An Abstract Presentation of Dialectical Explanations in Defeasible Argumentation

Alejandro J. García, Carlos I. Chesñevar, Nicolás D. Rotstein, and Guillermo R. Simari

Artificial Intelligence Research Group
Department of Computer Science and Engineering
Universidad Nacional del Sur, Av.Alem 1253, (8000) Bahía Blanca, ARGENTINA
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
e-mail: {ajg, cic, ndr, grs}@cs.uns.edu.ar

Abstract. Abstract argumentation frameworks have played a major role as a way of understanding argument-based inference, resulting in different argument-based proof procedures. We will provide an abstract characterization of the warrant construction in the context of Skeptical Argumentation Frameworks. Often in the literature an argument is regarded as an explanation as well as a form of support for a claim, and this argument is evaluated to decide if the claim is accepted. The concept of explanation has received attention from different areas in Artificial Intelligence, particulary in the Knowledge-Based Systems community. Only a few of them consider explanations in relation with argument systems. In this paper, we propose a type of explanation that attempts to fill this gap providing a perspective from the point of view of argumentation systems.

1 Introduction and Motivations

Lately, interest in argumentation has expanded at increasing pace, driven in part by theoretical advances but also by successful demonstrations of a substantial number of practical applications, such as multiagent systems [17, 1], legal reasoning [18], knowledge engineering [4], and e-government [2], among many others. In this context, abstract argumentation frameworks [9] have played a major role as a way of understanding argument-based inference, resulting in different argument-based semantics. The final goal of such semantics is to characterize which are the rationally justified (or *warranted*) beliefs associated with a given set of arguments.

Dialectical analysis in argumentation involves the exploration of an *argument search space* in order to provide a proof-theoretic characterization of an argument-based semantics. Dialectical proof procedures provide the mechanism for performing computations of warranted arguments, traversing this argument search space by generating tree-like structures (called argument trees [3] or dialectical trees [11, 7] in the literature). We will provide an abstract characterization of the warrant construction in the context of *Skeptical Argumentation Frameworks*.

From another point of view, often in the literature an argument is regarded as an explanation for a claim that is represented by a literal. That is, the claim which is being explained is put under discussion, and only after evaluating its support it will be accepted or not. The role of explanations has received attention from several areas of

Artificial Intelligence –especially in the expert systems community [15, 20, 12]. A few of them consider explanations in relation with argument systems [16]. In belief revision, the role of explanations has also been studied [10]: new knowledge is accompanied by an explanation, which is used (when needed) to resolve inconsistency with the agent's current beliefs. The piece of knowledge having the "best" explanation is the one that prevails, and is accepted as a new belief.

We will focus our discussion on those explanations that give the necessary information to understand the warrant status of a literal. Since we consider only skeptical argumentation systems based on a dialectical proof procedure, we study *dialectical explanations* (from now on, δ-Explanations). Although we consider arguments as an explanation for a literal, we are interested in obtaining the complete set of dialectical trees that justify the warrant status of that literal. We show how δ-Explanations can be a useful tool to comprehend and analyze the interactions among arguments, and for aiding in the encoding and debugging of the underlying knowledge base. Several examples, generated with an implemented system that returns, for a given query, both the answer and the associated δ-Explanation, are given throughout the paper.

An interesting review about explanations in heuristic expert systems is given in [15], which offers the following definition: "*...explaining* consists in *exposing something* in such a way that it is *understandable* for the receiver of the explanation –so that he/she improves his/her knowledge about the object of the explanation– and *satisfactory* in that it meets the receiver's expectations." In our approach, we *explain* through *exposing* the whole set of dialectical trees related to the queried literal. This information is *understandable* from the receiver's point-of-view, because all the arguments built, their statuses (*i.e.*, defeated/undefeated), and their interrelations are explicitly shown. This type of information would be *satisfactory* for the receiver, because it contains all the elements at stake in the dialectical analysis that supports the answer.

An empirical analysis about the impact of different types of explanations in the context of expert systems is given in [20] which offers a typology that includes: 1) *trace:* a record of the inferential steps that led to the conclusion; 2) *justification:* an explicit description of the rationale behind each inferential step; and 3) *strategy:* a high-level goal structure determining the problem-solving strategy used. In this typology, the authors claim that their empirical analysis have shown that the most useful type of explanation is "justification". Our δ-Explanations match both the "justification" and the "strategy" types. That is, δ-Explanations give not only the strategy used by the system to achieve the conclusion, but also the rationale behind each argument supporting that conclusion as it is clearly stated in the corresponding dialectical tree.

We agree with [16], in that "*argumentation and explanation facilities in knowledge-based systems should be investigated in conjunction*". Therefore, we propose a type of explanation that attempts to fill the gap in the area of explanations in argument systems. Our approach is to provide a higher-level explanation in a way that the whole context of a query can be revealed. The examples given will stress this point.

The rest of this paper is structured as follows. Next, we will present the basic ideas of an abstract argumentation framework with dialectical constraints, which includes several concepts common to most argument-based formalisms. Then, we will present an abstract characterization of explanation along with a concrete reification based on Defeasible Logic Programming (DeLP).

2 An Abstract Framework with Dialectical Constraints

Abstract argumentation frameworks [9, 13] are formalisms for modelling defeasible argumentation [19, 5] in which some components remain unspecified. In this paper we are concerned with the study of warrant computation in argumentation systems, with focus on skeptical semantics for argumentation. As a basis for our analysis we will use an abstract argumentation framework (following Dung's seminal approach to abstract argumentation [9]) enriched with the notion of *dialectical constraint*, which will allow us to model distinguished sequences of arguments. The resulting, extended framework will be called an *argumentation theory*.

Definition 1 (Argumentation framework). [9] *An argumentation framework Φ is a pair $\langle \mathfrak{Args}, \boldsymbol{R} \rangle$, where $\mathfrak{Args}$ is a finite set of arguments and $\boldsymbol{R}$ is a binary relation between arguments such that $\boldsymbol{R} \subseteq \mathfrak{Args} \times \mathfrak{Args}$. The notation $(\mathcal{A}, \mathcal{B}) \in \boldsymbol{R}$ (or equivalently $\mathcal{A}\, \boldsymbol{R}\, \mathcal{B}$) means that $\mathcal{A}$* attacks *$\mathcal{B}$.*

