Realizing Argumentation in Multi-Agent Systems using Defeasible Logic Programming

Matthias Thimm

Information Engineering Group, Department for Computer Science,
Technische Universität Dortmund, Germany

Abstract. We describe a working multi-agent architecture based on *Defeasible Logic Programming* (DeLP) by García and Simari where agents are engaged in an argumentation to reach a common conclusion. Due to the distributed approach personalities and opinions of the individual agents give rise to arguments and counterarguments concerning a particular query. We establish a sound theoretical framework of a specific type of argumentation in multi-agent systems and describe the computational issues involved in it. The framework described in this paper has been fully implemented and a short description of its features is given.

Keywords: Argumentation, Multi-Agent Systems, Defeasible Logic Programming, Implementation

1 Introduction

Argumentation has become a very active field in computer science research [10, 4]. There are mainly two issues computational models of argumentation are concerned with in the context of artificial intelligence, namely describing models of commonsense reasoning techniques within a single agent and formalizing meaningful communication between agents in a multi-agent system, e. g. negotiation and persuasion [17, 2, 18, 5]. In this paper we take a hybrid approach by modeling a multi-agent system where the agents are capable of argumentation but use argumentation in order to reach a common conclusion that can be regarded as the whole system's opinion on the given topic.

As the underlying logical foundation we use *Defeasible Logic Programming* (DeLP) [13] which is a form of defeasible argumentation [21]. DeLP is an approach to realise non-monotonic reasoning via dialectical argumentation by relating arguments and counterarguments for a given logical query. A dialectical process that considers all arguments and counterarguments for the query is used in order to decide whether the query is believed by the agent or not.

This paper proposes and discusses an approach for a distributed system which provides the capability of argumentation using the notions of DeLP. In this system agents exchange arguments and counterarguments in order to answer queries given from outside the system. The framework establishes a border between its interior and exterior as from outside the system it is seen as a general reasoning engine. Internally this reasoning is accomplished by defeasible argumentation where every agent tries to support or defeat the given query by generating arguments for or against it and by generating counterarguments against other agents' arguments. In the end the most plausible argument

prevails and its conclusion is the answer to the original query. We build on previous work [25–27] but give a much more detailed description of the computational issues in multi-agent argumentation and some description on an implementation.

The paper is organized as follows. In Section 2 we give a brief overview on defeasible logic programming adapted to our needs. In Section 3 we formalize the multi-agent setting and give detailed logical descriptions of the individual components and continue with computational techniques that implement this formalization. We give a brief overview on the implementation of the proposed system afterwards in Section 5 and continue with a short review of related work in Section 6. In Section 7 we conclude with some final remarks.

2 Defeasible Logic Programming

The basic elements of *Defeasible Logic Programming* (DeLP) are facts and rules. Let $\mathcal{L}$ denote a set of ground literals, where a literal h is a ground atom A or a negated ground atom $\sim A$, where the symbol $\sim$ represents the strong negation. Overlining will be used to denote the complement of a literal with respect to strong negation, i. e., it is $\overline{p} = \sim p$ and $\overline{\sim p} = p$ for a ground atom p. A single literal $h \in \mathcal{L}$ is also called a *fact*.

The set of rules is divided into strict rules, i. e. rules encoding strict consequences, and defeasible rules which derive uncertain or defeasible conclusions. A *strict rule* is an ordered pair $h \leftarrow B$, where $h \in \mathcal{L}$ and $B \subseteq \mathcal{L}$. A *defeasible rule* is an ordered pair $h \prec B$, where $h \in \mathcal{L}$ and $B \subseteq \mathcal{L}$. A defeasible rule is used to describe tentative knowledge as in "birds fly". We use the functions $body/1$ and $head/1$ to refer to the head resp. body of a defeasible or strict rule. Strict and defeasible rules are ground. However, following the usual convention, some examples will use "schematic rules" with variables (denoted with an initial uppercase letter). Let DEF_X resp. STR_X be the set of all defeasible resp. strict rules, that can be constructed from literals in $X \subseteq \mathcal{L}$. We will omit the subscripts when referring to the whole set of literals $\mathcal{L}$, e. g. we write DEF for $\mathsf{DEF}_{\mathcal{L}}$.

Using facts, strict and defeasible rules, one is able to derive additional beliefs as in other rule-based systems. Let $X \subseteq \mathcal{L} \cup \mathsf{STR} \cup \mathsf{DEF}$ be a set of facts, strict rules, defeasible rules, and let furthermore $h \in \mathcal{L}$. A *(defeasible) derivation* of h from X, denoted $X \mathrel{|\!\sim} h$, consists of a finite sequence $h_1, \ldots, h_n = h$ of literals ($h_i \in \mathcal{L}$) such that h_i is a fact ($h_i \in X$) or there is a strict or defeasible rule in X with head h_i and body $b_1, \ldots, b_k$, where every b_l ($1 \leq l \leq k$) is an element h_j with $j < i$. If the derivation of a literal h only uses facts and strict rules, the derivation is called a *strict* derivation. A set X is *contradictory*, denoted $X \mathrel{|\!\sim} \bot$, iff there exist defeasible derivations for two complementary literals from X. Every agent in our framework maintains a *local belief base* that is comprised of defeasible rules and thus describes the agent's own (uncertain) knowledge. Furthermore the framework provides a *global belief base*, consisting of facts and strict rules, that describes common knowledge to all agents.

Definition 1 (Belief bases). *A* global belief base $\Pi \subseteq \mathcal{L} \cup \mathsf{STR}$ *is a non-contradictory set of strict rules and facts. A set of defeasible rules* $\Delta \subseteq \mathsf{DEF}$ *is called a* local belief base.

Observe, that we require the global belief base to be non-contradictory. This is justifiable as the information stored in the global belief base should be regarded as indisputable by the agents. Hence, the agents undertake their argumentation only based on their own local belief bases, which can be – in general – contradictory to each other.

Example 1. Let a global belief base Π and local belief bases Δ_1 and Δ_2 be given by

$$\Pi = \left\{ \begin{array}{l} chicken(tina) \\ scared(tina) \\ penguin(tweety) \\ bird(X) \leftarrow chicken(X) \\ bird(X) \leftarrow penguin(X) \\ \sim flies(X) \leftarrow penguin(X) \end{array} \right\},$$

$$\Delta_1 = \left\{ \begin{array}{l} flies(X) \prec bird(X) \\ flies(X) \prec chicken(X), scared(X) \end{array} \right\},$$

$$\Delta_2 = \left\{ \begin{array}{l} \sim flies(X) \prec chicken(X) \\ nests\_in\_trees(X) \prec flies(X) \end{array} \right\}.$$

The global belief base Π contains the facts, that Tina is a scared chicken and that Tweety is penguin. The strict rules state that all chickens and all penguins are birds, and penguins cannot fly. The defeasible rules of the local belief base Δ_1 express that birds and scared chickens normally fly. The defeasible rules of the local belief base Δ_2 express that chickens normally do not fly and something that flies normally nests in trees.

As a means to reveal different opinions about certain pieces of information, agents use their local belief bases to construct arguments.

