# Dialectical Explanations in Defeasible Argumentation*

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

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur – Bahía Blanca, Argentina
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
{ajg,ndr,grs}@cs.uns.edu.ar

**Abstract.** This work addresses the problem of providing explanation capabilities to an argumentation system. Explanation in defeasible argumentation is an important, and yet undeveloped field in the area. Therefore, we move in this direction by defining a concrete argument system with explanation facilities.

We consider the structures that provide information on the warrant status of a literal. Our focus is put on argumentation systems based on a dialectical proof procedure, therefore we study *dialectical explanations*. Although arguments represent a form of explanation for a literal, we study the complete set of dialectical trees that justifies the warrant status of a literal, since this set has proved to be a useful tool to comprehend, analyze, develop, and debug argumentation systems.

## 1 Introduction

There has been attention focused on the role of explanations from several areas of Artificial Intelligence –especially from the expert systems community [1,2,3]. A few of them treat explanations in relation with argument systems [4]. In the literature, often an argument is regarded as an explanation for a certain literal. That is, the claim being explained is put under discussion, and only then it will be accepted or not. In belief revision, the role of explanations has also been studied [5]: a new perception 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 are concerned with the type of explanations that give the necessary information to understand the warrant status of a literal. Since our focus is put on argumentation systems based on a dialectical proof procedure, we study *dialectical explanations* (from now on, $\delta$-Explanations). Although we recognize arguments as an explanation for a literal, we are interested in obtaining the complete set of dialectical trees that justify the warrant status of a literal. We show how $\delta$-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 $\delta$-Explanation, are given throughout the paper.

* Partially supported by Universidad Nacional del Sur, CONICET (PIP 5050), and Agencia Nacional de Promoción Científica y Tecnológica.

K. Mellouli (Ed.): ECSQARU 2007, LNAI 4724, pp. 295–307, 2007.

An interesting review about explanations in heuristic expert systems is given in [1], in which a definition is given: "*...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. We believe that 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 [2]. The typology there described 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; 3) *strategy:* a high-level goal structure determining the problem-solving strategy used. From this typology, the authors claim that –through their empirical analysis– the most useful type of explanation is "justification". We contend that the type of explanations we propose correspond to both the "justification" and the "strategy" types; that is, we are giving not only the strategy used by the system to achieve the conclusion, but also the rationale behind each argument, which is clearly stated by its role in the dialectical tree.

We agree with [4], in that *"argumentation and explanation facilities in knowledge-base 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 in this paper stress this point.

This paper is organized as follows: first we will briefly outline the DeLP concepts, then we will introduce $\delta$-Explanations and their relation with DeLP's answers, and finally we will discuss the related literature.

## 2 DeLP Overview

Defeasible Logic Programming (DeLP) combines results of Logic Programming and Defeasible Argumentation. The system is fully implemented and available online [6]. A brief explanation is included below (see [7] for full details). It has the declarative capability of representing weak information in the form of *defeasible rules*, and a defeasible argumentation inference mechanism for warranting the entailed conclusions. A DeLP-program $\mathcal{P}$ is a set of facts, strict rules and defeasible rules defined as follows. Facts are ground literals representing atomic information or the negation of atomic information using the strong negation "$\sim$" (*e.g.*, $chicken(little)$ or $\sim scared(little)$). *Strict Rules* represent non-defeasible information and are denoted $L_0 \leftarrow L_1, \ldots, L_n$, where $L_0$ is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals (*e.g.*, $bird \leftarrow chicken$) or $\sim innocent \leftarrow guilty$). *Defeasible Rules* represent tentative information and are denoted $L_0 -\!\prec L_1, \ldots, L_n$, where $L_0$ is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals. (*e.g.*, $\sim flies -\!\prec chicken$ or $flies -\!\prec chicken, scared$).