Given an argumentation framework $\Phi = \langle \mathfrak{Args}, \boldsymbol{R} \rangle$, we will write $\mathfrak{Lines}_{\Phi}$ to denote the set of all the singleton sequences $[\mathcal{A}]$ with $\mathcal{A} \in \mathfrak{Args}$ and all possible finite sequences of arguments $[\mathcal{A}_0, \ldots, \mathcal{A}_k]$, with $k \geq 1$, such that for any pair of arguments $\mathcal{A}_i$, $\mathcal{A}_{i+1}$ it holds that $\mathcal{A}_{i+1}\, \boldsymbol{R}\, \mathcal{A}_i$, for $i = 0$ to k. Argumentation lines define a domain onto which different constraints can be defined. As such constraints are related to sequences which resemble an argumentation dialogue between two parties, we call them *dialectical constraints*. Formally:

Definition 2 (Dialectical Constraint). *Let $\Phi = \langle \mathfrak{Args}, \boldsymbol{R} \rangle$ be an argumentation framework. A* dialectical constraint $\mathbf{C}$ in the context of *Φ is any function $\mathbf{C} : \mathfrak{Lines}_{\Phi} \rightarrow \{True, False\}$. A given argument sequence $\lambda \in \mathfrak{Lines}_{\Phi}$ satisfies $\mathbf{C}$ in Φ when $\mathbf{C}(\lambda) = True$.*

An argumentation theory is defined by combining an argumentation framework with a particular set of dialectical constraints. Formally:

Definition 3 (Argumentation Theory). *An* argumentation theory *T (or just* theory*) is a pair $(\Phi, \mathbf{DC})$, where Φ is an argumentation framework, and $\mathbf{DC} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$ is a finite (possibly empty) set of* dialectical constraints.

Given a theory $T = (\Phi, \mathbf{DC})$, the intended role of $\mathbf{DC}$ is to avoid *fallacious* reasoning by imposing appropriate constraints on argumentation lines to be considered rationally *acceptable*. Such constraints are usually defined on disallowing certain moves which might lead to fallacious situations. Typical constraints to be found in $\mathbf{DC}$ are *non-circularity* (repeating the same argument twice in an argumentation line is forbidden), *commitment* (parties cannot contradict themselves when advancing arguments), etc. It must be noted that a full formalization for dialectical constraints is outside the scope of this work. We do not claim to be able to identify every one of such constraints either, as they may vary from one particular argumentation framework to another; that is the reason why $\mathbf{DC}$ is included as a parameter in T.[1]

[1] In this respect a similar approach is adopted in [14], where different characterizations of constraints give rise to different logic programming semantics.

2.1 Argumentation Lines

As already discussed before, argument games provide a useful form to characterize proof procedures for argumentation logics.Such games model defeasible reasoning as a dispute between two parties (*Proponent* and *Opponent* of a claim), who exchange arguments and counterarguments, generating *dialogues*. A proposition Q is provably justified on the basis of a set of arguments if its proponent has a *winning strategy* for an argument supporting Q, i.e. every counterargument (defeater) advanced by the Opponent can be ultimately defeated by the Proponent. Dialogues in such argument games have been given different names (dialogue lines, argumentation lines, dispute lines, etc.). A discussion on such aspects of different logical models of argument can be found in [5, 19]. The abstract framework presented in this section is based on the results presented in [6] and [8].

Definition 4 (Argumentation Line). *Let T be an argumentation theory. An* argumentation line *λ in T is any finite sequence of arguments $[\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n]$ such that every $\mathcal{A}_i$ attacks $\mathcal{A}_{i-1}$, for $0 < i \leq n$. If $\mathcal{A}_0$ is the first element in λ, we will also say that λ is* rooted in $\mathcal{A}_0$. *We will also write $\mid \lambda \mid = n$ to denote that λ has n arguments; we will also say that the* length *of λ is n.*

Definition 5 (Initial Argumentation Segment). *Let T be an argumentation theory and let $\lambda = [\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n]$ be an argumentation line in T. Then $\lambda' = [\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k]$ will be called an* initial argumentation segment *in λ of length k, $k \leq n$, denoted $\lfloor \lambda \rfloor_k$. When $k < n$ we will say that λ' is a* proper *initial argumentation segment in λ. We will use the term* initial segment *to refer to initial argumentation segments when no confusion arises.*

Example 1. Consider a theory $T = (\Phi, \mathbf{DC})$, with $\mathbf{DC} = \emptyset$, where the set $\mathfrak{Args}$ is $\{\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$, and assume that the following relationships hold: $\mathcal{A}_1$ attacks $\mathcal{A}_0$, $\mathcal{A}_2$ attacks $\mathcal{A}_0$, $\mathcal{A}_3$ attacks $\mathcal{A}_0$, $\mathcal{A}_4$ attacks $\mathcal{A}_1$. Three different argumentation lines rooted in $\mathcal{A}_0$ can be obtained, namely: λ_1 = [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_4$], λ_2 = [$\mathcal{A}_0$, $\mathcal{A}_2$], λ_3 = [$\mathcal{A}_0$, $\mathcal{A}_3$]. In particular, $\lfloor \lambda_1 \rfloor_2$ = $[\mathcal{A}_0, \mathcal{A}_1]$ is an initial argumentation segment in λ_1.

Example 2. Consider a theory $T' = (\Phi, \mathbf{DC})$ where the set $\mathfrak{Args}$ is $\{\mathcal{A}_0, \mathcal{A}_1\}$, and assume that the following relationships hold: $\mathcal{A}_0$ attacks $\mathcal{A}_1$, and $\mathcal{A}_1$ attacks $\mathcal{A}_0$. An infinite number of argumentation lines rooted in $\mathcal{A}_0$ can be obtained (e.g. λ_1 = [$\mathcal{A}_0$], λ_2 = [$\mathcal{A}_0$, $\mathcal{A}_1$], λ_3 = [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_0$], λ_4 = [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_0$, $\mathcal{A}_1$], etc.).

Remark 1. Note that from Def. 4, given an argumentation line $[\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ every subsequence $[\mathcal{A}_i, \mathcal{A}_{i+1}, \ldots \mathcal{A}_{i+k}]$ with $0 \leq i \leq n - k$ is also an argumentation line. In particular, every initial argumentation segment is also an argumentation line.

Intuitively, an argumentation line λ is acceptable iff it satisfies every dialectical constraint of the theory it belongs to. Formally:

Definition 6. *Given an argumentation theory T = $(\Phi, \mathbf{DC})$, an argumentation line λ is* acceptable *wrt T iff λ satisfies every $c \in \mathbf{DC}$.*

In what follows, we will assume that the notion of acceptability imposed by dialectical constraints is such that if λ is acceptable wrt a theory $T = (\Phi, \mathbf{DC})$, then any subsequence of λ is also acceptable.

Assumption 1 *If λ is an acceptable argumentation line wrt a theory $T = (\Phi, \mathbf{DC})$, then any subsequence of λ is also acceptable wrt T.*

Example 3. Consider the theory T' in Example 2, and assume that $\mathbf{DC}$=$\{$ Repetition of arguments is not allowed $\}$. Then λ_1 and λ_2 are acceptable argumentation lines in T', whereas λ_3 and λ_4 are not.

