–586. doi:10.1016/j.ins.2014.04.012.
[6] T.J.M. Bench-Capon, Persuasion in practical argument using value-based argumentation frameworks, *J. Log. Comput.* **13**(3) (2003), 429–448. doi:10.1093/logcom/13.3.429.
[7] E. Black and A. Hunter, A generative inquiry dialogue system, in: *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, Honolulu, Hawaii, USA, May 14–18, 2007, p. 241.
[8] E. Black and A. Hunter, An inquiry dialogue system, *Autonomous Agents and Multi-Agent Systems* **19**(2) (2009), 173–209. doi:10.1007/s10458-008-9074-5.
[9] C.E. Briguez, M.C. Budán, C.A.D. Deagustini, A.G. Maguitman, M. Capobianco and G.R. Simari, Argument-based mixed recommenders and their application to movie suggestion, *Expert Syst. Appl.* **41**(14) (2014), 6467–6482. doi:10.1016/j.eswa.2014.03.046.
[10] Á. Carrera and C.A. Iglesias, A systematic review of argumentation techniques for multi-agent systems research, *Artif. Intell. Rev.* **44**(4) (2015), 509–535. doi:10.1007/s10462-015-9435-9.
[11] S. Coste-Marquis, S. Konieczny, J.-G. Mailly and P. Marquis, On the revision of argumentation systems: Minimal change of arguments statuses, in: *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.
[12] J. Devereux and C. Reed, Strategic argumentation in rigorous persuasion dialogue, in: *Argumentation in Multi-Agent Systems, 6th International Workshop, ArgMAS 2009*, Budapest, Hungary, May 12, 2009, Revised Selected and Invited Papers, 2009, pp. 94–113.
[13] M. Diller, A. Haret, T. Linsbichler, S. Rümmele and S. Woltran, An extension-based approach to belief revision in abstract argumentation, *International Journal of Approximate Reasoning* **93** (2018), 395–423. doi:10.1016/j.ijar.2017.11.013.
[14] S. Doutre, A. Herzig and L. Perrussel, A dynamic logic framework for abstract argumentation, in: *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.
[15] 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) (1995), 321–357. doi:10.1016/0004-3702(94)00041-X.
[16] P.M. Dung, R.A. Kowalski and F. Toni, Assumption-based argumentation, in: *Argumentation in Artificial Intelligence*, Springer, 2009, pp. 199–218. doi:10.1007/978-0-387-98197-0_10.
[17] X. Fan and F. Toni, Assumption-based argumentation dialogues, in: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, Barcelona, Catalonia, Spain, July 16–22, 2011, pp. 198–203.
[18] X. Fan and F. Toni, Agent strategies for ABA-based information-seeking and inquiry dialogues, in: *ECAI 2012 – 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track*, Montpellier, France, August 27–31, 2012, pp. 324–329.
[19] X. Fan and F. Toni, Mechanism design for argumentation-based information-seeking and inquiry, in: *International Conference on Principles and Practice of Multi-Agent Systems*, Springer, 2015, pp. 519–527.
[20] E. Ferretti, M. Errecalde, A.J. García and G.R. Simari, Decision rules and arguments in defeasible decision making, in: *Computational Models of Argument: Proceedings of COMMA 2008*, Toulouse, France, May 28–30, 2008, pp. 171–182.
[21] A.J. García and G.R. Simari, Defeasible logic programming: An argumentative approach, *TPLP* **4**(1–2) (2004), 95–138.
[22] A.J. García and G.R. Simari, Defeasible logic programming: DeLP-servers, contextual queries, and explanations for answers, *Argument & Computation* **5**(1) (2014), 63–88.
[23] S.A. Gómez, C.I. Chesñevar and G.R. Simari, ONTOarg: A decision support framework for ontology integration based on argumentation, *Expert Syst. Appl.* **40**(5) (2013), 1858–1870. doi:10.1016/j.eswa.2012.10.025.
[24] S.A. Gómez, A. Goron, A. Groza and I.A. Letia, Assuring safety in air traffic control systems with argumentation and model checking, *Expert Syst. Appl.* **44** (2016), 367–385. doi:10.1016/j.eswa.2015.09.027.
[25] S. Gottifredi, L.H. Tamargo, A.J. García and G.R. Simari, Arguing about informant credibility in open multi-agent systems, *Artif. Intell.* **259** (2018), 91–109. doi:10.1016/j.artint.2018.03.001.