Definition 2 (Argument, Subargument). *Let $h \in \mathcal{L}$ be a literal and let Π resp. Δ be a global resp. local belief base. $\langle \mathcal{A}, h \rangle$ is an* argument *for h, iff 1.) $\mathcal{A} \subseteq \Delta$, 2.) there exists a defeasible derivation of h from $\Pi \cup \mathcal{A}$, 3.) the set $\Pi \cup \mathcal{A}$ is non-contradictory, and 4.) $\mathcal{A}$ is minimal with respect to set inclusion. The literal h will be called* conclusion *and the set $\mathcal{A}$ will be called* support *of the argument $\langle \mathcal{A}, h \rangle$. An argument $\langle \mathcal{B}, q \rangle$ is a* subargument *of an argument $\langle \mathcal{A}, h \rangle$, iff $\mathcal{B} \subseteq \mathcal{A}$. Let $\mathsf{ARG}_{\Pi,\Delta}$ be the set of all arguments that can be built from Π and Δ.*

Two literals h and h_1 *disagree* regarding a global belief base Π, iff the set $\Pi \cup \{h, h_1\}$ is contradictory. Two complementary literals p and $\sim p$ disagree trivially, because for every Π the set $\Pi \cup \{p, \sim p\}$ is contradictory. But two literals which are not contradictory, can disagree as well. For $\Pi = \{(\sim h \leftarrow b), (h \leftarrow a)\}$ the literals a and b disagree, because $\Pi \cup \{a, b\}$ is contradictory. We call an argument $\langle \mathcal{A}_1, h_1 \rangle$ a *counterargument* to an argument $\langle \mathcal{A}_2, h_2 \rangle$ at a literal h, iff there is a subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ such that h and h_1 disagree. If $\langle \mathcal{A}_1, h_1 \rangle$ is a counterargument to $\langle \mathcal{A}_2, h_2 \rangle$ at a literal h, then the subargument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$ is called the *disagreement subargument*.

In order to deal with counterarguments to other arguments, a central aspect of defeasible logic programming is a formal comparison criterion among arguments. A possible preference relation among arguments is *Generalized Specificity* [24]. According to this

criterion an argument is preferred to another argument, iff the former one is more *specific* than the latter, i. e., (informally) iff the former one uses more facts or less rules. For example, $\langle\{c \prec a, b\}, c\rangle$ is more specific than $\langle\{\sim c \prec a\}, \sim c\rangle$. For a formal definition and desirable properties of preference criterions in general see [24, 13]. For the rest of this paper we use $\succ$ to denote an arbitrary but fixed preference criterion among arguments. The preference criterion is needed to decide whether an argument defeats another or not, as disagreement does not imply preference.

Definition 3 (Defeater). *An argument $\langle\mathcal{A}_1, h_1\rangle$ is a* defeater *of an argument $\langle\mathcal{A}_2, h_2\rangle$, iff there is a subargument $\langle\mathcal{A}, h\rangle$ of $\langle\mathcal{A}_2, h_2\rangle$ such that $\langle\mathcal{A}_1, h_1\rangle$ is a counterargument of $\langle\mathcal{A}_2, h_2\rangle$ at literal h and either $\langle\mathcal{A}_1, h_1\rangle \succ \langle\mathcal{A}, h\rangle$ (*proper defeat*) or $\langle\mathcal{A}_1, h_1\rangle \not\succ \langle\mathcal{A}, h\rangle$ and $\langle\mathcal{A}, h\rangle \not\succ \langle\mathcal{A}_1, h_1\rangle$ (*blocking defeat*).*

When considering sequences of arguments, the definition of defeat is not sufficient to describe a conclusive argumentation line. Defeat only takes an argument and its counterargument into consideration, but disregards preceeding arguments. But we expect also properties like *non-circularity* or *concordance* from an argumentation sequence. See [13] for a more detailed motivation of acceptable argumentation lines.

Definition 4 (Acceptable Argumentation Line). *Let Π be a global belief base. Let $\Lambda = [\langle\mathcal{A}_1, h_1\rangle, \ldots, \langle\mathcal{A}_m, h_m\rangle]$ be a sequence of some arguments. Λ is called an* acceptable argumentation line*, iff 1.) Λ is a finite sequence, 2.) every argument $\langle\mathcal{A}_i, h_i\rangle$ with $i > 1$ is a defeater of its predecessor $\langle\mathcal{A}_{i-1}, h_{i-1}\rangle$ and if $\langle\mathcal{A}_i, h_i\rangle$ is a blocking defeater of $\langle\mathcal{A}_{i-1}, h_{i-1}\rangle$ and $\langle\mathcal{A}_{i+1}, h_{i+1}\rangle$ exists, then $\langle\mathcal{A}_{i+1}, h_{i+1}\rangle$ is a proper defeater of $\langle\mathcal{A}_i, h_i\rangle$, 3.) $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_3 \cup \ldots$ is non-contradictory (*concordance of supporting arguments*), 4.) $\Pi \cup \mathcal{A}_2 \cup \mathcal{A}_4 \cup \ldots$ is non-contradictory (*concordance of interfering arguments*), and 5.)no argument $\langle\mathcal{A}_k, h_k\rangle$ is a subargument of an argument $\langle\mathcal{A}_i, h_i\rangle$ with $i < k$. Let* SEQ *denote the set of all sequences of arguments that can be built using rules from* DEF*,* STR *and facts from $\mathcal{L}$.*

We use the notation $\Lambda + \langle\mathcal{A}, h\rangle$ to denote the concatenation of argumentation lines and arguments.

In DeLP a literal h is *warranted*, if there is an argument $\langle\mathcal{A}, h\rangle$ which is non-defeated in the end. To decide whether $\langle\mathcal{A}, h\rangle$ is defeated or not, every acceptable argumentation line starting with $\langle\mathcal{A}, h\rangle$ has to be considered.

Definition 5 (Dialectical Tree). *Let Π be a global belief base and $\Delta_1, \ldots, \Delta_n$ be local belief bases. Let $\langle\mathcal{A}_0, h_0\rangle$ be an argument. A* dialectical tree *for $\langle\mathcal{A}_0, h_0\rangle$, denoted $\mathcal{T}_{\langle\mathcal{A}_0, h_0\rangle}$, is defined as follows.*

1. *The root of $\mathcal{T}$ is $\langle\mathcal{A}_0, h_0\rangle$.*
2. *Let $\langle\mathcal{A}_n, h_n\rangle$ be a node in $\mathcal{T}$ and let $\Lambda = [\langle\mathcal{A}_0, h_0\rangle, \ldots, \langle\mathcal{A}_n, h_n\rangle]$ be the sequence of nodes from the root to $\langle\mathcal{A}_n, h_n\rangle$. Let $\langle\mathcal{B}_1, q_1\rangle, \ldots, \langle\mathcal{B}_k, q_k\rangle$ be the defeaters of $\langle\mathcal{A}_n, h_n\rangle$. For every defeater $\langle\mathcal{B}_i, q_i\rangle$ with $1 \leq i \leq k$ such that the argumentation line $\Lambda' = [\langle\mathcal{A}_0, h_0\rangle, \ldots, \langle\mathcal{A}_n, h_n\rangle, \langle\mathcal{B}_i, q_i\rangle]$ is acceptable, the node $\langle\mathcal{A}_n, h_n\rangle$ has a child $\langle\mathcal{B}_i, q_i\rangle$. If there is no such $\langle\mathcal{B}_i, q_i\rangle$, the node $\langle\mathcal{A}_n, h_n\rangle$ is a leaf.*

Let DIA *denote the set of all dialectical trees with arguments that can be built using rules from* DEF*,* STR *and facts from $\mathcal{L}$.*

In order to decide whether the argument at the root of a given dialectical tree is defeated or not, it is necessary to perform a *bottom-up*-analysis of the tree. Every leaf of the tree is marked "undefeated" and every inner node is marked "defeated", if it has at least one child node marked "undefeated". Otherwise it is marked "undefeated". Let $\mathcal{T}^*_{\langle \mathcal{A},h \rangle}$ denote the marked dialectical tree of $\mathcal{T}_{\langle \mathcal{A},h \rangle}$.