Definition 7 (λ' extends λ). *Let T be an argumentation theory, and let λ and λ' be two argumentation lines in T. We will say that λ'* extends *λ in T iff λ = $\lfloor \lambda' \rfloor_k$, for some $k <\, \mid \lambda' \mid$, that is, λ' extends λ iff λ is a proper initial argumentation segment of λ'.*

Definition 8. *Let T be an argumentation theory, and let λ be an acceptable argumentation line in T. We will say that λ is* exhaustive *if there is no acceptable argumentation line λ' in T such that $\mid \lambda \mid < \mid \lambda' \mid$, and for some k, λ = $\lfloor \lambda' \rfloor_k$, that is, there is no λ' such that λ' extends λ. Non-exhaustive argumentation lines will be referred to as* partial *argumentation lines.*

Example 4. Consider the theory T presented in Example 1. Then λ_1, λ_2 and λ_3 are exhaustive argumentation lines whereas $\lfloor \lambda_1 \rfloor_2$ is a partial argumentation line. In the case of the theory T' in Example 2, the argumentation line λ_2 extends λ_1. Argumentation line λ_2 is exhaustive, as it cannot be further extended on the basis of T' with the dialectical constraint introduced in Example 3.

Definition 9. *Given a theory T, a set S = $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of argumentation lines rooted in a given argument $\mathcal{A}$, denoted $S_{\mathcal{A}}$, is called a* bundle set *wrt T iff there is no pair $\lambda_i, \lambda_j \in S_{\mathcal{A}}$ such that λ_i extends λ_j.*

Example 5. Consider the theory $T = (\Phi, \mathbf{DC})$ from Example 1, and the argumentation lines λ_1, λ_2, and λ_3. Then $S_{\mathcal{A}_0} = \{\lambda_1, \lambda_2, \lambda_3\}$ is a bundle set wrt T.

2.2 Dialectical Trees

A bundle set $S_{\mathcal{A}}$ is a set of argumentation lines rooted in a given argument $\mathcal{A}$. Such set can be thought of as a tree structure, where every line corresponds to a branch in the tree. Formally:

Definition 10 (Dialectical tree). *Let T be a theory, and let $\mathcal{A}$ be an argument in T, and let $S_{\mathcal{A}}$ = $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ be an acceptable set of argumentation lines rooted in $\mathcal{A}$. The* dialectical tree *rooted in $\mathcal{A}$* based on *$S_{\mathcal{A}}$ (denoted $\mathcal{T}_{\mathcal{A}}$) is a tree-like structure defined as follows:*

1. *The root node of $\mathcal{T}_{\mathcal{A}}$ is $\mathcal{A}$.*

2. *Let* F=$\{\mathsf{tail}(\lambda)$*, for every* $\lambda \in S_{\mathcal{A}}\}$*, and* H=$\{\mathsf{head}(\lambda)$*, for every* $\lambda \in F\}$.[2]
   If $H = \emptyset$ *then* $\mathcal{T}_{\mathcal{A}}$ *has no subtrees.*
   Otherwise, if $H = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$*, then for every* $\mathcal{B}_i \in H$*, we define*

$$\mathsf{getBundle}(\mathcal{B}_i) = \{\lambda \in F \mid \mathsf{head}(\lambda) = \mathcal{B}_i\}$$

   We put $\mathcal{T}_{\mathcal{B}_i}$ *as an immediate subtree of* $\mathcal{A}$*, where* $\mathcal{T}_{\mathcal{B}_i}$ *is a dialectical tree based on* getBundle$(\mathcal{B}_i)$.

We will write $\mathfrak{Tree}_{\mathcal{A}}$ *to denote the family of all possible dialectical trees based on* $\mathcal{A}$*. We will represent as* $\mathfrak{Tree}_{T}$ *the family of all possible dialectical trees in the theory* T.

Example 6. Consider the theory $T = (\Phi, \mathbf{DC})$ from Example 1, and the acceptable set $S_{\mathcal{A}_0}$ from Example 5. Fig. 1(a) shows the associated exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}_0}$.

The above definition shows how to build a dialectical tree from a bundle set of argumentation lines rooted in a given argument. It is important to note that the "shape" of the resulting tree will depend on the order in which the subtrees are attached. Each possible order will produce a tree with a different geometric configuration. All the differently conformed trees are nevertheless "equivalent" in the sense that they will contain exactly the same argumentation lines as branches from its root to its leaves. This observation is formalized by introducing the following relation which can be trivially shown to be an equivalence relation.

Definition 11. *Let* T *be a theory, and let* $\mathfrak{Tree}_{\mathcal{A}}$ *be the set of all possible dialectical trees rooted in an argument* $\mathcal{A}$ *in theory* T*. We will say that* $\mathcal{T}_{\mathcal{A}}$ *is equivalent to* $\mathcal{T}'_{\mathcal{A}}$*, denoted* $\mathcal{T}_{\mathcal{A}} \equiv_{\tau} \mathcal{T}'_{\mathcal{A}}$ *iff they are obtained from the the same bundle set* $S_{\mathcal{A}}$ *of argumentation lines rooted in* $\mathcal{A}$.

Given an argument $\mathcal{A}$, there is a one-to-one correspondence between a bundle set $S_{\mathcal{A}}$ of argumentation lines rooted in $\mathcal{A}$ and the corresponding equivalence class of dialectical trees that share the same bundle set as their origin (as specified in Def. 10). In fact, a dialectical tree $\mathcal{T}_{\mathcal{A}}$ based on $S_{\mathcal{A}}$ is just *an alternative way* of expressing the same information already present in $S_{\mathcal{A}}$. Each member of an equivalence class represents a different way in which a tree could be built. Each particular computational method used to generate the tree from the bundle set will produce one particular member on the equivalence class. In that manner, the equivalence relation will represent a tool for exploring the computational process of warrant and as we will see later, trees provide a powerful way of conceptualize the computation of warranted arguments. Next, we will define mappings which allow to re-formulate a bundle set $S_{\mathcal{A}}$ as a dialectical tree $\mathcal{T}_{\mathcal{A}}$ and viceversa.

