# A Distributed Argumentation Framework using Defeasible Logic Programming

Matthias Thimm [a] Gabriele Kern-Isberner [a]
[a] *Information Engineering Group, Faculty of Computer Science*
*Technische Universität Dortmund, Germany*

**Abstract.** *Defeasible Logic Programming* (DeLP) by García and Simari is an approach to realise non-monotonic reasoning via dialectical argumentation. We extend their approach by explicitly supporting distributed entities in the argumentation process on a structural basis. This makes the modelling of distributed argumentation systems like a jury court by using DeLP techniques possible. In this framework possibly many different agents with different opinions argue with each other on a given logical subject. We compare our framework with general DeLP and present the results.

**Keywords.** Logic Programming, Argumentation, Defeasible Argumentation, Distributed Argumentation, Multi Agent Systems.

## 1. Introduction

Mimicking *commonsense-reasoning* using non-monotonic logics is one of the main topics in AI. *Defeasible Logic Programming* (DeLP) [5] 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. So in DeLP argumentation is treated as an internal deliberation mechanism of one agent to determine the set of pieces of information which are most believed.

But in general the term *argumentation* is much more abstract. It can also be the exchange of arguments and counterarguments between several agents in a multi agent environment where every agent tries to convince other agents of a specific opinion. Consider a jury court, where every juror has a personal opinion about the guilt or innocence of an accused person. The jurors give arguments for one or the other and attack other jurors' arguments with counterarguments. In the end one argument may prevail and its conclusion is given to the initiator of the query, e. g. the judge.

There are many approaches to realize negotiation in multi agent systems. Whereas in [1,7] and especially in [2], the focus is on using argumentation for

persuasion, in this paper we use argumentation to reach a common conclusion of a group of agents. Considering the jury court it is reasonable to assume that there are jurors who are less competent in jurisdiction than others. However it is the main goal to reach an agreement regarding the given case rather than unifying the jurors beliefs.

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.

The rest of this paper is structured as follows: in Section 2 we give a brief overview on DeLP. In Section 3 a framework for modelling distributed argumentation using DeLP is proposed. Section 4 compares the framework with general DeLP and in Section 5 we conclude.

## 2. Defeasible Argumentation

Defeasible Logic Programming (DeLP) [5] is a logic programming language which is capable of modelling defeasible knowledge. With the use of a defeasible argumentation process it is possible to derive conclusive knowledge.

The basic elements of DeLP are facts and rules. The set of rules is divided into strict rules, i. e. rules which derive certain knowledge, and defeasible rules, i. e. rules which derive uncertain or defeasible knowledge. We use a first-order language without function symbols except constants, so let $\mathfrak{L}$ be a set of literals, where a literal $h$ is atom $A$ or a negated atom $\neg A$, where the symbol $\neg$ represents the strong logic negation. Overlining will be used to denote the complement of a literal with respect to strong negation, i. e. it is $\overline{p} = \neg p$ and $\overline{\neg p} = p$ for a ground atom $p$.

**Definition 1** (Fact, strict rule, defeasible rule)**.** A *fact* is a literal $h \in \mathfrak{L}$. A *strict rule* is an ordered pair $h \leftarrow B$, where $h \in \mathfrak{L}$ and $B \subseteq \mathfrak{L}$. A *defeasible rule* is an ordered pair $h \prec B$, where $h \in \mathfrak{L}$ and $B \subseteq \mathfrak{L}$.

A defeasible rule is used to describe uncertain 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.

**Definition 2** (Defeasible Logic Program)**.** A *Defeasible Logic Program* $\mathcal{P} = (\Pi, \Delta)$, abbreviated *de.l.p.*, consists of a (possibly infinite) set $\Pi$ of facts and strict rules and of a (possibly infinite) set $\Delta$ of defeasible rules.

**Example 1** ([5], example 2.1)**.** Let $\mathcal{P} = (\Pi, \Delta)$ be given by

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

$$\Delta = \left\{ \begin{array}{l} flies(X) \prec bird(X) \\ \neg flies(X) \prec chicken(X) \\ flies(X) \prec chicken(X), scared(X) \\ nests\_in\_trees(X) \prec flies(X) \end{array} \right\} .$$

In the following examples we abbreviate the above predicates by their first letters, e. g. in the following the predicate $c/1$ stands for $chicken/1$.