When required, $\mathcal{P}$ is denoted $(\Pi, \Delta)$ distinguishing the subset $\Pi$ of facts and strict rules, and the subset $\Delta$ of defeasible rules (see Example 1). *Strong negation* is allowed in the head of rules, and hence may be used to represent contradictory knowledge. From a program $(\Pi, \Delta)$ contradictory literals could be derived. Nevertheless, the set $\Pi$ (which is used to represent non-defeasible information) must possess certain internal coherence. Therefore, no pair of contradictory literals can be derived from $\Pi$.

A defeasible rule is used to represent tentative information that may be used if nothing could be posed against it. Observe that strict and defeasible rules are ground. However, following the usual convention [8], some examples use "schematic rules" with variables. To distinguish variables, as usual, they start with an uppercase letter.

**Example 1.** *Consider the* DELP*-program* $(\Pi_1, \Delta_1)$ *where:*

$$\Pi_1 = \left\{ \begin{array}{ll} (bird(X) \leftarrow chicken(X)) & chicken(little) \\ chicken(tina) & bird(rob) \\ scared(tina) & \end{array} \right\}$$

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

*This program has three defeasible rules representing tentative information about the flying ability of birds in general, and about regular chickens and scared ones. It also has a strict rule expressing that every chicken is a bird, and three facts: 'tina' and 'little' are chickens, and 'tina' is scared.*

From a program is possible to derive contradictory literals, *e.g.*, from $(\Pi_1, \Delta_1)$ of Example 1 it is possible to derive $flies(tina)$ and $\sim flies(tina)$. For the treatment of contradictory knowledge DELP incorporates a defeasible argumentation formalism. This formalism allows the identification of the pieces of knowledge that are in contradiction, and a *dialectical process* is used for deciding which information prevails as warranted. This dialectical process (see below) involves the construction and evaluation of arguments that either support or interfere with the query under analysis. Once the analysis is done, the generated arguments will represent *an explanation* for the query. As we will show next, arguments that explain an answer for a given query will be shown in a particular way using *dialectical trees*. The definition of dialectical tree will be included below, but first, we will give a brief explanation of other related concepts (for the details see [7]).

**Definition 1 (Argument Structure).** *Let* $(\Pi, \Delta)$ *be a* DELP*-program,* $\langle \mathcal{A}, L \rangle$ *is an argument structure for a literal* $L$ *from* $(\Pi, \Delta)$*, if* $\mathcal{A}$ *is the minimal set of defeasible rules* ($\mathcal{A} \subseteq \Delta$)*, such that: (1) there exists a defeasible derivation for* $L$ *from* $\Pi \cup \mathcal{A}$*, and (2) the set* $\Pi \cup \mathcal{A}$ *is non-contradictory.*

**Example 2.** *From the* DELP*-program* $(\Pi_1, \Delta_1)$ *the following arguments can be obtained (due to space restrictions 'tina' will be abbreviated to 't' and 'flies(tina)' to 'f'):*
$\langle \mathcal{A}_1, f \rangle = \langle \{flies(t) -\!\prec bird(t)\}, flies(t) \rangle$
$\langle \mathcal{A}_2, \sim f \rangle = \langle \{\sim flies(t) -\!\prec chicken(t)\}, \sim flies(t) \rangle$
$\langle \mathcal{A}_3, f \rangle = \langle \{flies(t) -\!\prec chicken(t), scared(t)\}, flies(t) \rangle$