Definition 12 (Mapping $\mathbb{T}$). *Let* T *be an argumentative theory, and let* $S_{\mathcal{A}}$ *be a bundle set of argumentation lines rooted in an argument* $\mathcal{A}$ *of* T*. We define the mapping*

$$\mathbb{T} : \wp(\mathfrak{Lines}_{\mathcal{A}}) \setminus \{\emptyset\} \rightarrow \overline{\mathfrak{Tree}_{\mathcal{A}}}$$

as $\mathbb{T}(S_{\mathcal{A}}) =_{\text{def}} \overline{\mathcal{T}_{\mathcal{A}}}$*, where* $\mathfrak{Lines}_{\mathcal{A}}$ *is the set of all argumentation lines rooted in* $\mathcal{A}$*,* $\overline{\mathfrak{Tree}_{\mathcal{A}}}$ *is the quotient set of* $\mathfrak{Tree}_{\mathcal{A}}$ *by* $\equiv_{\tau}$*, and* $\overline{\mathcal{T}_{\mathcal{A}}}$ *denotes the equivalence class of* $\mathcal{T}_{\mathcal{A}}$.

[2] The functions $\mathsf{head}(\Delta)$ and $\mathsf{tail}(\Delta)$ have the usual meaning in list processing, returning the first element in a list and the list formed by all elements except the first, respectively.

Proposition 1. *For any argument $\mathcal{A}$ in an argumentative theory T, the mapping $\mathbb{T}$ is a bijection.*[3]

As the mapping $\mathbb{T}$ is a bijection, so that we can define also the inverse mapping $\mathbb{S} =_{def} \mathbb{T}^{-1}$ which allow us to determine the acceptable set of argumentation lines corresponding to an arbitrary dialectical tree rooted in an argument $\mathcal{A}$. In what follows, we will use indistinctly a *set notation* (an acceptable bundle set of argumentation lines rooted in an argument $\mathcal{A}$) or a *tree notation* (a dialectical tree rooted in $\mathcal{A}$), as the former mappings $\mathbb{S}$ and $\mathbb{T}$ allow us to go from any of these notation to the other.

Proposition 2. *Let T be an argumentation theory, and let $S_{\mathcal{A}}$ be an acceptable bundle set of argumentation lines rooted in a given argument $\mathcal{A}$, $S_{\mathcal{A}} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Let $S'_{\mathcal{A}}$ = $\{\lambda'_1, \ldots, \lambda'_m\}$, $m \leq n$, be a set of initial argumentation segments, where every $\lambda'_i = \lfloor \lambda_i \rfloor_{k_i}$, for some $k_i \leq \mid \lambda_i \mid$, $i \leq m$. Let*

$$S'' = S'_{\mathcal{A}} \setminus \{\lambda \in S'_{\mathcal{A}} \mid \textit{ there exists } \lambda' \in S'_{\mathcal{A}} \textit{ and } \lambda' \textit{ extends } \lambda\}. \tag{1}$$

Then S'' is also an acceptable set of argumentation lines rooted in $\mathcal{A}$.

The following proposition shows that dialectical trees can be thought of as compositional structures, in the sense that any subtree $\mathcal{T}'_{\mathcal{A}}$ of a dialectical tree $\mathcal{T}_{\mathcal{A}}$ is also a dialectical tree.

Proposition 3. *Let T be a theory, and $\mathcal{T}_{\mathcal{A}}$ a dialectical tree in T. Then it holds that any subtree $\mathcal{T}'_{\mathcal{A}}$ in $\mathcal{T}_{\mathcal{A}}$ rooted in $\mathcal{A}$ is also a dialectical tree wrt T.*

2.3 Acceptable dialectical trees

The notion of acceptable argumentation line will be used to characterize acceptable dialectical trees, which will be fundamental as a basis for formalizing the computation of *warranted arguments* in our setting.

Definition 13 (Acceptable dialectical tree). *Let T be a theory, a dialectical tree $\mathcal{T}_{\mathcal{A}}$ in T is acceptable iff every argumentation line in the associated bundle set $\mathbb{S}(\overline{\mathcal{T}_{\mathcal{A}}})$ is acceptable. We will distinguish the subset $\mathfrak{ATree}_{\mathcal{A}}$ (resp. $\mathfrak{ATree}_T$) of all acceptable dialectical trees in $\mathfrak{Tree}_{\mathcal{A}}$ (resp. $\mathfrak{Tree}_T$).*

As acceptable dialectical trees are a subclass of dialectical trees, all the properties previously shown apply also to them. In the sequel, we will just write “dialectical trees” to refer to acceptable dialectical trees, unless stated otherwise.

Definition 14 (Exhaustive Dialectical tree). *A dialectical tree $\mathcal{T}_{\mathcal{A}}$ will be called* exhaustive *iff it is constructed from the set $S_{\mathcal{A}}$ of all possible exhaustive argumentation lines rooted in $\mathcal{A}$, otherwise $\mathcal{T}_{\mathcal{A}}$ will be called* partial.

Besides, the exhaustive dialectical tree for any argument $\mathcal{A}$ can be proven to be unique (up to an equivalence).

[3] Due to space constrains proofs will be omitted.

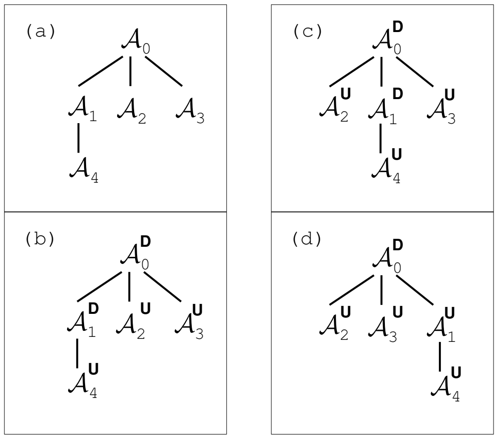

Fig. 1. (a) Exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}_0}$ for Example 6; (b) resulting tree after applying and-or marking; (c)–(d) two other exhaustive dialectical trees belonging to the equivalence class $\overline{\mathcal{T}_{\mathcal{A}_0}}$

Proposition 4. *Let T be a theory, for any argument $\mathcal{A}$ in T there is a unique exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}}$ in T (up to an equivalence wrt $\equiv_\tau$ as defined in Def. 11).*

Acceptable dialectical trees allow to determine whether the root node of the tree is to be accepted (ultimately *undefeated*) or rejected (ultimately *defeated*) as a rationally justified belief. A *marking function* provides a definition of such acceptance criterion. Formally:

Definition 15 (Marking criterion). *Let T be a theory. A marking criterion for T is a function* Mark : $\mathfrak{Tree}_T \rightarrow \{D, U\}$. *We will write* Mark$(\mathcal{T}_i) = U$ *(resp.* Mark$(\mathcal{T}_i) = D$*) to denote that the root node of $\mathcal{T}_i$ is marked as U-node (resp. D-node).*

Definition 16 (Warrant). *Let T be an argumentative theory and* Mark *a marking criterion for T. An argument $\mathcal{A}$ is a* warranted argument *(or just* warrant*) wrt a marking criterion* Mark *in T iff the exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}}$ is such that* Mark$(\mathcal{T}_{\mathcal{A}}) = U$. *We will denote a marked dialectical tree as $\mathcal{T}^*_{\mathcal{A}}$.*

3 Answers and δ-Explanations

An argument is a piece of reasoning that supports a claim Q from certain evidence. The tenability of this claim must be confirmed by analyzing other arguments for and against such claim. Next, we will define *queries*, *answers* and *explanations* in the abstract context introduced in the previous Section.