A *de.l.p.* $\mathcal{P} = (\Pi, \Delta)$ describes the beliefbase of an agent and therefore contains not all of its beliefs. With the use of strict and defeasible rules it is possible to derive other literals, which may be in the agent's state of belief.

**Definition 3** (Defeasible Derivation)**.** Let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.* and let $h \in \mathfrak{L}$. A *(defeasible) derivation* of $h$ from $\mathcal{P}$, denoted $\mathcal{P} \mathrel{|\!\sim} h$, consists of a finite sequence $h_1, \ldots, h_n = h$ of literals ($h_i \in \mathfrak{L}$) such that $h_i$ is a fact ($h_i \in \Pi$) or there exists a strict or defeasible rule in $\mathcal{P}$ 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 $j < i$. Let $\mathcal{F}(\mathcal{P})$ denote the set of all literals that have a defeasible derivation from $\mathcal{P}$.

If the derivation of a literal $h$ only uses strict rules, the derivation is called a *strict* derivation.

As facts and strict rules describe strict knowledge, it is reasonable to assume $\Pi$ to be non-contradictory, i. e. there are no derivations for complementary literals from $\Pi$ only. But if $\Pi \cup \Delta$ is contradictory (denoted $\Pi \cup \Delta \mathrel{|\!\sim} \bot$), then there exist defeasible derivations for two complementary literals.

**Definition 4** (Argument, Subargument)**.** Let $h \in \mathfrak{L}$ be a literal and let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.*. $\langle \mathcal{A}, h \rangle$ is an *argument* for $h$, iff $\mathcal{A} \subseteq \Delta$, there exists a defeasible derivation of $h$ from $\mathcal{P}' = (\Pi, \mathcal{A})$, the set $\Pi \cup \mathcal{A}$ is non-contradictory and $\mathcal{A}$ 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 $\mathcal{B} \subseteq \mathcal{A}$.

**Example 2.** In the *de.l.p.* $\mathcal{P}$ from Example 1 the literal $f(tina)$ has the two arguments: $\langle \{ f(tina) \prec b(tina) \}, f(tina) \rangle$ and $\langle \{ f(tina) \prec c(tina), s(tina) \}, f(tina) \rangle$.

**Definition 5** (Disagreement)**.** Let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.*. Two literals $h$ and $h_1$ *disagree*, iff the set $\Pi \cup \{h, h_1\}$ is contradictory.

Two complementary literals $p$ und $\neg p$ disagree trivially, because for every *de.l.p.* $\mathcal{P} = (\Pi, \Delta)$ the set $\Pi \cup \{p, \neg p\}$ is contradictory. But two literals which are not contradictory, can disagree either. For $\Pi = \{(\neg h \leftarrow b), (h \leftarrow a)\}$ the literals $a$ and $b$ disagree, because $\Pi \cup \{a, b\}$ is contradictory.

**Definition 6** (Counterargument)**.** An argument $\langle \mathcal{A}_1, h_1 \rangle$ is a *counterargument* to an argument $\langle \mathcal{A}_2, h_2 \rangle$ at a literal $h$, iff there exists 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*. If $h = h_2$, then $\langle \mathcal{A}_1, h_1 \rangle$ is called a *direct attack* on $\langle \mathcal{A}_2, h_2 \rangle$ and *indirect attack*, otherwise.

**Example 3.** In $\mathcal{P}$ from Example 1 there is $\langle \{\neg f(tina) \prec c(tina)\}, \neg f(tina) \rangle$ a direct attack to $\langle \{f(tina) \prec b(tina)\}, f(tina) \rangle$. Furthermore $\langle \{\neg f(tina) \prec c(tina)\}, \neg f(tina) \rangle$ is an indirect attack on $\langle \{(n(tina) \prec f(tina)), (f(tina) \prec b(tina))\}, n(tina) \rangle$ with the disagreement subargument $\langle \{(f(tina) \prec b(tina))\}, f(tina) \rangle$.

A central aspect of defeasible argumentation is a formal comparison criterion among arguments. For some examples of preference criterions see [5]. For the rest of this paper we use an abstract preference criterion $\succ$ defined as follows.