In DELP a literal $L$ is *warranted* if there exists a non-defeated argument $\mathcal{A}$ supporting $L$. To establish if $\langle \mathcal{A}, L \rangle$ is a non-defeated argument, *defeaters* for $\langle \mathcal{A}, L \rangle$ are

considered, *i.e.*, counter-arguments that by some criterion are preferred to $\langle \mathcal{A}, L \rangle$. It is important to note that in DELP the argument comparison criterion is modular and thus, the most appropriate criterion for the domain that is being represented can be selected. In the examples in this paper we will use *generalized specificity* [9], a criterion that favors two aspects in an argument: it prefers (1) a *more precise* argument (*i.e.*, with greater information content) or (2) a *more concise* argument (*i.e.*, with less use of rules). Using this criterion in Ex. 2 $\langle \mathcal{A}_3, f \rangle$ is preferred to $\langle \mathcal{A}_2, \sim f \rangle$ (more precise) and $\langle \mathcal{A}_2, \sim f \rangle$ is preferred to $\langle \mathcal{A}_1, f \rangle$ (the later use the strict rule $bird(X) \leftarrow chicken(X)$).

A defeater $\mathcal{D}$ for an argument $\mathcal{A}$ can be *proper* ($\mathcal{D}$ is preferred to $\mathcal{A}$) or *blocking* (same strength). Since defeaters are arguments, there may exist defeaters for them, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called *argumentation line* is constructed, where each argument defeats its predecessor. To avoid undesirable sequences, that may represent circular or fallacious argumentation lines, in DELP an argumentation line has to be *acceptable*, that is, it has to be finite, an argument can not appear twice, and supporting arguments, *i.e.*, in odd positions, (resp. interfering arguments) have to be not contradictory (see [7]).

**Example 3.** *(Extends Ex. 2) The argument $\langle \mathcal{A}_2, \sim f \rangle$ is a proper defeater of $\langle \mathcal{A}_1, f \rangle$, and $\langle \mathcal{A}_3, f \rangle$ is a proper defeater of $\langle \mathcal{A}_2, \sim f \rangle$. Hence, $[\langle \mathcal{A}_1, f \rangle, \langle \mathcal{A}_2, \sim f \rangle, \langle \mathcal{A}_3, f \rangle]$ is an acceptable argumentation line.*

Clearly, there can be more than one defeater for a particular argument $\mathcal{A}$. Therefore, many acceptable argumentation lines could arise from $\mathcal{A}$, leading to a tree structure. Given an argument $\langle \mathcal{A}_0, h_0 \rangle$, a *dialectical tree* [7] for $\langle \mathcal{A}_0, h_0 \rangle$, denoted $\mathcal{T}(\langle \mathcal{A}_0, h_0 \rangle)$, is a tree where every node is an argument. The root of $\mathcal{T}(\langle \mathcal{A}_0, h_0 \rangle)$ is $\langle \mathcal{A}_0, h_0 \rangle$, and every inner node is a defeater (proper or blocking) of its parent. Leaves correspond to non-defeated arguments. In a dialectical tree every path from the root to a leaf corresponds to a different acceptable argumentation line. Thus, a dialectical tree provides a structure for considering all the possible acceptable argumentation lines that can be generated for deciding whether an argument is defeated. We call this tree *dialectical* because it represents an exhaustive dialectical analysis for the argument in its root.

Given a literal $h$ and an argument $\langle \mathcal{A}, h \rangle$ to decide whether a literal $h$ is warranted, every node in the dialectical tree $\mathcal{T}(\langle \mathcal{A}, h \rangle)$ is recursively marked as "$D$" (*defeated*) or "$U$" (*undefeated*), obtaining a marked dialectical tree $\mathcal{T}^*(\langle \mathcal{A}, h \rangle)$. Nodes are marked by a bottom-up procedure that starts marking all leaves in $\mathcal{T}^*(\langle \mathcal{A}, h \rangle)$ as "$U$"s. Then, for each inner node $\langle \mathcal{B}, q \rangle$ of $\mathcal{T}^*(\langle \mathcal{A}, h \rangle)$, (a) $\langle \mathcal{B}, q \rangle$ will be marked as "$U$" iff every child of $\langle \mathcal{B}, q \rangle$ is marked as "$D$", or (b) $\langle \mathcal{B}, q \rangle$ will be marked as "$D$" iff it has at least a child marked as "$U$".