We call a literal h *warranted*, iff there is an argument $\langle \mathcal{A}, h \rangle$ for h such that the root of the marked dialectical tree $\mathcal{T}^*_{\langle \mathcal{A},h \rangle}$ is marked "undefeated". Then $\langle \mathcal{A}, h \rangle$ is a *warrant* for h. Observe that, if a literal h is a fact or has a strict derivation from a global belief base Π alone, then h is also warranted as there are no counterarguments for $\langle \emptyset, h \rangle$. The answer of a DeLP interpreter to a literal h is YES iff h is warranted, NO iff $\overline{h}$ is warranted, and UNDECIDED iff neither h nor $\overline{h}$ are warranted. Notice, that it can not be the case that both h and $\overline{h}$ are warranted [28].

3 The formal Agent Architecture

Our framework consists of several agents and a central moderator, which coordinates the argumentation process undertaken by the agents. An overview of this system is depicted in Figure 1. The moderator accepts a query, consisting of a single literal, and asks the agents to argue about the warrant status of it, i. e., whether the literal or its negation can be supported by an ultimately undefeated argument. Agents use the global belief base of the system, which contains strict knowledge, and their own local belief bases consisting of defeasible knowledge to generate arguments. Eventually the system returns an answer to the questioner that describes the final status of the literal based on the agents' individual beliefs.

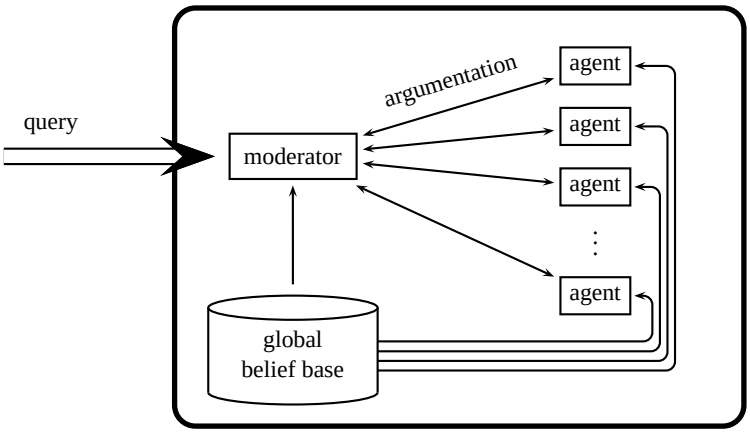

Fig. 1. An overview of the framework.

We now describe the components of the distributed framework, namely the moderator and the agents, using a functional description of their intended behaviour. As the

framework is flexible, many different definitions of the functions to be presented can be thought of. But we restrain them on the notions of DeLP as described above, so we use the subscript "D" to denote the DeLP specific definition. Furthermore we give specific algorithms describing the behavior of the system in the next section.

When the moderator receives arguments from the agents, he builds up several dialectical trees and finally he has to evaluate them using the bottom-up evaluation method described above.

Definition 6 (Analysis function χ_{D}). *The* analysis function χ_{D} *is a function* $\chi_{\mathsf{D}} : \mathsf{DIA} \to \{0,1\}$ *such that for every dialectical tree* $\upsilon \in \mathsf{DIA}$ *it holds* $\chi_{\mathsf{D}}(\upsilon) = 1$ *iff the root argument of* υ^* *is undefeated.*

Furthermore the evaluation of dialectical trees makes only sense, if the tree was built up according to the definition of an acceptable argumentation line. Hence, the moderator and the agents as well, have to check whether new arguments are valid in the current argumentation line.

Definition 7 (Acceptance function $\eta_{\mathsf{D},\succ}$). *For a given preference relation* $\succ$ *among arguments, the* acceptance function $\eta_{\mathsf{D},\succ}$ *is a function* $\eta_{\mathsf{D},\succ} : \mathsf{SEQ} \to \{0,1\}$ *such that for every argument sequence* $\Lambda \in \mathsf{SEQ}$ *it holds* $\eta_{\mathsf{D},\succ}(\Lambda) = 1$ *iff* Λ *is acceptable according to Definition 4.*

It is possible to assume different acceptance functions for different agents according to different definitions of an acceptable argumentation line. But in our multi-agent system, we assume $\eta_{\mathsf{D},\succ}$ to be fixed and the same for the moderator and all agents by convention.

At the end of the argumentation process for a query h, the agents have produced a set of dialectical trees with root arguments for h or $\overline{h}$, respectively. As we have to distinguish several different cases, the moderator has to decide, whether the query h is warranted, the negation of h is warranted, or none of them are warranted in the framework. Let $\mathfrak{P}(S)$ denote the power set of a set S.

Definition 8 (Decision function μ_{D}). *The* decision function μ_{D} *is a function* $\mu_{\mathsf{D}} : \mathfrak{P}(\mathsf{DIA}) \to \{\text{YES,NO,UNDECIDED,UNKNOWN}\}$. *Let* $Q_{\dot{p}} \subseteq \mathsf{DIA}$ *such that all root arguments of dialectical trees in* $Q_{\dot{p}}$ *are arguments for* p *or for* $\overline{p}$*, then* μ_{D} *is defined as*

1. $\mu_{\mathsf{D}}(Q_{\dot{p}}) = \text{YES}$, *if there is a dialectical tree* $\upsilon \in Q_{\dot{p}}$ *s. t. the root of* υ *is an argument for* p *and* $\chi_{\mathsf{D}}(\upsilon) = 1$.
2. $\mu_{\mathsf{D}}(Q_{\dot{p}}) = \text{NO}$, *if there is a dialectical tree* $\upsilon \in Q_{\dot{p}}$ *s. t. the root of* υ *is an argument for* $\overline{p}$ *and* $\chi_{\mathsf{D}}(\upsilon) = 1$.
3. $\mu_{\mathsf{D}}(Q_{\dot{p}}) = \text{UNDECIDED}$, *if* $\chi_{\mathsf{D}}(\upsilon) = 0$ *for all* $\upsilon \in Q_{\dot{p}}$.
4. $\mu_{\mathsf{D}}(Q_{\dot{p}}) = \text{UNKNOWN}$, *if* p *is not in the language* ($p \notin \mathcal{L}$).

The function μ_{D} is well-defined, as it cannot be the case that both conditions 1. and 2. are simultaneously fulfilled, see for example [28].

The above functions are sufficient to define the moderator of the framework.

Definition 9 (Moderator). *For a given preference relation* $\succ$ *among arguments, the* moderator *is a tuple* $(\mu_{\mathsf{D}}, \chi_{\mathsf{D}}, \eta_{\mathsf{D},\succ})$.