**Definition 7** (Preference Criterion $\succ$)**.** A preference criterion among arguments is an irreflexive, antisymmetric relation and will be denoted by $\succ$. If $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ are arguments, $\langle \mathcal{A}_1, h_1 \rangle$ will be *strictly preferred* over $\langle \mathcal{A}_2, h_2 \rangle$, iff $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$.

**Example 4.** A possible preference relation among arguments is *Generalized Specificty* [8]. 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 $\langle \{\neg c \prec a\}, \neg c \rangle$. For a formal definition see [8,5].

As $\succ$ is antisymmetric by definition, there cannot be an equipreference among an argument and its counterargument. So we only have to consider the cases, that one argument is better than the other or that two arguments are incomparable with $\succ$.

**Definition 8** (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, then 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 [5] for a more detailed description of acceptable argumentation lines.

**Definition 9** (Acceptable Argumentation Line)**.** Let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.* and let $\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \mathcal{A}_n, h_n \rangle]$ be a sequence of arguments. $\Lambda$ is called *acceptable argumentation line*, iff 1.) $\Lambda$ is a finite sequence, 2.) every argument $\langle \mathcal{A}_i, h_i \rangle$ with $i > 1$ is a defeater of his 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 $+$ denote the concatenation of argumentation lines and arguments, e. g. $[\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle] + \langle \mathcal{B}, h \rangle$ stands for $[\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \langle \mathcal{B}, h \rangle]$.

In DeLP a literal $h$ is *warranted*, if there exists 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 10** (Dialectical Tree)**.** Let $\langle \mathcal{A}_0, h_0 \rangle$ be an argument of a *de.l.p.* $\mathcal{P} = (\Pi, \Delta)$. A *dialectical tree* for $\langle \mathcal{A}_0, h_0 \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, is defined by

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.

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. There 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 $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$.

**Definition 11** (Warrant)**.** A literal $h \in \mathfrak{L}$ is *warranted*, iff there exists 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 $h$.

If a literal $h$ is a fact or has a strict derivation from a *de.l.p.*, then $h$ is also warranted as there are no counterarguments for $\langle \emptyset, h \rangle$. Based on the notion of warrant, the answer behaviour of a DeLP-interpreter can be defined as follows.

**Definition 12** (Answers to queries)**.** The answer of a DeLP-interpreter to a query $h$ is defined as 1.) YES, iff $h$ is warranted, 2.) NO, iff $\overline{h}$ is warranted, 3.) UNDECIDED, iff neither $h$ nor $\overline{h}$ are warranted and 4.) UNKNOWN, iff $h \notin \mathfrak{L}$.

## 3. Using Defeasible Logic Programming For a Distributed Environment

In an argumentation-based multi agent system (ArgMAS) several agents argue with each other about the truth value of a given logical sentence. In contrast to general DeLP the agents do not have knowledge about the beliefs of other agents and only react on their arguments. So as an ArgMAS consists of several components, the belief of an ArgMAS is divided among the components. Some belief can be seen as strict knowledge, that should be shared among all agents

of the system, e. g. knowledge about the current law or general facts as "every penguin is a bird". But every agent possesses also some individual beliefs like preferences or personal opinions, e. g. "if $X$ is a gardener, then $X$ is rather the murderer than anyone else". Thus the belief in an ArgMAS is divided into a *global belief base* and several *local belief bases*. An important constraint on belief bases is consistency. As the global belief base represents strict knowledge, it should be consistent in itself. Furthermore every local belief base should be consistent with the global belief base, as every agent's belief should not contradict with common knowledge, e. g. an agent should not argue that *john* is the murderer, if "*john* has an alibi" is strict knowledge. But as the opinions of different agents can differ, the union of all local belief bases and the global belief base can be inconsistent. The formal definition of belief bases will be given below.

### *3.1. Overview*

An ArgMAS takes a literal as a query, generates arguments with an internal deliberation mechanism and then returns a statement about acceptance or disproval.

In order to make a set of agents interact in a cooperative manner a coordination mechanism is needed. We use the centralized approach of *organisational structuring* [6] to get the external communications of the systems separated from the internal deliberation and coordination. A special agent, called *moderator*, will be used as an interface of the system to the outside world and as contact for queries. Furthermore the moderator coordinates the argumentation process between the other agents und finally analyzes the resulting argumentation structures to come up with an answer to the given query. Figure 1 shows a pattern of the message transfer in an ArgMAS and the special role of the moderator. The global belief base consists of the common knowledge of the whole system.