Given an argument $\langle \mathcal{A}, h \rangle$ obtained from $\mathcal{P}$, if the root of $\mathcal{T}^*(\langle \mathcal{A}, h \rangle)$ is marked as "$U$", then we will say that $\mathcal{T}^*(\langle \mathcal{A}, h \rangle)$ *warrants* $h$ and that $h$ is *warranted* from $\mathcal{P}$.

In this paper, marked dialectical trees will be depicted as a tree of labelled triangles where edges denote the defeat relation (in Figure 1 three marked dialectical trees are shown). A double arrow edge represents a blocking defeat, whereas a single arrow represents a proper defeat. An argument $\langle \mathcal{A}, h \rangle$ will be depicted as a triangle, where its upper vertex is labelled with the conclusion $h$, and the set of defeasible rules $\mathcal{A}$ are associated with the triangle itself. At the right of each node the associated mark ("$U$" or "$D$") will be shown.

**Example 4.** *(Extends Ex. 3) Figure 1 shows the marked dialectical tree for* $\mathcal{T}^*(\langle \mathcal{A}_1, f\rangle)$ *(the leftmost tree), which has only one argumentation line. Observe that the argument* $\langle \mathcal{A}_2, \sim f\rangle$ *interferes with the warrant of 'flies(tina)' and the argument* $\langle \mathcal{A}_3, f\rangle$ *reinstates* $\langle \mathcal{A}_1, f\rangle$*. The root of* $\mathcal{T}^*(\langle \mathcal{A}_1, f\rangle)$ *is marked as "U" and therefore the literal 'flies(tina)' is warranted.*

## 3 DeLP Answers and $\delta$-Explanations

Next, we will define *queries*, *answers* and *explanations*. 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 2 (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 represent the set of* DELP*-queries that unify with the schematic one.*

The dialectical process for warranting a query involves the construction and evaluation of several arguments that either support or interfere with the query under analysis. These generated arguments are connected through the defeat relation and are organized in dialectical trees. Observe that given a query $Q$ there could exist different arguments that support $Q$, and each argument will generate a different dialectical tree. Therefore, as we will show below, the returned answer for $Q$ will be only 'the tip of the iceberg' of a set of several dialectical trees that have been explored to support the resulting answer.

Thus, to understand why a query has a particular answer, it is essential to consider which arguments have been generated and what connections exist among them. In DELP, $\delta$*-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 $\delta$-Explanation for a DELP-query follows, whereas explanations for schematic queries will be introduced by the end of this Section.

### 3.1 $\delta$-Explanations for DELP-Queries

We contend that $\delta$-Explanations are a central part of an argumentation system whose proof procedure is based on dialectical trees, because they allow to visualize the reasoning carried out by the system, and the support for the answer. It is clear that without this information at hand it will be very difficult to understand the returned answer. Next, we will introduce explanations for ground queries. Then, we will generalize explanations for schematic queries. Given a literal $L$, the complement with respect to strong negation will be denoted $\overline{L}$ (*i.e.*, $\overline{a}$=$\sim a$ and $\overline{\sim a}$=$a$).

**Definition 3 ($\delta$-Explanation).**
*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* $\mathcal{E}_{\mathcal{P}}(Q) = \{\mathcal{T}^*(\langle \mathcal{A}_0, Q\rangle),\ldots,\mathcal{T}^*(\langle \mathcal{A}_n, Q\rangle)\} \cup \{\mathcal{T}^*(\langle \mathcal{B}_0, \overline{Q}\rangle),\ldots,\mathcal{T}^*(\langle \mathcal{B}_m, \overline{Q}\rangle)\}$.

Now it is possible to define DELP-answers in terms of their $\delta$-Explanation.