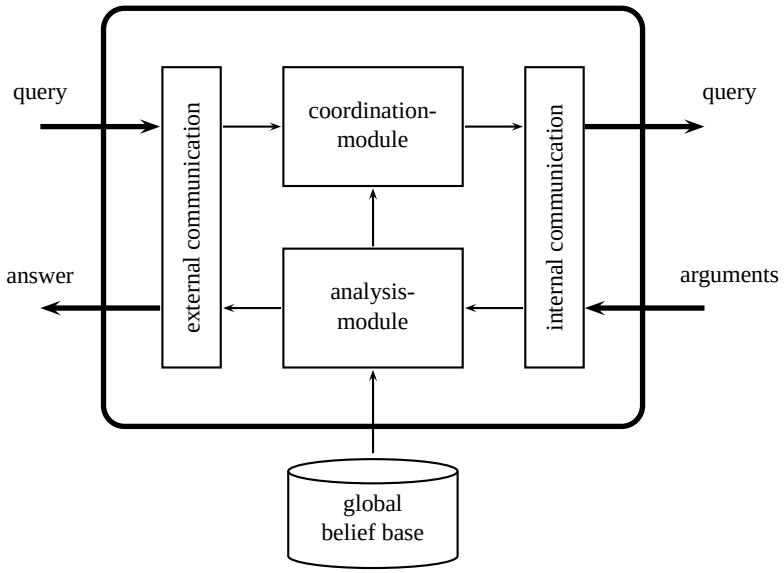

Fig. 2. The internal components of a moderator

An overview of the moderator is depicted in Figure 2. There, the analysis-module is responsible for the evaluation of the received arguments while the coordination-module is responsible for querying the agents for arguments in a systematic way. Furthermore, the moderator acts as an interface between the outside of the system and the system's interior by explicitly separating external (with the user) and internal communication (with the agents).

The agents of the framework provide two functionalities. First, they propose initial arguments for a given literal (or its negation) submitted by the moderator of the framework, which will be roots of the dialectical trees to be constructed. For a given query h it may be necessary to examine both, all dialectical trees with a root argument for h and all dialectical trees with a root argument for $\overline{h}$, as a query for h can only be answered with NO if there is a warrant for $\overline{h}$. Second, the agents propose counterarguments to arguments of other agents that are valid in the given argumentation line. We neglect the case that agents can give counterarguments to their own arguments here for simplicity. We achieve this by ensuring that each agent's local belief base is consistent with respect to the global belief base (see below). An agent is not obliged to return all his valid arguments for a given query or all his counterarguments for a given argument. Therefore, it is possible to model different kinds of argumentation strategies given different instantiations of the following argument functions.

Definition 10 (Root argument function). *Let Π be a global belief base and let Δ be a local belief base. A* root argument function *$\varphi_{\Pi,\Delta}$ is a function $\varphi_{\Pi,\Delta} : \mathcal{L} \rightarrow$*

$\mathfrak{P}(\mathsf{ARG}_{\Pi,\Delta})$ *such that for every literal* $h \in \mathcal{L}$ *the set* $\varphi_{\Pi,\Delta}(h)$ *is a set of arguments for* h *or for* $\overline{h}$ *from* Π *and* Δ*.*

Definition 11 (Counterargument function). *Let* Π *be a global belief base and let* Δ *be a local belief base. A* counterargument function $\psi_{\Pi,\Delta}$ *is a function* $\psi_{\Pi,\Delta} : \mathsf{SEQ} \to \mathfrak{P}(\mathsf{ARG}_{\Pi,\Delta})$ *such that for every argumentation sequence* $\Lambda \in \mathsf{SEQ}$ *the set* $\psi_{\Pi,\Delta}(\Lambda)$ *is a set of attacks from* Π *and* Δ *on the last argument of* Λ *and for every* $\langle \mathcal{B}, h\rangle \in \psi_{\Pi,\Delta}(\Lambda)$ *it holds that* $\eta_{\mathsf{D},\succ}(\Lambda + \langle \mathcal{B}, h\rangle) = 1$*.*

Here we assume that the root argument and counterargument functions of all agents are the same and especially *complete*, i. e, they return all possible arguments for the given situation and do not omit one.

Given the above definitions an agent of the framework is defined as follows.

Definition 12 (Agent). *An* agent *is a tuple* $(\Delta, \varphi_{\Pi,\Delta}, \psi_{\Pi,\Delta})$ *with a local belief base* Δ*, a root argument function* $\varphi_{\Pi,\Delta}$ *and a counterargument function* $\psi_{\Pi,\Delta}$*.*

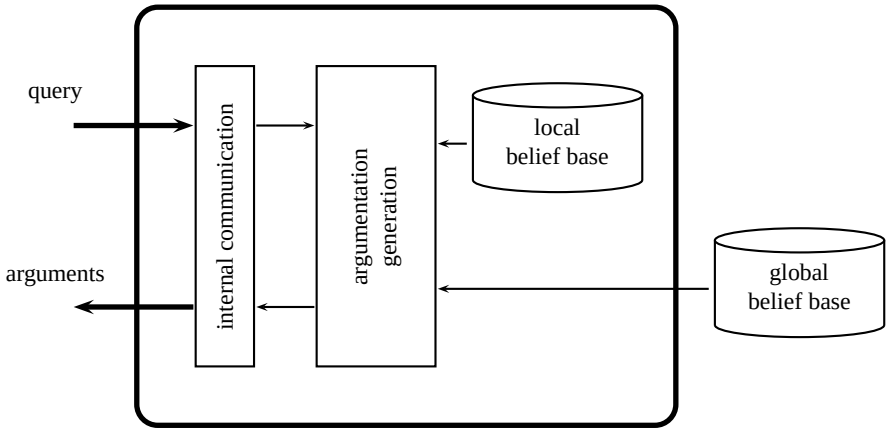

Fig. 3. An agent capable of argumentation

An overview of an agent is depicted in Figure 3.

Finally, the definition of a distributed argumentation system can be given as follows.

Definition 13 (Distributed argumentation system). *A* distributed argumentation system *is a tuple* $(M, \Pi, \{A_1, \ldots, A_n\})$ *with a moderator* M*, a global belief base* Π *and agents* $A_1, \ldots, A_n$*.*

For what is coming we assume that each agent's local belief base is consistent with Π, i. e., it is $\Pi \cup \Delta_i \not\vdash_{\sim} \bot$ for an agent A_i. By doing so, we forbid agents to counterargue their own arguments. Still, the union of the local belief bases of all agents and the global belief base may remain inconsistent, i. e. $\Pi \cup \Delta_1 \cup \ldots \cup \Delta_n \vdash_{\sim} \bot$ and thus gives rise to argumentation between the agents. We illustrate the above ideas with a simple example.

Example 2. Anna and Bob are planning for their holiday trip. They have already narrowed down the possible holiday destinations to $hawaii$ and $switzerland$, but as they can only afford for one trip the two possibilities are mutually exclusive ($\sim goto(hawaii) \leftarrow goto(switzerland)$ and $\sim goto(switzerland) \leftarrow goto(hawaii)$). Also, common knowledge includes that switzerland is a good place for skiing ($skiing(switzerland)$) an hawaii has access to the ocean ($ocean(hawaii)$). But they had already learned that hawaii also has a dangerous sea life ($dangerousSealife(hawaii)$). Furthermore, if they decide to go to switzerland, they can go by train ($goByTrain(switzerland)$) but have to get a ski pass ($needSkiPass(switzerland)$).