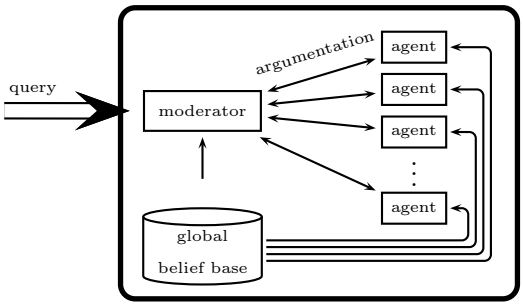


**Figure 1.** An argumentation-based multi agent system (ArgMAS)

The moderator of an ArgMAS must accept literals as queries from the outside world and be able to return answers. Furthermore he must send and receive internal queries to and from other agents and analyze the resulting dialectical trees of the actual argumentation process. Figure 2 shows the components of a moderator.

The actual argumentation process is done by the agents. They generate arguments on the basis of the global and their particular local belief bases and react

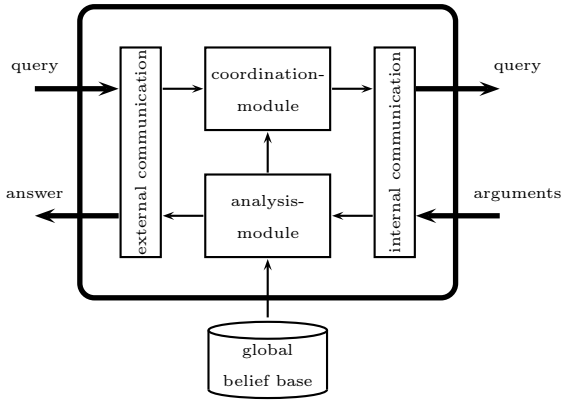


**Figure 2.** The internal components of a moderator

on arguments of other agents with counterarguments. An agent must be capable of inferring new beliefs by using the defeasible rules of his local belief base and the strict rules of the global belief base. Based on these inferences he generates suitable counterarguments to arguments from other agents. As the proposed system is centralized an agent only has to communicate with the moderator and the latter arbitrates between the individual agents. Figure 3 shows the components of an agent capable of argumentation. Notice that every agent has access to the global belief base and also has a local belief base.

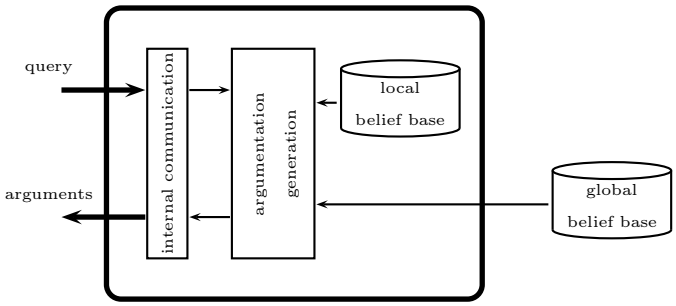


**Figure 3.** An argumentation-capable agent

In the next subsection we will formalize this argumentation process, the moderator and the agents.

### *3.2. Formalization*

As stated above the belief of an ArgMAS is divided into global and local belief bases. While the global belief base contains strict knowledge and therefore is built of facts and strict rules, the local belief bases are assumed to comprise only defeasible rules.

**Definition 13** (Belief bases)**.** A *global belief base* $\Pi$ is a non-contradictory set of strict rules and facts. A set of defeasible rules $\Delta$ is called a *local belief base relative to* $\Pi$, if $\Pi$ is a global belief base and $\Pi \cup \Delta$ is non-contradictory.

Besides the data structures for knowledge representation the components of an ArgMAS consist of functions to realize argumentation generation and analysis. The moderator must be capable of checking argumentation sequences for acceptance and evaluation of dialectical trees. Therefore in this section the necessary functions will be formally specified.

As in Section 2 the symbol $\mathfrak{L}$ denotes the set of all literals that may appear in a belief base. Furthermore let $\mathcal{R}$ be the set of all defeasible rules, that can be constructed with literals from $\mathfrak{L}$, $\Omega$ the set of all possible arguments that can be built using rules from $\mathcal{R}$ and conclusions from $\mathfrak{L}$, $\Sigma$ the set of all sequences of arguments from $\Omega$ and $\Upsilon$ the set of all dialectical trees of arguments from $\Omega$.