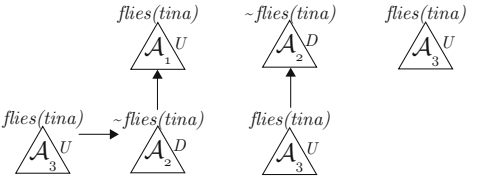


**Fig. 1.** $\delta$-Explanation for $flies(tina)$

**Definition 4** (DELP**-answer).** *Given a* DELP*-program $\mathcal{P}$ and a* DELP*-query $Q$, the answer for $Q$ is either:*

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 5.** *(Extends Ex. 4) Figure 1 shows the $\delta$-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 $\delta$-Explanation of Figure 1 is also an explanation for query '$\sim flies(tina)$' which answer is* NO*. Finally, observe that the answer for 'walks(tim)' is* UNKNOWN*, because it is not in the program signature.*

**Remark 1.** *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 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), $\delta$-Explanations become essential for understanding the reasons that support an answer.

**Example 6.** *Consider the* DELP*-program $(\Pi_6, \Delta_6)$:*

$$\Pi_6 = \{q, t\}\ \Delta_6 = \left\{ \begin{array}{l} (r \prec q)\ (\sim r \prec q, s) \\ (r \prec s)\ (\sim r \prec t) \end{array} \right\}$$

*where the following arguments can be built:*
$\langle \mathcal{R}_1, \sim r \rangle = \langle \{\sim r \prec t\}, \sim r \rangle \quad \langle \mathcal{R}_2, r \rangle = \langle \{r \prec q\}, r \rangle$
*From this program the answer for the query 'r' is* UNDECIDED*, and Figure 2 shows its $\delta$-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 $\delta$-Explanation is the empty set (i.e., $\mathcal{E}_{(\Pi_6, \Delta_6)}(s) = \emptyset$).*

**Remark 2.** DELP*-queries with* UNKNOWN *answers always have an empty $\delta$-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.*

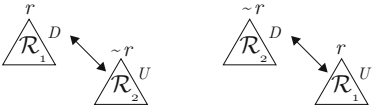

**Fig. 2.** $\delta$-Explanation $\mathcal{E}_{(\Pi_6,\Delta_6)}(r)$

Example 7 shows how the introduction of a single fact in $(\Pi_6, \Delta_6)$ makes a significant difference in $\mathcal{E}_{(\Pi_6,\Delta_6)}(r)$.

**Example 7.** *(Extends Ex. 6) Consider the* DELP*-program* $(\Pi_6 \cup \{s\}, \Delta_6)$ *where the fact 's' is added to the program of Example 6. If we query for 'r' again, we get the answer* NO *with the* $\delta$*-Explanation shown in Figure 3. Note that this* $\delta$*-Explanation consists now of two more trees than the one in the previous example. This is so because there are two newly generated arguments:*

$\langle \mathcal{R}_3, r\rangle = \langle \{r \prec s\}, r\rangle \quad \langle \mathcal{R}_4, \sim r\rangle = \langle \{\sim r \prec q, s\}, \sim r\rangle$

It is our contention that, in DELP, the answer for a query can be easily explained by presenting the user the associated dialectical trees. From this set of trees the answer becomes thoroughly justified, and the context of the query is revealed. The following examples have more elaborated DELP-programs and the $\delta$-Explanations show that a defeater $\mathcal{D}$ for $\mathcal{A}$ may attack an inner point of $\mathcal{A}$.

**Example 8.** *Consider the* DELP*-program* $(\Pi_8, \Delta_8)$*:*

$$\Delta_8 = \left\{ \begin{array}{ll} (a \prec b) & (b \prec c) \\ (\sim b \prec d) & (d \prec e) \\ (\sim d \prec f, e) & (\sim b \prec e) \\ (a \prec x) & (x \prec c) \\ (\sim x \prec e) & (a \prec h) \\ (h \prec f) & (\sim h \prec i) \end{array} \right\} \Pi_8 = \{c, e, f\}$$