The dialectical process for warranting a claim involves finding the arguments that either support or interfere with that claim. These arguments are connected through the defeat relation and are organized in dialectical trees. Observe that given a claim there could exist in T different arguments that support it, and each argument will generate a different dialectical tree.

Definition 17 (T-Queries). *Let T be an argumentation theory. A* T-query Q *posed to the theory T will represent the process of finding out the existence, and the warranting status, of the posible arguments for Q and $\overline{Q}$.*[4]

We will show below that the returned answer for Q will be only the result of analyzing a set of dialectical trees that have been built and considered as to support this answer. Thus, to understand why a query has that particular answer, it is essential to consider which arguments have been considered and what connections exist among them.

It is important to notice that δ-Explanations are at the crux of an argumentation system whose proof procedure is based on the construction of dialectical trees. They present the reasoning carried out by the system, and they allow to visualize the support for the answer given. It is clear that without this information it will be very difficult to understand the returned answer.

Definition 18 (δ-Explanation). *Let T be an argumentation theory and let Q be a claim. Let $\mathcal{A}_0,\ldots,\mathcal{A}_n$ be all the arguments for Q from T, and $\mathcal{B}_0,\ldots,\mathcal{B}_m$ be all the arguments for $\overline{Q}$ from T. Then, the* explanation *for Q in T is the set of marked dialectical trees $\mathcal{E}_T(Q) = \{\mathcal{T}^*_{\mathcal{A}_0},\ldots,\mathcal{T}^*_{\mathcal{A}_n}\} \cup \{\mathcal{T}^*_{\mathcal{B}_0},\ldots,\mathcal{T}^*_{\mathcal{B}_m}\}$.*

Now it is possible to define T-answers in terms of the associated δ-Explanations.

Definition 19 (T-answer). *Given an argumentation theory T and a query Q, the answer for Q is:*

- YES, *if at least one tree in $\mathcal{E}_T(Q)$ warrants Q.*
- NO, *if at least one tree in $\mathcal{E}_T(Q)$ warrants $\overline{Q}$.*
- UNDECIDED, *if $\mathcal{E}_T(Q)$ is non empty, but no tree in $\mathcal{E}_T(Q)$ warrants Q nor $\overline{Q}$.*
- UNKNOWN, *if there is no argument for Q in T.*

Notice that if there is a dialectical that shows that an argument warrants Q then there is no argument that warrants $\overline{Q}$.

4 Answers and δ-Explanations in DELP: A Reification

Next, we will define *queries*, *answers* and *explanations* using the framework provided by DELP (see [11] for full details on DELP). Extending the abstract presentation above, we will introduce two types of queries: ground (called DELP-queries) and schematic. For both types of queries we will define explanations and a way to obtain the corresponding answer, that is: YES, NO, UNDECIDED or UNKNOWN.

Definition 20 (DELP-queries). *A* DELP-query *is a ground literal that* DELP *will try to warrant. A query with at least one variable will be called* schematic query *and will account for the set of* DELP*-queries that unify with the schematic one.*

[4] The notation $\overline{Q}$ is used to represent the complement of Q with respect to strong negation, *i.e.*, $\overline{a}$=$\sim a$ and $\overline{\sim a}$=a.

In DELP, δ-*Explanations* for answers will be the set of dialectical trees that have been explored to obtain a warrant for that query. The definition for a δ-Explanation for a DELP-query follows, whereas explanations for schematic queries will be introduced by the end of this Section. It is clear that without the information regarding the dialectical trees it will be very difficult to understand the returned answer. Next, we will introduce explanations for ground queries and we will generalize them for schematic queries.

Definition 21 (δ-Explanations for a DELP-query).
Let $\mathcal{P}$ be a DELP-program and Q a DELP-query. Let $\langle\mathcal{A}_0, Q\rangle$,...,$\langle\mathcal{A}_n, Q\rangle$ be all the arguments for Q from $\mathcal{P}$, and $\langle\mathcal{B}_0, \overline{Q}\rangle$,...,$\langle\mathcal{B}_m, \overline{Q}\rangle$ be all the arguments for $\overline{Q}$ from $\mathcal{P}$. Then, the explanation *for Q in $\mathcal{P}$ is the set of marked dialectical trees $\mathcal{E}_{\mathcal{P}}(Q)$ = $\{\mathcal{T}^*_{\langle\mathcal{A}_0, Q\rangle},\ldots,\mathcal{T}^*_{\langle\mathcal{A}_n, Q\rangle}\} \cup \{\mathcal{T}^*_{\langle\mathcal{B}_0, \overline{Q}\rangle},\ldots,\mathcal{T}^*_{\langle\mathcal{B}_m, \overline{Q}\rangle}\}$.*

Using these concepts we can define DELP-answers.

Definition 22 (DELP-answer). *Given a DELP-program $\mathcal{P}$ and a DELP-query Q, the answer for Q is:*

- YES, *if at least one tree in $\mathcal{E}_{\mathcal{P}}(Q)$ warrants Q.*
- NO, *if at least one tree in $\mathcal{E}_{\mathcal{P}}(Q)$ warrants $\overline{Q}$.*
- UNDECIDED, *if no tree in $\mathcal{E}_{\mathcal{P}}(Q)$ warrants Q nor $\overline{Q}$.*
- UNKNOWN, *if Q is not in the signature of $\mathcal{P}$.*

Example 7. Consider the DELP-program (Π_7, Δ_7) where:

$$\Pi_7 = \left\{\begin{array}{l} bird(X) \leftarrow chicken(X) \\ chicken(little) \\ chicken(tina) \\ scared(tina) \\ bird(rob) \end{array}\right\} \quad \Delta_7 = \left\{\begin{array}{l} flies(X) \prec bird(X) \\ flies(X) \prec chicken(X), scared(X) \\ {\sim}flies(X) \prec chicken(X) \end{array}\right\}$$

From the DELP-program (Π_7, Δ_7) the following arguments can be obtained (due to space restrictions *'tina'* will be abbreviated to *'t'* and *'flies(tina)'* to *'f'*): $\langle\mathcal{A}_1, f\rangle$ = $\langle\{flies(t) \prec bird(t)\}, flies(t)\rangle$, $\langle\mathcal{A}_2, {\sim}f\rangle$ = $\langle\{{\sim}flies(t) \prec chicken(t)\}, {\sim}flies(t)\rangle$, and $\langle\mathcal{A}_3, f\rangle$ = $\langle\{flies(t) \prec chicken(t), scared(t)\}, flies(t)\rangle$. The argument $\langle\mathcal{A}_2, {\sim}f\rangle$ defeats $\langle\mathcal{A}_1, f\rangle$, $\langle\mathcal{A}_3, f\rangle$ defeats $\langle\mathcal{A}_2, {\sim}f\rangle$, and *[*$\langle\mathcal{A}_1, f\rangle$, $\langle\mathcal{A}_2, {\sim}f\rangle$, $\langle\mathcal{A}_3, f\rangle$*]* is an acceptable argumentation line.