We start with formal description of the functional components of a moderator.

**Definition 14** (Analysis function)**.** An *analysis function* $\chi$ is a function $\chi : \Upsilon \rightarrow \{0, 1\}$, such that for every dialectical tree $\upsilon \in \Upsilon$ it holds $\chi(\upsilon) = 1$ iff the root argument of $\upsilon$ is undefeated.

The definition of an analysis function is independent of the definition of dialectical trees.

**Example 5.** Let $\upsilon \in \Upsilon$ be a dialectical tree according to Definition 10 and let $\upsilon^*$ be the corresponding marked dialectical tree. Then the analysis function $\chi_{\mathsf{D}}$ is defined as $\chi_{\mathsf{D}}(\upsilon) = 1$ iff the root of $\upsilon^*$ "undefeated" and $\chi_{\mathsf{D}}(\upsilon) = 0$ otherwise.

An acceptance function tests a given argument sequence for acceptance.

**Definition 15** (Acceptance function)**.** An *acceptance function* $\eta$ is a function $\eta : \Sigma \rightarrow \{0, 1\}$, such that for every argument sequence $\Lambda \in \Sigma$ it holds $\eta(\Lambda) = 1$ iff $\Lambda$ is accepted.

**Example 6.** Let $\succ$ be a preference relation among arguments. Then the acceptance function $\eta_{\mathsf{D},\succ}$ is defined as $\eta_{\mathsf{D},\succ}(\Lambda) = 1$ iff $\Lambda$ is acceptable with respect to $\succ$ and $\eta_{\mathsf{D},\succ}(\Lambda) = 0$ otherwise.

Let $\mathfrak{P}(S)$ denote the power set of $S$.

**Definition 16** (Decision function)**.** A *decision function* $\mu$ is a function $\mu : \mathfrak{P}(\Upsilon) \rightarrow$ {YES,NO,UNDECIDED,UNKNOWN}.

A decision function $\mu$ maps a set of dialectical trees to the suitable answer of the system. It is sufficient to define $\mu$ only on sets of dialectical trees with root arguments for or against the same atom as no other dialectical tree can be generated in an argumentation process for this particular atom.

**Example 7.** Let the moderator decide the answer to a query $p$ on the basis of Definition 12. Let $Q \subseteq \Upsilon$ such that all root arguments of dialectical trees in $Q$ are arguments for $p$ or for $\overline{p}$, then the decision function $\mu_{\mathsf{D}}$ is defined as

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

**Definition 17** (Moderator)**.** A *moderator* is a tuple $(\mu, \chi, \eta)$ with a decision function $\mu$, an analysis function $\chi$ and an acceptance function $\eta$. A moderator $(\mu, \chi, \eta)$ is called a DeLP-*moderator*, if $\mu = \mu_{\mathsf{D}}$, $\chi = \chi_{\mathsf{D}}$ and $\eta = \eta_{\mathsf{D},\succ}$ for a given preference relation $\succ$ among arguments.

An agent of an ArgMAS has to provide two functions: On the one hand he must be capable of generating initial arguments based on given literals and on the other hand he must be capable of generating counterarguments to arguments given by other agents.

**Definition 18** (Root argument function)**.** Let $\Pi$ be a global belief base and let $\Delta$ be a local belief base. A *root argument function* $\varphi_{\Pi,\Delta}$ relative to $\Pi$ and $\Delta$ is a function $\varphi_{\Pi,\Delta} : \mathfrak{L} \to \mathfrak{P}(\Omega)$, such that for every literal $h \in \mathfrak{L}$ the set $\varphi_{\Pi,\Delta}(h)$ is a set of arguments for $h$ or for $\overline{h}$ from $\Pi$ and $\Delta$.