In order to decide whether to go to hawaii or switzerland, the different opinions of the two persons lead to a different structure of there local belief bases. While Anna likes swimming in the ocean ($goto(X) \prec swimming(X)$ and $swimming(X) \prec ocean(X)$), Bob prefers to ski ($goto(X) \prec skiing(X)$). He thinks also that a cheap holiday should be preferred ($goto(X) \prec cheap(X)$) and that going by train is a reasonable justification to consider a holiday cheap ($cheap(X) \prec goByTrain(X)$). Anna insists on her to have the possibility of swimming ($\sim goto(X) \prec \sim swimming(X)$), preferable in an ocean ($\sim swimming(X) \prec \sim ocean(X)$) and thinks that the need of ski pass does not constitute a trip to be cheap ($\sim cheap(X) \prec needSkiPass(X)$). Also, Bob thinks that a dangerous sea life in the ocean should prevent someone from swimming in it ($\sim swimming(X) \prec ocean(X), dangerousSealife(X)$).

In summary, the global belief base Π and the local belief bases of Bob (Δ_{Bob}) and Anna (Δ_{Anna}) that constitute the above described multi-agent system are given as follows:

$$
\begin{aligned}
\Pi = \quad & \{\ \sim goto(hawaii) \leftarrow goto(switzerland). \\
& \ \ \sim goto(switzerland) \leftarrow goto(hawaii). \\
& \ \ skiing(switzerland). \\
& \ \ ocean(hawaii). \\
& \ \ goByTrain(switzerland). \\
& \ \ dangerousSealife(hawaii). \\
& \ \ needSkiPass(switzerland). \quad \} \\
\Delta_{Bob} = \quad & \{\ goto(X) \prec skiing(X). \\
& \ \ goto(X) \prec cheap(X). \\
& \ \ cheap(X) \prec goByTrain(X). \\
& \ \ \sim swimming(X) \prec ocean(X), dangerousSealife(X). \quad \} \\
\Delta_{Anna} = \quad & \{\ goto(X) \prec swimming(X). \\
& \ \ swimming(X) \prec ocean(X). \\
& \ \ \sim goto(X) \prec \sim swimming(X). \\
& \ \ \sim swimming(X) \prec \sim ocean(X). \\
& \ \ \sim cheap(X) \prec needSkiPass(X). \quad \}
\end{aligned}
$$

We will continue this scenario in Example 3.

Given a system T and a query h, the framework produces an answer to h as follows. First, the moderator of T asks all agents for initial arguments for h and for $\overline{h}$ and starts a dialectical tree with each of them as root arguments. Then for each of these arguments, the moderator asks every agent for counterarguments and incorporates them into the corresponding dialectical trees accordingly. This process is repeated for every new argument until no more arguments can be constructed. Eventually the moderator analyses the resulting dialectical trees and returns the appropriate answer to the questioner. A dialectical tree built via this process is called an *argumentation product*. The answer behaviour of T is determined by the decision function of its moderator and is formalized as follows.

Definition 14 (Argumentation product). *Let $h \in \mathcal{L}$ be a query and $T = (M, \Pi, \{A_1, \ldots, A_n\})$ an distributed argumentation system with $M = (\mu_{\mathsf{D}}, \chi_{\mathsf{D}}, \eta_{\mathsf{D},\succ})$ and $A_i = (\Delta_i, \varphi_i, \psi_i)$ for $1 \leq i \leq n$. A dialectical tree $\upsilon \in$ DIA is called an* argumentation product *of T and h, iff the following conditions hold:*

1. *there exists a j with $1 \leq j \leq n$ such that the root of υ is an element of $\varphi_j(h)$, and*
2. *for every path $\Lambda = [\langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_n, h_n\rangle]$ in υ and the set K of child nodes of $\langle \mathcal{A}_n, h_n\rangle$ it holds $K = \{\langle \mathcal{B}, h'\rangle \mid \langle \mathcal{B}, h'\rangle \in \psi_1(\Lambda) \cup \ldots \cup \psi_n(\Lambda)$ and $\eta_{\mathsf{D},\succ}(\Lambda + \langle \mathcal{B}, h'\rangle) = 1\}$ (K is the set of all acceptable attacks on Λ).*

Example 3. We continue Example 2. Assume that *Generalized Specificity* is the chosen preference relation among arguments and let $goto(switzerland)$ be the query under consideration. Both agents, Anna and Bob, can put forward initial arguments for and against the query $goto(switzerland)$. For example, Bob has the argument $\langle \mathcal{A}, goto(switzerland)\rangle$ with

$$\mathcal{A} = \{ \; goto(switzerland) \prec cheap(switzerland), \; cheap(switzerland) \prec goByTrain(switzerland) \; \}$$

which makes use of the fact $goByTrain(switzerland)$. When asked for counterarguments to $\langle \mathcal{A}, goto(switzerland)\rangle$ Anna responds with $\langle \mathcal{B}_1, \sim goto(switzerland)\rangle$ and $\langle \mathcal{B}_2, \sim cheap(switzerland)\rangle$ with

$$\mathcal{B}_1 = \{ \; goto(hawaii) \prec swimming(hawaii), \; swimming(hawaii) \prec ocean(hawaii) \; \}$$

which makes use of the fact $ocean(hawaii)$ and the strict rule $\sim goto(switzerland) \leftarrow goto(hawaii)$ and

$$\mathcal{B}_2 = \{ \; \sim cheap(switzerland) \prec needSkiPass(switzerland) \; \}$$

which makes use of the fact $needSkiPass(switzerland)$. Observe that both arguments $\langle \mathcal{B}_1, \sim goto(switzerland)\rangle$ and $\langle \mathcal{B}_2, \sim cheap(switzerland)\rangle$ are blocking defeaters for $\langle \mathcal{A}, goto(switzerland)\rangle$. Hence, Bob can only bring forward the proper attack $\langle \mathcal{C}, \sim swimming(hawaii)\rangle$ with

$$\mathcal{C} = \{ \; \sim swimming(hawaii) \prec ocean(hawaii), dangerousSealife(hawaii) \}$$

to the argument $\langle \mathcal{B}_1, \sim goto(switzerland)\rangle$. All other possible counterarguments to $\langle \mathcal{B}_1, \sim goto(switzerland)\rangle$ or $\langle \mathcal{B}_2, \sim cheap(switzerland)\rangle$ would result in an unacceptable argumentation line. No other arguments can be brought forward by Anna and Bob and the resulting argumentation product is shown in Figure 4 (left). The analysis of the tree states that the root argument $\langle \mathcal{A}, goto(switzerland)\rangle$ is defeated due to its undefeated defeater $\langle \mathcal{B}_2, \sim cheap(switzerland)\rangle$.