Figure 2 shows the δ-Explanation for the DELP-query '$flies(tina)$', where two dialectical trees for '$flies(tina)$' are marked "U". Therefore, '$flies(tina)$' is warranted and the answer is YES. Note that the δ-Explanation of Figure 2 is also an explanation for query '${\sim}flies(tina)$' whose answer is NO. Finally, observe that the answer for *'walks(tim)'* is UNKNOWN, because it is not in the program signature.

Remark 2. The explanation for complementary literals will always be the same, since it is composed by both the trees for the literal and the trees for its complement.

As we will show in the examples below, the semantics of the programs is *sensitive* to the addition or deletion of rules and facts. That is, a new fact added to a program

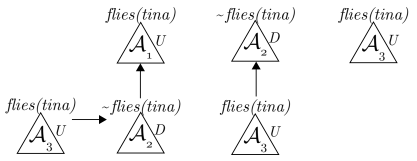

Fig. 2. δ-Explanation for $flies(tina)$

can have a big impact on the number of arguments that can be built from the modified program. Taking into account this characteristic and considering the many possible interactions among arguments via the defeat relation (that lead to the construction of different dialectical trees), δ-Explanations become essential for understanding the reasons that support an answer.

Example 8. Consider the DELP-program $(\varPi_8, \varDelta_8)$: $\varPi_8 = \{q, t\}, \varDelta_8 = \{(r \prec q), (\sim r \prec q, s), (r \prec s), (\sim r \prec t)\}$, where the following arguments can be built: $\langle \mathcal{R}_1, \sim r\rangle = \langle\{\sim r \prec t\}, \sim r\rangle$, and $\langle \mathcal{R}_2, r\rangle = \langle\{r \prec q\}, r\rangle$. From this program the answer for the query 'r' is UNDECIDED, and Figure 3 shows its δ-Explanation. Note that, although the literal 's' is in the program signature (in the body of a rule), there is no supporting argument for it. Therefore, the answer for query 's' is UNDECIDED, and the δ-Explanation is the empty set (*i.e.*, $\mathcal{E}_{(\varPi_8,\varDelta_8)}(s)$=$\emptyset$).

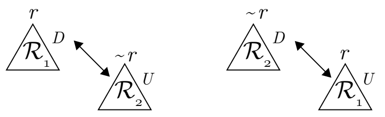

Fig. 3. δ-Explanation $\mathcal{E}_{(\varPi_8,\varDelta_8)}(r)$

Remark 3. DELP-queries with UNKNOWN answers always have an empty δ-Explanation. However, DELP-queries that have UNDECIDED answers may have empty or non-empty explanations. Finally, DELP-queries with YES or NO answers will always have a non-empty explanation.

Example 9. (Extends Ex. 8) In this example we see how the introduction of a single fact in $(\varPi_8, \varDelta_8)$ makes a significant difference in $\mathcal{E}_{(\varPi_8,\varDelta_8)}(r)$. Consider the DELP-program $(\varPi_8 \cup \{s\}, \varDelta_8)$ where the fact 's' is added to the program of Example 8. If we query for 'r' again, we get the answer NO with the δ-Explanation shown in Figure 4. Note that this δ-Explanation consists now of two more trees than the one in the previous example. This is so because there are two newly generated arguments: $\langle \mathcal{R}_3, r\rangle = \langle\{r \prec s\}, r\rangle$, and $\langle \mathcal{R}_4, \sim r\rangle = \langle\{\sim r \prec q, s\}, \sim r\rangle$

It is our contention that, in DELP, the answer for a query should be easily explained by presenting the user the associated dialectical trees. From this set of trees the answer

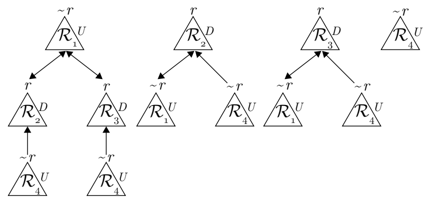

Fig. 4. δ-Explanation $\mathcal{E}_{(\Pi_8 \cup \{s\}, \Delta_8)}(r)$

becomes thoroughly justified, and the context of the query is revealed. The following examples have more elaborated DELP-programs and the δ-Explanations show that a defeater $\mathcal{D}$ for $\mathcal{A}$ may attack an inner point of $\mathcal{A}$.

Example 10. Consider the DELP-program (Π_{10}, Δ_{10}), where $\Pi_{10} = \{c, e, f\}$ and

$$\Delta_{10} = \left\{ \begin{array}{llllll} (a \prec b), & (b \prec c), & (\sim b \prec d), & (d \prec e), & (\sim d \prec f, e), & (\sim b \prec e), \\ (a \prec x), & (x \prec c), & (\sim x \prec e), & (a \prec h), & (h \prec f), & (\sim h \prec i) \end{array} \right\}$$

the following arguments can be built: $\langle \mathcal{A}_1, a \rangle = \langle \{(a \prec h), (h \prec f)\}, a \rangle$
$\langle \mathcal{B}_1, b \rangle = \langle \{b \prec c\}, b \rangle$ $\quad \langle \mathcal{B}_2, \sim b \rangle = \langle \{\sim b \prec e\}, \sim b \rangle$
$\langle \mathcal{D}_1, d \rangle = \langle \{d \prec e\}, d \rangle$ $\quad \langle \mathcal{D}_2, \sim d \rangle = \langle \{(\sim d \prec f, e)\}, \sim d \rangle$
$\langle \mathcal{X}_1, x \rangle = \langle \{x \prec c\}, x \rangle$ $\quad \langle \mathcal{X}_2, \sim x \rangle = \langle \{\sim x \prec e\}, \sim x \rangle$

From (Π_{10}, Δ_{10}) the answer for 'a' is YES, and the answer for '$\sim a$' is NO. As stated in Remark 2, although both queries have different answers, they both have the same δ-Explanation, which is depicted in Figure 5.