To guarantee autonomy in argument generation of an agent, every agent has an own acceptance function $\eta$ to test his generated arguments on acceptance in the current argument sequence. On the basis of $\eta$ the counterargument function is defined as follows

**Definition 19** (Counterargument function)**.** Let $\Pi$ be a global belief base, let $\Delta$ be a local belief base, and let $\eta$ be an acceptance function. A *counterargument function* $\psi_{\Pi,\Delta}$ relative to $\Pi$ and $\Delta$ is a function $\psi_{\Pi,\Delta} : \Sigma \to \mathfrak{P}(\Omega)$, such that for every argumentation sequence $\Lambda \in \Sigma$ the set $\psi_{\Pi,\Delta}(\Lambda)$ is a set of attacks from $\Pi$ and $\Delta$ on the last argument of $\Lambda$ from $\Omega$ and for every $\langle \mathcal{B}, h \rangle \in \psi_{\Pi,\Delta}(\Lambda)$ it holds that $\eta(\Lambda + \langle \mathcal{B}, h \rangle) = 1$.

An agent of an ArgMAS is then defined as:

**Definition 20** (Agent)**.** Let $\Pi$ be a global belief base. An *agent* relative to $\Pi$ is a tuple $(\Delta, \varphi_{\Pi,\Delta}, \psi_{\Pi,\Delta}, \eta)$ with a local belief base $\Delta$ relative to $\Pi$, a root argument function $\varphi_{\Pi,\Delta}$, a counterargument function $\psi_{\Pi,\Delta}$ and an acceptance function $\eta$.

In the following we omit the subscripts $\Pi$ and $\Delta$ for root argument and counterargument functions when they are clear from context.

Putting things together, we define an argumentation-based multi agent system to consist of one moderator, a global belief base and a set of agents:

**Definition 21** (Argumentation-based multi agent system)**.** An *argumentation-based multi agent system* (ArgMAS) is a tuple $(M, \Pi, \{A_1, \ldots, A_n\})$ with a moderator $M$, a global belief base $\Pi$ and agents $A_1, \ldots, A_n$ relative to $\Pi$.

We now develop a functional description of the actual argumentation process to determine the answer to a query $h \in \mathfrak{L}$.

**Definition 22** (Argumentation product)**.** Let $h \in \mathfrak{L}$ be a query and $T = (M, \Pi, \{A_1, \ldots, A_n\})$ an ArgMAS with $M = (\mu, \chi, \eta)$ and $A_i = (\Delta_i, \varphi_i, \psi_i, \eta_i)$ for $1 \leq i \leq n$. A dialectical tree $\upsilon$ is called *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 $\upsilon$ 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 $\upsilon$ and the set $K$ of child nodes of $\langle \mathcal{A}_n, h_n \rangle$ it holds $K = \{\langle \mathcal{B}, h' \rangle | \langle \mathcal{B}, h' \rangle \in \psi_1(\Lambda) \cup \ldots \cup \psi_n(\Lambda)$ and $\eta(\Lambda + \langle \mathcal{B}, h' \rangle) = 1\}$ ($K$ is the set of all acceptable attacks on $\Lambda$).

The answer behaviour of an ArgMAS is based on the argumentation products and the decision function of the moderator:

**Definition 23** (Answer behaviour)**.** Let $h \in \mathfrak{L}$ be a query, $T = (M, \Delta, \{A_1, \ldots, A_n\})$ an ArgMAS with $M = (\mu, \chi, \eta)$ and let $\{\upsilon_1, \ldots, \upsilon_n\}$ be the set of all argumentation products of $T$ and $h$. The answer $A(T, h)$ of $T$ on the query $h$ is $A(T, h) = \mu(\{\upsilon_1, \ldots, \upsilon_n\})$.

The possible answers of an ArgMAS to a query are therefore YES, NO, UNDECIDED and UNKNOWN as it is for general DeLP.

## 4. Comparison with general Defeasible Logic Programming

In this section we compare the distributed framework for defeasible argumentation with general DeLP. As the definition of the components of an ArgMAS are based upon DeLP the comparison is realized via translation of an DeLP-program into an ArgMAS. We will show, that the answer behaviour of the original system and its translated counterpart will remain the same in most cases. All proofs of the following theorems can be found in an extended version of this paper [10].

For this section, the acceptance functions of all agents are identical with the acceptance function of the moderator, as this is also the situation in ordinary DeLP. Furthermore the root argument and counterargument functions of all agents are *maximal*, i. e. these functions always return the maximal set of possible arguments with respect to their beliefs.

To translate a DeLP-program $\mathcal{P}$ into an ArgMAS and maintain consistent belief bases, the set of defeasible rules of $\mathcal{P}$ have to be appropriatly divided among several agents. To sustain identical answer behaviours a very naive translation will suffice.