Bob can bring forward another argument $\langle \mathcal{D}, goto(switzerland)\rangle$ for the initial query $goto(switzerland)$ with

$$\mathcal{D} = \{\ goto(switzerland) \prec skiing(switzerland)\ \}$$

that uses the fact $skiing(switzerland)$ and is brought forward by Bob. Anna can respond to $\langle \mathcal{D}, goto(switzerland)\rangle$ by stating again $\langle \mathcal{B}_1, \sim goto(switzerland)\rangle$. No other counterarguments can be brought forward by her. And again, Bob responds to $\langle \mathcal{D}, goto(switzerland)\rangle$ with the proper attack $\langle \mathcal{C}, \sim swimming(hawaii)\rangle$ and thus completes the only argumentation line in the current dialectical tree, see Figure 4 (right). The analysis of the resulting argumentation product reveals $\langle \mathcal{D}, goto(switzerland)\rangle$ to be undefeated and thus the answer of the system to the query $goto(switzerland)$ is YES.

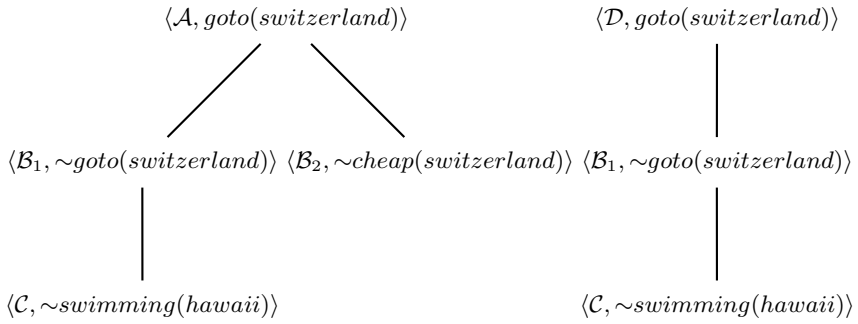

Fig. 4. Two argumentation products from Example 3.

4 Realizing Argumentation

After the theoretical elaboration of our framework in the previous section, we are now going describe the behavior of the agents and the whole system in terms of algorithms that implement the abstract functions defined above.

The control of the argumentation process is mainly handled by the moderator of the system. Given a specific query to the system, the moderator starts by asking each agent for arguments for or against the query's literal. Afterwards, for each initial argument the moderator builds recursively a dialectical tree by asking all agents for counterarguments to the intermediate "leafs" of the trees. If no agent can give any more counterarguments,

Algorithm 1 `RootArguments`

```
01 RootArguments(h,Δ,Π)
02     queryArguments = Arguments(h,Δ,Π);
03     nqueryArguments = Arguments(h̄,Δ,Π);
04     return queryArguments ∪ nqueryArguments;
```

the process finishes and the moderator analyses the given trees and returns the appropriate answer to the caller.

What follows is a description of the individual algorithms used in this process by the agents themselves, in particular implementations of the root argument, counterargument, and acceptance functions.

4.1 Generating root arguments

The first step in distributed argumentation in our framework consists of the generation of root arguments, i. e. arguments that form the root of a dialectical tree and directly refer to the given query. Let Π be a global belief base and Δ the local belief base of the agent under consideration. In order to retrieve a well-defined answer to the query h, all dialectical trees for both literals h and $\overline{h}$ have to be determined, as the non-existence of undefeated arguments for h does not automatically result in the answer NO. To distinguish the cases NO and UNDECIDED, one must verify the existence or non-existence of undefeated arguments for $\overline{h}$. Hence, if h is a query, the moderator asks all agents for arguments for h and for $\overline{h}$. The general algorithm to determine the arguments for and against the query is depicted in Algorithm 1. The algorithm uses the algorithm `Arguments` which is described below.

In order to determine all arguments for a literal h, given a global belief base Π and the local belief base Δ, the algorithm `Arguments` has to compute all possible derivations of h from Π and Δ. If h is a fact in Π then h has the sole argument $\langle \emptyset, h \rangle$ [13]. Otherwise the algorithm `Arguments` uses *backward-chaining* [9] to construct all possible arguments. The algorithm starts by searching for strict and defeasible rules with conclusion h. It then iteratively tries to find derivations of the body literals of the rules. The algorithm maintains a stack `S` that consists of tuples (R, L) with a set of rules R and a set of literals L. The element R contains the defeasible rules already added to this (partial) argument and L contains the literals that have not been derived yet. By constantly expanding these partial arguments with new rules from the agent's local belief base full arguments are being built in the component R. If L is empty the initial literal can be derived using the defeasible rules in R and the strict knowledge in Π. At the end of the algorithm non-minimal arguments are removed to meet the minimality condition of arguments. The complete algorithm can be seen in Algorithm 2.

Observe that in line 25/26 only the literals b_i are added to the set L that are not already derivable from the available rules. Partial arguments that cannot be completed are automatically dropped by the algorithm as no extension of them is added again to the stack S.

Algorithm 2 `Arguments`

```
01 Arguments(conclusion, Δ, Π)
02    if conclusion is a fact in Π then
03       return {⟨∅, conclusion⟩}
04    S = ∅
05    arguments = ∅
06    for each rule r: conclusion <-- b1,...,bn ∈ Δ ∪ Π do
07       if r is a defeasible rule then
08          Push ({r},{b1,...,bn}) on S
09       else
10          Push ({},{b1,...,bn}) on S
11    while S not empty do
12        Pop (R,L) from S
13        if L is empty then
14           arguments = arguments ∪ {⟨ R, conclusion ⟩}
15        else
16           Pop l from L
17           if l is a fact in Π then
18              Push (R,L) on S
19           else
20              for each rule r: l <-- b1,...,bn ∈ Δ ∪ Π do
21                 if r is a defeasible rule then
22                    R' = R ∪ {r}
23                 L' = L
24                 for each b_i with 1 ≤ i ≤ n do
25                    if b_i is not the head of a rule in R' then
26                       L' = L' ∪ {bi}
27                 Push (R',L') on S
28    for each a ∈ arguments do
29       if there exists a' ∈ arguments with a ≠ a'
30                              and a is a subargument of a'
31       arguments = arguments \ {a'}
32    return arguments
```

Based on the algorithm `RootArguments` we are able to define the root argument function $\varphi_{\Pi,\Delta}$ for DeLP.

Definition 15 ($\varphi_{\Pi,\Delta}$). *Let Π be a global belief base, Δ be a local belief base, and h a literal. Then the function $\varphi_{\Pi,\Delta} : \mathcal{L} \rightarrow \mathfrak{P}(\mathsf{ARG}_{\Pi,\Delta})$ is defined as*

$$\varphi_{\Pi,\Delta}(h) =_{def} \texttt{RootArguments(}h\texttt{,}\Delta\texttt{,}\Pi\texttt{)}.$$

The algorithm `Arguments` is sound and complete in the following sense (The proofs are omitted but can be found in [25]).

Proposition 1 (Soundness). *Let h be a literal, Δ a local belief base, and Π a global belief base. Then* `Arguments(`h`,`Δ`,`Π`)` *is a set of arguments for h.*

Algorithm 3 `PCL`

```
01 PCL(⟨A,h⟩, Δ, Π)
02     pcl_1 = {h̄ | h ≺ B ∈ A}
03     pcl_2 = {f | Π ∪ Δ ∪ pcl_1 |~ f, Π ∪ Δ |/~ f}
04     return pcl_1 ∪ pcl_2
```

Proposition 2 (Completeness). *Let h be a literal, Δ a local belief base, and Π a global belief base. Then for every argument $\langle \mathcal{A}, h\rangle$ with respect to Π and $\mathcal{A} \subseteq \Delta$ it is $\langle \mathcal{A}, h\rangle \in$* `Arguments(`$h, \Delta, \Pi$`)`.