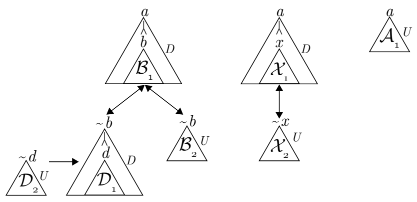

Fig. 5. δ-Explanation $\mathcal{E}_{(\Pi_{10}, \Delta_{10})}(a)$

From the DELP programmer point-of-view, δ-Explanations give a global idea of the interactions among arguments within the context of a query. This is an essential debugging tool when programming: if unexpected behavior arises, the programmer can check the given explanations to detect errors.

In the previous examples we have not shown an explanation associated with a query with an UNKNOWN answer, because this type of answers have an empty δ-Explanation.

In a similar manner, observe that queries that do not correspond to the intended domain of the program will return the answer UNKNOWN. This will capture errors like querying for *"fly"* instead of *"flies"*, or a query like *"penguin(X)"* in Example 7.

Now we will extend the notion of explanation to encompass *schematic queries*. A schematic query is a query that has at least one variable (see Definition 20), and hence it represents the set of DELP-queries that unify with it. We will extend the definition of δ-Explanation to include schematic queries. In the DELP-program of Example 7, the schematic query $flies(X)$ will refer to $flies(tina)$ and $flies(little)$.

Observe that there are actually infinite terms that unify with variable X. However, all queries with terms that are not in the program signature will produce an UNKNOWN answer and therefore an empty explanation. Thus, the set of instances of a schematic query that will be considered for generating an explanation will refer only to those instances of DELP-queries that contain constants from the program signature.

Definition 23 (Generalized δ-Explanation).
Let $\mathcal{P}$ be a DELP*-program and Q a schematic query. Let $\{Q_1, \ldots, Q_z\}$ be all the instances of Q so that their* DELP*-answer is different from* UNKNOWN*. Let $\mathcal{E}_{\mathcal{P}}(Q_i)$ be the δ-Explanation for the* DELP*-query Q_i ($1 \leq i \leq z$) from program $\mathcal{P}$. Then, the* generalized δ-Explanation *for Q in $\mathcal{P}$ is $\mathcal{E}_{\mathcal{P}}(Q)$ = $\{$ $\mathcal{E}_{\mathcal{P}}(Q_1)$, ..., $\mathcal{E}_{\mathcal{P}}(Q_z)\}$.*

Observe that a δ-Explanation (Definition 21) is a particular case of a Generalized δ-Explanation, where the set $\mathcal{E}_{\mathcal{P}}(Q)$ is a singleton.

Example 11. Consider again the DELP-program (Π_7, Δ_7), and suppose that we want to know if from this program it can be warranted that a certain individual does not fly. If we query for $\sim flies(X)$, the answer is YES, because there is a warranted instance: $\sim flies(little)$. The supporting argument is (*'little'* was abbreviated to *'l'*): $\langle \mathcal{B}_1, \sim flies(l)\rangle$ = $\langle\{\sim flies(l) \prec chicken(l)\}, \sim flies(l)\rangle$. The trees of the generalized explanation are shown in Figure 6. This explanation also shows that the other instance ($\sim flies(tina)$) is not warranted. Note that the answer for the schematic query $flies(X)$ is also YES, but with a different set of warranted instances: $flies(tina)$ and $flies(rob)$. The supporting argument for instance '$X = tina$' was already discussed, and the undefeated argument for instance '$X = rob$' is: $\langle \mathcal{C}_1, flies(rob)\rangle$ = $\langle\{flies(rob) \prec bird(rob)\}, flies(rob)\rangle$. The generalized δ-Explanation for $flies(X)$ is the same as the one for $\sim flies(X)$, depicted in Figure 6 (see Remark 2).

Definition 24 (DELP-answer for a schematic query). *Given a* DELP*-program $\mathcal{P}$ and a schematic query Q, the answer for Q is*

– YES, *if there exists an instance Q_i of Q such that at least one tree in $\mathcal{E}_{\mathcal{P}}(Q_i)$ warrants Q_i.*
– NO, *if there exists an instance Q_i of Q such that at least one tree in $\mathcal{E}_{\mathcal{P}}(Q_i)$ warrants $\overline{Q_i}$.*
– UNDECIDED, *if for every instance Q_i of Q that is in the signature of $\mathcal{P}$, there is no tree in $\mathcal{E}_{\mathcal{P}}(Q_i)$ that warrants Q_i nor $\overline{Q_i}$.*
– UNKNOWN, *if there is no instance Q_i of Q such that Q_i is in the signature of $\mathcal{P}$.*

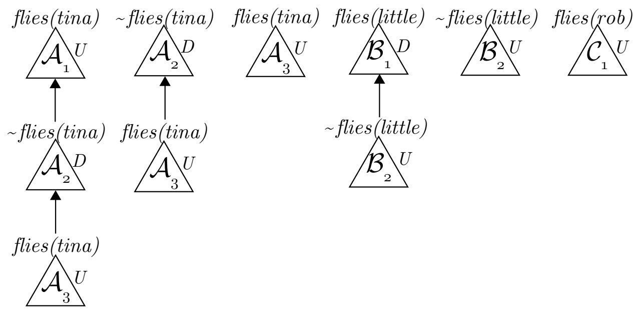

Fig. 6. Generalized δ-Explanation for '$\sim flies(X)$'

Observe that Definition 22 is a particular case of the previous definition, where there is a single instance of Q.

Example 12. Consider the following DeLP-program:

$$\Pi_{12} = \left\{ \begin{array}{l} adult(peter) \\ adult(annie) \\ unemployed(peter) \\ student(annie) \end{array} \right\} \Delta_{12} = \left\{ \begin{array}{l} has\_a\_car(X) \prec adult(X) \\ \sim has\_a\_car(X) \prec unemployed(X) \\ \sim has\_a\_car(X) \prec student(X) \end{array} \right\}$$

where the following arguments can be built ('has_a_car' was replaced by 'car', '$annie$' by 'a', and '$peter$' by 'p'): $\langle \mathcal{A}_1, car(a) \rangle = \langle \{car(a) \prec adult(a)\}, car(a) \rangle$,
$\langle \mathcal{A}_2, \sim car(a) \rangle = \langle \{\sim car(a) \prec student(a)\}, \sim car(a) \rangle$,
$\langle \mathcal{P}_1, car(p) \rangle = \langle \{car(p) \prec adult(p)\}, car(p) \rangle$, and
$\langle \mathcal{P}_2, \sim car(p) \rangle = \langle \{\sim car(p) \prec unemployed(p)\}, \sim car(p) \rangle$.
When querying for '$has\_a\_car(X)$', variable 'X' unifies with both '$annie$' and '$peter$'. Then, DeLP builds arguments for both instances: $\mathcal{A}_1$ and $\mathcal{A}_2$ for '$X = annie$', and $\mathcal{P}_1$ and $\mathcal{P}_2$ for '$X = peter$'. From Figure 7, it is clear that no argument is undefeated, *i.e.*, there is no tree that warrants '$has\_a\_car(X)$', for either of the two instances. Therefore, the answer is UNDECIDED, and the variable remains unbound.