**Definition 24** (Argument-based rule-division)**.** Let $\mathcal{P}$ be a *de.l.p.* and $Q$ the set of all arguments of $\mathcal{P}$. Then the *argument-based rule-division* $AD(\mathcal{P})$ of $\mathcal{P}$ is defined by $AD(\mathcal{P}) = \{\mathcal{A} | \langle \mathcal{A}, h \rangle \in Q\}$.

For every possible argument $\langle \mathcal{A}, h \rangle$ there will be one agent with a local belief base $\mathcal{A}$. As arguments are consistent by definition, every agent has a consistent belief base. Furthermore no argument gets lost by translation of a DeLP into an ArgMAS.

**Definition 25** ($\mathcal{P}$-Induced ArgMAS)**.** Let $AD(\mathcal{P}) = \{\Delta_1, \ldots, \Delta_n\}$ the argument-based rule-devision of a *de.l.p.* $\mathcal{P} = (\Pi, \Delta)$. Let $A_1, \ldots, A_n$ be agents with local belief bases $\Delta_1, \ldots, \Delta_n$ respectively and let $M$ be a DeLP-moderator. Then $T = (M, \Pi, \{A_1, \ldots, A_n\})$ is the $\mathcal{P}$*-induced* ArgMAS.

As the local belief bases of the agents $A_1, \ldots, A_n$ are consistent by construction, every $\mathcal{P}$-induced ArgMAS is indeed an ArgMAS according to Definition 21. Furthermore the answer behaviour of the $\mathcal{P}$-induced ArgMAS is identical with the answer behaviour of $\mathcal{P}$ given identical preference relations for arguments.

**Theorem 1.** *Let* $\mathcal{P} = (\Pi, \Delta)$ *be a* de.l.p. *und let* $T = (M, \Pi, \{A_1, \ldots, A_n\})$ *be the* $\mathcal{P}$*-induced* ArgMAS*. Let* $h$ *be a query to* $\mathcal{P}$ *and* $A$ *the answer of* $\mathcal{P}$*, e. g.* $A \in \{$YES,NO,UNDECIDED,UNKNOWN$\}$*. Then* $A$ *is also the answer of* $T$ *to* $h$*.*

So every *de.l.p.* can be transformed into an ArgMAS while preserving its answer behaviour. The converse naive translation of an ArgMAS into a *de.l.p.* is not so easily possible without some restrictions.

**Definition 26** ($T$-induced *de.l.p.*)**.** Let $T = (M, \Pi, \{A_1, \ldots, A_n\})$ be an ArgMAS with a DeLP-moderator $M$ and let $\Delta_1, \ldots, \Delta_n$ be the local belief base of agents $A_1, \ldots, A_n$. Then $\mathcal{P} = (\Pi, \Delta_1 \cup \ldots \cup \Delta_n)$ is called the $T$*-induced* de.l.p..

For arbitrary local belief bases $\Delta_1, \ldots, \Delta_n$ the answer behaviour of $T$ and the $T$-induced *de.l.p.* is not necessarily the same as the following example shows.

**Example 8.** Let $T = (M, \Pi, \{A_1, A_2\})$ be an ArgMAS and let $\Delta_1$ and $\Delta_2$ be the local belief bases of $A_1$ and $A_2$ with $\Delta_1 = \{(b \prec a), (b \prec a, c)\}$ and $\Delta_2 = \{(\neg b \prec a), (c \prec d)\}$ and let furthermore $\Pi = \{a, d\}$. Given the query $b$, $T$ yields two argumentation products $[\langle\{(b \prec a)\}, b\rangle, \langle\{(\neg b \prec a)\}, \neg b\rangle]$ and $[\langle\{(\neg b \prec a)\}, \neg b\rangle, \langle\{(b \prec a)\}, b\rangle]$. As the roots of both argumentation products will be marked "defeated", the answer of $T$ on $b$ is UNDECIDED.