The soundness and completeness of the algorithm `RootArguments` follows directly.

4.2 Generating counterarguments

Let $\Lambda = (\langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_n, h_n\rangle)$ be an argumentation line. Another important task of an agent is to propose counterarguments $\langle \mathcal{B}, b\rangle$ for $\langle \mathcal{A}_n, h_n\rangle$, such that $\Lambda' = \Lambda + \langle \mathcal{B}, b\rangle$ is an acceptable argumentation line. To describe the generation of counterarguments in an algorithmic manner we need the notion of *potentially counterarguing literals*. Let again Δ be the local belief base of the agent under consideration.

Definition 16 (Potentially counterarguing literals). *Let Π be a global belief base, Δ a local belief base, and $\langle \mathcal{A}, h\rangle$ an argument. Then the set of* potentially counterarguing literals $pcl_{\Pi,\Delta}(\langle \mathcal{A}, h\rangle)$ *is defined by*

$$pcl_{\Pi,\Delta}(\langle \mathcal{A}, h\rangle) = \{f \mid \Pi \cup \mathcal{A} \cup \{f\} \mathrel{|\!\sim} \bot\}.$$

Hence, for every conclusion h' of a counterargument $\langle \mathcal{B}, h'\rangle$ with $\mathcal{B} \subseteq \Delta$ to $\langle \mathcal{A}, h\rangle$ it must hold $h' \in pcl_{\Pi,\Delta}(\langle \mathcal{A}, h\rangle)$.Therefore it is sufficient to look only for potential counterarguments among the arguments with conclusion in $pcl_{\Pi,\Delta}(\langle \mathcal{A}, h\rangle)$.

The set $pcl_{\Pi,\Delta}(\langle \mathcal{A}, h\rangle)$ can be characterized as follows. Let $\mathcal{A} = \{h_1 \prec B_1, \ldots, h_m \prec B_m\}$, then all literals $\overline{h_1}, \ldots, \overline{h_m}$ are potentially counterarguing literals, as for every h_i ($1 \leq i \leq n$), $\langle \mathcal{A}, h\rangle$ contains a subargument for h_i. Furthermore, due to the derivation of the literals $\{h_1, \ldots, h_n\}$ by $\langle \mathcal{A}, h\rangle$ strict rules in Π might get "fired". Negations of the conclusions of these strict rules are also potentially counterarguing literals. Algorithm 3 describes this computation.

In order to validate the acceptability of potential counterarguments within the given argumentation line, the algorithm `Acceptable` (see Algorithm 4) must be applied, which is a straightforward implementation of Definition 4. In the algorithm, $\succ$ is an arbitrary preference relation, e. g. *Generalized Specificity* [24].

Based on the algorithm `Acceptable` the acceptance function $\eta_{\mathsf{D},\succ}$ can be defined as follows.

Definition 17 ($\eta_{\mathsf{D},\succ}$). *Let Π be a global belief base, Δ be a local belief base, Λ be an argumentation line, and $\langle \mathcal{A}, h\rangle$ be an argument. The function $\eta_{\mathsf{D},\succ} : \Sigma(\Omega) \to \{0, 1\}$ is defined as*

$$\eta_{\mathsf{D},\succ}(\Lambda + \langle \mathcal{A}, h\rangle) =_{def} \begin{cases} 1 \text{ if } \texttt{Acceptable(}\Lambda, \langle \mathcal{A}, h\rangle, \Pi\texttt{)}{=}true \\ 0 \text{ otherwise} \end{cases}.$$

Algorithm 4 `Acceptable`

```
01 Acceptable([⟨A_1, h_1⟩, ..., ⟨A_n, h_n⟩], ⟨B, h⟩, Π)
02    let ⟨A', h'⟩ be the disagreement sub-argument
03          of ⟨A_n, h_n⟩ relative to ⟨B, h⟩
04    if B ⊆ A_j for one 1 ≤ j ≤ n then
05       return false
06    if n is even then
07       if A_1 ∪ A_3 ... ∪ A_{n-1} ∪ B ∪ Π |~ ⊥ then
08          return false
09    if n is odd then
10       if A_2 ∪ A_4 ... ∪ A_n ∪ B ∪ Π) |~ ⊥ then
11          return false
12    if ⟨A', h'⟩ ≻ ⟨B, h⟩ then
13       return false
14    if n > 1 then
15       if ⟨A_n, h_n⟩ and ⟨A_{n-1}, h_{n-1}⟩ are incomparable with
16                                 respect to ≻ then
17          if not ⟨B, h⟩ ≻ ⟨A', h'⟩ then
18             return false
19    return true
```

Algorithm 5 `Attacks`

```
01 Attacks((⟨A_1, s_1⟩, ..., ⟨A_n, s_n⟩), Δ, Π)
02    pcl = PCL(⟨A_n, s_n⟩, Δ, Π)
03    result = ∅
04    for each d ∈ pcl do
05       arguments = Arguments(d, Δ, Π)
06       for each ⟨B, d⟩ ∈ arguments do
07          if Acceptable((⟨A_1, s_1⟩, ..., ⟨A_n, s_n⟩), ⟨B, d⟩, Π) then
08             result = result ∪ {⟨B, d⟩}
09    return result
```

Given a global belief base Π, a local belief base Δ and an argumentation line $\Lambda = [\langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_n, h_n\rangle]$ the algorithm `Attacks` (see Algorithm 1.5) uses the algorithm `Arguments` to compute all arguments with conclusions in $pcl_{\Pi,\Delta}(\langle \mathcal{A}, h\rangle)$. All these arguments that are acceptable regarding Λ are added to the result set.
Using algorithm `Attacks` the counterargument function$\psi_{\Pi,\Delta}$ can be defined as follows.

Definition 18 ($\psi_{\Pi,\Delta}$). *Let Π be a global belief base, Δ be a local belief base, and Λ be an argumentation line. The function $\psi_{\Pi,\Delta} : \Sigma \to \mathfrak{P}(\Omega)$ is defined as*

$$\psi_{\Pi,\Delta}(\Lambda) =_{def} \texttt{Attacks(}\Lambda\texttt{,}\Delta\texttt{,}\Pi\texttt{)}.$$

The soundness and completeness of the algorithm `Attacks` follows directly from Propositions 1 and 2.

5 Implementation

The system described in this paper has been fully implemented in Java [14] and can be directly obtained from the author[1]. Besides the general argumentation capabilities described above, also the comparison relation *Generalized Specificity* [24] has been implemented. This has been done using its characterization by activation sets [24]. The framework also supports the representation of P-DeLP [1], which is an extension of DeLP using a possibilistic language. Within P-DeLP defeasible rules are annotated with reals that measure the certainty of the rules. The comparison relation for arguments in P-DeLP derives naturally from the annotated numbers by aggregating the annotations of all rules in an argument and using these as necessity measures. Therefore, the implemented framework features two powerful representation languages for defeasible argumentation and two comparison relations for arguments.