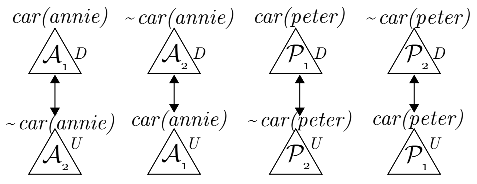

Fig. 7. Generalized δ-Explanation for '$has\_a\_car(X)$'

Schematic queries give us the possibility of asking more general questions than ground queries. Now we are not asking whether a certain piece of knowledge can be believed, but we are asking if there exists an instance of that piece of knowledge (related to an individual) that can be warranted in the system. This could lead to deeper reasoning as we may pose a query, gather the warranted instances and continue reasoning with those individuals.

The δ-Explanations system receives a DELP-program `P`, a query `Q`, and an argument comparison criterion `C`, and returns a δ-Explanation `EX` and the corresponding answer `ANS`. The system is described by the following algorithm in a Prolog-like notation:

```
d_Explanations(P,C,Q,EX,ANS):-
    warrants(P,Q,C,WSQ), complement(Q,NQ), warrants(P,NQ,C,WSNQ),
    get_trees(WSQ,WSNQ,EX),  get_answer(Q,WSQ,WSNQ,ANS).

get_answer(_,WSQ,WSNQ,yes):-WSQ \= [].
get_answer(_,WSQ,WSNQ,no):-WSNQ \= [].
get_answer(Q,_,_,unknown):-not_in_signature(Q).
get_answer(_,_,_,undecided).
```

The above described system is fully implemented and offers support for queries, answers and explanations. Explanations are written into an XML file, which is parsed by a visualization applet. The visualization of trees belonging to dialectical explanations is enhanced by allowing the user to zoom-in/out, implode/explode arguments, *etc*. The internal structure of an argument is hidden when imploding, and a unique tag is shown instead.

Lemma 1 (δ-Explanation Soundness). *Let $\mathcal{P}$ be a* DELP*-program, C an argument comparison criterion, and Q a schematic query posed to $\mathcal{P}$. Let E be the δ-Explanation returned in support of the answer A. Then E justifies (Definition 24) A.*

Lemma 2 (δ-Explanation Completeness). *Let $\mathcal{P}$ be a* DELP*-program, C an argument comparison criterion, and Q a schematic query posed to $\mathcal{P}$. Let E be the δ-Explanation returned in support of the answer A. Then E contains all the possible justifications (Definition 24) for any instance of A.*

5 Conclusions

In this paper, we have addressed the problem of providing explanation capabilities to an argumentation system. This is an important, and yet undeveloped field in the area. Our focus is put on argumentation systems based on a dialectical proof procedure, studying *dialectical explanations*. We have defined an abstract system and a concrete reification with explanation facilities. We consider the structures that provide information on the warrant status of a literal. As the system has been implemented, we are developing applications that use the δ-Explanation system as subsystem.

Acknowledgments

This work was partially supported by CONICET (PIP 5050), ANPCyT (PICT 15043, PAV076), Project TIN2006-15662-C02-02 (MEC, Spain) and SGCyT-UNS.

References

1. L. Amgoud, N. Maudet, and S. Parsons. An argumentation-based semantics for agent communication languages. In *Proc. of the 15th. ECAI, Lyon, France*, pages 38–42, 2002.
2. K. Atkinson, T. J. M. Bench-Capon, and P. McBurney. Multi-agent argumentation for edemocracy. In *Proceedings of the Third European Workshop on Multi-Agent Systems, Brussels, Belgium*, pages 35–46. Koninklijke Vlaamse Academie, 2005.
3. P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 1:2(128):203–235, 2001.
4. D. Carbogim, D. Robertson, and J. Lee. Argument-based applications to knowledge engineering. *The Knowledge Engineering Review*, 15(2):119–149, 2000.
5. C. Chesñevar, A. Maguitman, and R. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.
6. C. Chesñevar and G. Simari. A lattice-based approach to computing warranted belief in skeptical argumentation frameworks. In *Proc. of the 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2007), Hyberabad, India*, page (in press), January 2007.
7. C. Chesñevar, G. Simari, T. Alsinet, and L. Godo. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. of the Intl. Conf. in Uncertainty in Art. Intelligence. (UAI 2004). Banff, Canada*, pages 76–84, July 2004.
8. C. Chesñevar, G. Simari, and L. Godo. Computing dialectical trees efficiently in possibilistic defeasible logic programming. *Proc. of the 8th Intl. Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005)*, pages 158–171, September 2005.
9. P. Dung. On the Acceptability of Arguments and its Fundamental Role in Nomonotonic Reasoning and Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–358, 1995.
10. Marcelo A. Falappa, Gabriele Kern-Isberner, and Guillermo R. Simari. Explanations, belief revision and defeasible reasoning. *Artif. Intell.*, 141(1):1–28, 2002.
11. A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.
12. Giovanni Guida and Marina Zanella. Bridging the gap between users and complex decision support systems: the role of justification. In *ICECCS '97: Proc. 3rd IEEE Int. Conf. on Engineering of Complex Computer Systems*, pages 229–238, Washington, DC, 1997.
13. H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *ICAIL*, pages 53–62, 1999.
14. A. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic Programming*, 9(4):515:562, 1999.
15. Carmen Lacave and Francisco J. Diez. A review of explanation methods for heuristic expert systems. *Knowl. Eng. Rev.*, 19(2):133–146, 2004.
16. B. Moulin, H. Irandoust, M. Bélanger, and G. Desbordes. Explanation and argumentation capabilities: Towards the creation of more persuasive agents. *Artif. Intell. Rev.*, 17(3):169–222, 2002.
17. S. Parsons, C. Sierrra, and N. Jennings. Agents that Reason and Negotiate by Arguing. *Journal of Logic and Computation*, 8:261–292, 1998.
18. H. Prakken and G. Sartor. The role of logic in computational models of legal argument - a critical survey. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, pages 342–380. Springer, 2002.
19. H. Prakken and G. Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F.Guenther, editors, *Handbook of Philosophical Logic*, pages 219–318. Kluwer Academic Publishers, 2002.
20. L. Richard Ye and Paul E. Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *MIS Q.*, 19(2):157–172, 1995.