*where the following arguments can be built:*

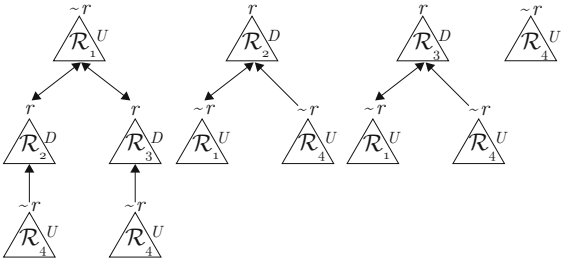

**Fig. 3.** $\delta$-Explanation $\mathcal{E}_{(\Pi_6\cup\{s\},\Delta_6)}(r)$

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

*From $(\Pi_8, \Delta_8)$ the answer for 'a' is* YES*, and the answer for '$\sim a$' is* NO*. As stated in Remark 1, although both queries have different answers, they both have the same $\delta$-Explanation, which is depicted in Figure 4. In that figure, sub-arguments are represented as smaller triangles contained in the triangle which corresponds to the main argument at issue. For instance, the argument $\langle \mathcal{B}_2, \sim b\rangle$ defeats $\langle \mathcal{B}_1, b\rangle$ that is a subargument of $\langle \{(a \prec b), (b \prec c)\}, a\rangle$.*

**Example 9.** *Consider the* DELP*-program $(\Pi_8 \cup \{i\}, \Delta_8)$ where the fact 'i' is added to the program of Example 8. Now the argument $\langle \mathcal{H}_2, \sim h\rangle$ can be generated which is a defeater for $\langle \mathcal{H}_1, h\rangle$ (a subargument of $\langle \mathcal{A}_1, a\rangle$):*
$\langle \mathcal{H}_2, \sim h\rangle = \langle \{\sim h \prec i\}, \sim h\rangle$ $\quad$ $\langle \mathcal{H}_1, h\rangle = \langle \{h \prec f\}, h\rangle$
*Here, argument $\mathcal{H}_2$ blocks argument $\mathcal{H}_1$ (subargument of $\mathcal{A}_1$), leaving no undefeated arguments for 'a'; then, the answer for both 'a' and '$\sim a$' is* UNDECIDED*. The rest of the explanation remains the same as the one in Figure 4.*

From the DELP programmer point-of-view, $\delta$-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 behaviour 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 $\delta$-Explanation. Finally, 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 1.

### 3.2 Explanations for Schematic Queries

A *schematic query* is a query that has at least one variable (see Definition 2), and hence, it represents the set of DELP-queries that unify with it. Now, we will extend the definition of $\delta$-Explanation to include schematic queries. Consider again the DELP-program

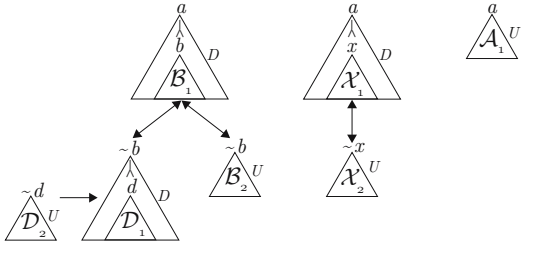


**Fig. 4.** $\delta$-Explanation $\mathcal{E}_{(\Pi_8, \Delta_8)}(a)$

of Example 1, the schematic query $flies(X)$ has 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. An explanation for a schematic query will be the set of $\delta$-Explanations of those instances whose answers are YES, NO, or UNDECIDED.

**Definition 5 (Generalized $\delta$-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 $\delta$-Explanation for the* DELP*-query $Q_i$ ($1 \leq i \leq z$) from program $\mathcal{P}$. Then, the* generalized $\delta$-Explanation *for $Q$ in $\mathcal{P}$ is $\mathcal{E}_{\mathcal{P}}(Q) = \{ \mathcal{E}_{\mathcal{P}}(Q_1), \ldots, \mathcal{E}_{\mathcal{P}}(Q_z)\}$.*

Observe that a $\delta$-Explanation (Definition 3) is a particular case of a Generalized $\delta$-Explanation, where the set $\mathcal{E}_{\mathcal{P}}(Q)$ is a singleton.