The framework allows the specification of local belief bases of an arbitrary number of agents and the specification of the global belief base within the chosen language. The user can query the system for the warrant status of literals and the result of the argumentation process is visualized as a set of dialectical trees.

The framework has been applied to a real world example involving two agents acting as accuser and defender in a legal case [25]. There, the specific setting of the multi-agent scenario in our framework has a real-world analogy (at least in german law, see [25]). Both, accuser and defender state arguments for and against a specific claim, for example the guilt or innocence of a given accused, but these arguments are evaluated by a neutral moderator, in this case the judge.

6 Related Work

The research on argumentation in multi-agent systems is a very active field, see for example the annually ArgMAS workshop [22, 17]. Current research includes besides others argumentation-based negotiation approaches [16, 23, 15, 2], persuasion [5, 19] and general dialogue formalizations [3, 7, 8]. All these approaches are related to the framework developed here regarding the aim of formalizing agent interaction in form of argumentation. But, to our knowledge, the framework of Black et al. [7, 8] is the only one which also uses defeasible logic programming as the underlying representation formalism to model distributed argumentation. Complementary to the proposal in this paper, the focus of [7] is on modeling communication protocols and strategies for successful argumentation between agents. They introduce two kinds of inquiry dialogues, one to generate combined arguments and one for the actual argumentation.

The framework of [11] uses extended logic programs to model an agent's belief and defines a notion of distributed argumentation using these extended logic programs. The framework uses the argumentation semantics from [20] and defines a notion of cooperation, that allows the agents to share their beliefs in order to construct new arguments. As this framework uses extended logic programs as the underlying representation formalism, it has a declarative semantics in contrast to the dialectical semantics of DeLP

[1] matthias.thimm@tu-dortmund.de

used here. Yet, in another work [26] we also extended the framework described here by introducing *collaborations* that allow the agents to share their beliefs and construct new arguments.

7 Final Remarks

We have developed a multi-agent architecture that uses argumentation in order to reach a common conclusion acceptable by all agents. The framework uses *Defeasible Logic Programming* as the underlying argumentation mechanism but distributes the beliefs among several agents. We have given a functional formalization of the system and described the computational issues involved in implementing it. The framework has successfully been implemented and applied to a real world example.

Ongoing research includes collaborations in the multi-agent setting [26], security issues in agent interactions [6] and a generalization of the framework to abstract argumentation systems. The complex dialectical semantics of DeLP does not offer a quite understandable anticipation of the interaction of arguments. Thus we aim at extending the described framework to abstract argumentation systems [12] in order to enrich it with a declarative semantics.

References

1. Teresa Alsinet, Carlos I. Chesñevar, Lluis Godo, and Guillermo R. Simari. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems (FSS)*, 2008.
2. Leila Amgoud, Yannis Dimopoulos, and Pavlos Moraitis. A general framework for argumentation-based negotiation. In I. Rahwan, S. Parsons, and C. Reed, editors, *Fourth International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2007*, volume 4946 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2007.
3. Katie Atkinson, Trevor Bench-Capon, and Peter McBurney. A dialogue game protocol for multi-agent argument over proposals for action. In *Journal of Autonomous Agents and Multi-Agent Systems*, pages 149–161. Springer, 2004.
4. T. J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171:619–641, 2007.
5. T.J.M. Bench-Capon. Persuasion in practical argument using value based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.
6. Joachim Biskup, Gabriele Kern-Isberner, and Matthias Thimm. Towards enforcement of confidentiality in agent interactions. In M. Pagnucco and M. Thielscher, editors, *Proceedings of the 12th International Workshop on Non-Monotonic Reasoning (NMR'08)*, pages 104–112, September 2008.
7. E. Black. *A Generative Framework for Argumentation-Based Inquiry Dialogues*. PhD thesis, University College London, 2007.
8. E. Black and A. Hunter. A generative inquiry dialogue system. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07)*. IEEE Press, 2007.
9. C. Chesñevar, G. Simari, and A. García. Making argument system computationally attractive. In *Proceedings of the XIII International Conference of the Chilean Society for Computer Science*, 1993.

10. Carlos I. Chesñevar, Ana G. Maguitman, and Ronald P. Loui. Logical models of argument. *ACM Comput. Surv.*, 32(4):337–383, 2000.
11. Iara Carnevale de Almeida and José Júlio Alferes. An argumentation-based negotiation for distributed extended logic programs. In *Proceedings of CLIMA VII*, pages 191–210, 2006.
12. Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.
13. A. García and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138, 2004.
14. James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java language specification*. Addison-Wesley, third edition, 2005.
15. Nishan C. Karunatillake, Nicholas R. Jennings, Iyad Rahwan, and Timothy J. Norman. Argument-based negotiation in a social context. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1331–1332, New York, NY, USA, 2005. ACM.
16. Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–97, 1997.
17. Nicolas Maudet, Simon Parsons, and Iyad Rahwan. Argumentation in multi-agent systems: Context and recent developments. In *Third International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2006*, volume 4766 of *LNCS*, pages 1–16. Springer, 2007.
18. Simon Parsons, Carles Sierra, and Nick Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.
19. Laurent Perrussel, Sylvie Doutre, Jean-Marc Thévenin, and Peter McBurney. A persuasion dialog for gaining access to information. In I. Rahwan, S. Parsons, and C. Reed, editors, *Fourth International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2007*, volume 4946 of *Lecture Notes in Computer Science*, pages 63–79. Springer, 2007.
20. Henry Prakken. Dialectical proof theory for defeasible argumentation with defeasible priorities (preliminary report). In *ModelAge Workshop*, pages 202–215, 1997.
21. Henry Prakken and Gerard Vreeswijk. Logical systems for defeasible argumentation. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer Academic Publishers, second edition, 2002.
22. I. Rahwan, S. Parsons, and C. Reed, editors. *Fourth International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2007*, Honolulu, USA, May 2007. Springer.
23. Sonia V. Rueda, A. Garcia, and Guillermo R. Simari. Argument-based negotiation among BDI agents. *Journal of Computer Science and Technology*, 2(7), 2002.
24. F. Stolzenburg, A. García, Carlos I. Chesnevar, and G. Simari. Computing generalized specificity. *Journal of Non-Classical Logics*, 13(1):87–113, 2003.
25. Matthias Thimm. *Verteilte logikbasierte Argumentation: Konzeption, Implementierung und Anwendung im Rechtswesen*. VDM Verlag Dr. Müller, 2008.
26. Matthias Thimm, Alejandro J. Garcia, Gabriele Kern-Isberner, and Guillermo R. Simari. Using collaborations for distributed argumentation with defeasible logic programming. In M. Pagnucco and M. Thielscher, editors, *Proceedings of the 12th International Workshop on Non-Monotonic Reasoning (NMR'08)*, pages 179–188, 2008.
27. Matthias Thimm and Gabriele Kern-Isberner. A distributed argumentation framework using defeasible logic programming. In P. Besnard, S. Doutre, and A. Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA'08)*, pages 381–392. IOS Press, 2008.
28. Matthias Thimm and Gabriele Kern-Isberner. On the relationship of defeasible argumentation and answer set programming. In P. Besnard, S. Doutre, and An Hunter, editors, *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA'08)*, pages 393–404. IOS Press, 2008.