[26] N.C. Karunatillake, N.R. Jennings, I. Rahwan and T.J. Norman, Argument-based negotiation in a social context, in: *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, Utrecht, The Netherlands, July 25–29, 2005, pp. 1331–1332.
[27] S. Kraus, Negotiation and cooperation in multi-agent environments, *Artif. Intell.* **94**(1–2) (1997), 79–97. doi:10.1016/S0004-3702(97)00025-8.

[28] V. Lifschitz, Foundations of logic programs, in: *Principles of Knowledge Representation*, G. Brewka, ed., CSLI Pub., 1996, pp. 69–128.
[29] S. Modgil and H. Prakken, The ASPIC+ framework for structured argumentation: A tutorial, *Argument & Computation* **5**(1) (2014), 31–62. doi:10.1080/19462166.2013.869766.
[30] M.O. Moguillansky, N.D. Rotstein, M.A. Falappa, A.J. García and G.R. Simari, Argument theory change through defeater activation, in: *COMMA*, 2010, pp. 359–366.
[31] S. Parsons, M. Wooldridge and L. Amgoud, An analysis of formal inter-agent dialogues, in: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, ACM, 2002, pp. 394–401. doi:10.1145/544741.544835.
[32] L. Perrussel, S. Doutre, J. Thévenin and P. McBurney, A persuasion dialog for gaining access to information, in: *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007*, Honolulu, HI, USA, May 15, 2007, Revised Selected and Invited Papers, 2007, pp. 63–79.
[33] H. Prakken, Formal systems for persuasion dialogue, *Knowledge Eng. Review* **21**(2) (2006), 163–188. doi:10.1017/S0269888906000865.
[34] H. Prakken, A.Z. Wyner, T.J.M. Bench-Capon and K. Atkinson, A formalization of argumentation schemes for legal case-based reasoning in ASPIC+, *J. Log. Comput.* **25**(5) (2015), 1141–1166. doi:10.1093/logcom/ext010.
[35] L. Riley, K. Atkinson, T.R. Payne and E. Black, An implemented dialogue system for inquiry and persuasion, in: *Theorie and Applications of Formal Argumentation – First International Workshop, TAFA 2011*, Barcelona, Spain, July 16–17, 2011, Revised Selected Papers, 2011, pp. 67–84.
[36] S.V. Rueda, A.J. García and G.R. Simari, Argument-based negotiation among BDI agents, *Journal of Computer Science & Technology* **2** (2002).
[37] G.R. Simari and I. Rahwan (eds), *Argumentation in Artificial Intelligence*, Springer, 2009.
[38] M. Snaith and C. Reed, Argument revision, *Journal of Logic and Computation* **27**(7) (2016), 2089–2134.
[39] L.H. Tamargo, A.J. García, M.A. Falappa and G.R. Simari, On the revision of informant credibility orders, *Artif. Intell.* **212** (2014), 36–58. doi:10.1016/j.artint.2014.03.006.
[40] L.H. Tamargo, S. Gottifredi, A.J. García and G.R. Simari, Sharing beliefs among agents with different degrees of credibility, *Knowl. Inf. Syst.* **50**(3) (2017), 999–1031. doi:10.1007/s10115-016-0964-6.
[41] J.C. Teze, S. Gottifredi, A.J. García and G.R. Simari, Improving argumentation-based recommender systems through context-adaptable selection criteria, *Expert Syst. Appl.* **42**(21) (2015), 8243–8258. doi:10.1016/j.eswa.2015.06.048.
[42] M. Thimm, Realizing argumentation in multi-agent systems using defeasible logic programming, in: *Argumentation in Multi-Agent Systems, 6th International Workshop, ArgMAS 2009*, Budapest, Hungary, May 12, 2009, Revised Selected and Invited Papers, 2009, pp. 175–194.
[43] M. Thimm, Strategic argumentation in multi-agent systems, *KI* **28**(3) (2014), 159–168.
[44] M. Thimm and A.J. García, On strategic argument selection in structured argumentation systems, in: *International Workshop on Argumentation in Multi-Agent Systems*, 2010, pp. 286–305.
[45] M. Thimm, A.J. Garcia, G. Kern-Isberner and G.R. Simari, Using collaborations for distributed argumentation with defeasible logic programming, in: *Proceedings of the 12th International Workshop on Non-Monotonic Reasoning (NMR'08)*, 2008, pp. 179–188.
[46] D. Walton, *Appeal to Expert Opinion: Arguments from Authority*, Penn State Press, 2010.
[47] D. Walton and E.C. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*, SUNY Press, 1995.