The $T$-induced *de.l.p.* $\mathcal{P} = (\Pi', \Delta')$ is given by $\Pi' = \{a, d\}$ and $\Delta = \{(b \prec a), (b \prec a, c), (\neg b \prec a), (c \prec d)\}$ and yields on the query $b$ among others the argumentation product $[\langle\{(b \prec a)\}, b\rangle, \langle\{(\neg b \prec a)\}, \neg b\rangle, \langle\{(b \prec a, c), (c \prec d)\}, b\rangle]$. As there the root will be marked with "undefeated", the answer of $\mathcal{P}$ on $b$ is YES.

The reason for the different answer behaviour of $T$ and $\mathcal{P}$ in Example 8 is the "misplaced" rule $c \prec d$, which can not be used for any argument on the query $b$ by $A_2$. By forbidding "misplaced" rules in an ArgMAS, a translation into a *de.l.p.* with protection of answer behaviour is possible. This yields the notion of a well-formed ArgMAS.

**Definition 27** (Well-formed ArgMAS)**.** Let $T = (M, \Pi, \{A_1, \ldots, A_n\})$ be an ArgMAS and let $\Delta_1, \ldots, \Delta_n$ the local belief bases of agents $A_1, \ldots, A_n$. Let furthermore $\mathcal{P}$ be the $T$-induced *de.l.p.*. $T$ is called a *well-formed* ArgMAS iff for every argument $\langle \mathcal{A}, h\rangle$ in $\mathcal{P}$ there exist an $i$ $(1 \leq i \leq n)$ with $\mathcal{A} \subseteq \Delta_i$.

**Theorem 2.** *Let* $T = (M, \Pi, \{A_1, \ldots, A_n\})$ *be a well-formed* ArgMAS *and let* $\Delta_1, \ldots, \Delta_n$ *be the local belief bases of agents* $A_1, \ldots, A_n$*. Let furthermore* $\mathcal{P}$ *be*

*the $T$-induced* de.l.p.. *If $h$ is a query and $A$ the answer of $T$ to $h$, then $A$ is also the answer of $\mathcal{P}$ to $h$.*

As every *de.l.p.* can be translated in an ArgMAS without losing information, distributed defeasible argumentation as proposed in this paper can be seen as a generalization of ordinary defeasible argumentation.

## 5. Remarks and Conclusion

The examination of distributed argumentation leads to new insights into logic-based argumentation in AI and discloses new application areas. The approach of distributed argumentation proposed in this paper distinguishes explicitly between several entities with different opinions on a structural basis. The proposed framework was implemented for a diploma thesis and a complex legal dispute was realised to illustrate the concept [9]. In that example two agents took the roles of accuser and defender, respectively, and argue about a legal claim. Parts of german law were translated to support the arguments and counterarguments of the two agents.

As the results in Section 4 show the proposed framework can subsume general defeasible logic programming and therefore is compatible with the existing theory on DeLP. As part of our ongoing work we plan to generalize the proposed system with the use of abstract argumentation frameworks [4] and investigate relationships of distributed argumentation using DeLP with game theory [3].

**Acknowledgments** The authors thank the reviewers for their helpful comments to improve the original version of this paper.

## References

[1] Leila Amgoud, Yannis Dimopolous, and Pavlos Moraitis. A unified and general framework for argumentation-based negotiation. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agents Systems, AAMAS'2007*, May 2007.

[2] T.J.M. Bench-Capon. Persuasion in practical argument using value based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[3] Laura A. Cecchi, Pablo R. Fillottrani, and Guillermo R. Simari. On the complexity of DeLP through game semantics. In J. Dix and A. Hunter, editors, *Proc. 11th Intl. Workshop on Nonmonotonic Reasoning (NMR 2006)*, pages 386–394, Windermere, UK, 2006.

[4] Phan M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AI Journal*, 77(2):321–358, 1995.

[5] A. García and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138, 2002.

[6] H. S. Nwana, L. C. Lee, and N. R. Jennings. Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4):79–88, 1996.

[7] Simon Parsons, Carles Sierra, and Nick Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[8] F. Stolzenburg, A. García, C. Chesñevar, and G. Simari. Computing generalized specificity. *Journal of Non-Classical Logics*, 13(1):87–113, 2003.

[9] M. Thimm. *Verteilte logikbasierte Argumentation: Konzeption, Implementierung und Anwendung im Rechtswesen*. VDM Verlag Dr. Müller, 2008.

[10] M. Thimm and G. Kern-Isberner. A distributed argumentation framework using defeasible logic programming (extended version). Technical report, TU Dortmund, 2008.