**Example 10.** *Consider again the* DELP*-program $(\Pi_1, \Delta_1)$, 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 5. This explanation also shows that the other instance ($\sim flies(tina)$) is not warranted.*

*It is important to 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 $\delta$-Explanation for $flies(X)$ is the same as the one for $\sim flies(X)$, depicted in Figure 5 (see Remark 1).*

**Definition 6 (**DELP**-answer for a schematic query).** *Given a* DELP*-program $\mathcal{P}$ and a schematic query $Q$, the answer for $Q$ is*

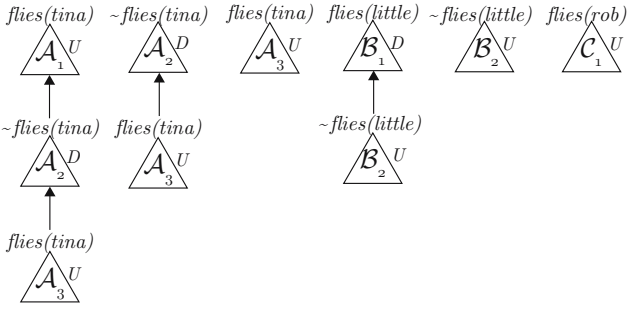


**Fig. 5.** Generalized $\delta$-Explanation for '$\sim flies(X)$'

- 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 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 $Q_i$, and there exists an instance $\overline{Q_i}$ of $\overline{Q}$ such that at least one tree in $\mathcal{E}_{\mathcal{P}}(\overline{Q_i})$ 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 $\mathcal{P}$.*

Observe that Definition 4 is a particular case of the previous definition, where there is a single instance of $Q$.

**Example 11.** *Consider the following* DELP*-program:*

$$\Pi_{11} = \left\{ \begin{array}{ll} adult(peter) & adult(annie) \\ unemployed(peter) & student(annie) \end{array} \right\}$$

$$\Delta_{11} = \left\{ \begin{array}{l} has\_a\_car(X) -\!\prec adult(X) \\ \sim has\_a\_car(X) -\!\prec unemployed(X) \\ \sim has\_a\_car(X) -\!\prec student(X) \end{array} \right\}$$

*where the following arguments can be built('$has\_a\_car$' was replaced by '$car$', '$annie$' by '$a$', and '$peter$' by '$p$'):*
$\langle \mathcal{A}_1, car(a) \rangle$ = $\langle \{car(a) -\!\prec adult(a)\}, car(a) \rangle$
$\langle \mathcal{A}_2, \sim car(a) \rangle$ = $\langle \{\sim car(a) -\!\prec student(a)\}, \sim car(a) \rangle$
$\langle \mathcal{P}_1, car(p) \rangle$ = $\langle \{car(p) -\!\prec adult(p)\}, car(p) \rangle$
$\langle \mathcal{P}_2, \sim car(p) \rangle$ = $\langle \{\sim car(p) -\!\prec unemployed(p)\}, \sim car(p) \rangle$

*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 6, 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.*

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.

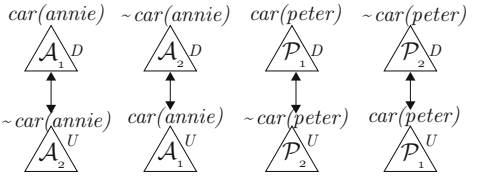


**Fig. 6.** Generalized $\delta$-Explanation for '$has\_a\_car(X)$'

The $\delta$-Explanations system receives a DELP-program P, a query Q and an argument comparison criterion C, and returns a $\delta$-Explanation EX along with the proper answer ANS. The system is described by the following algorithm in a Prolog-like notation:

```
d_Explanations(P,C,Q,EX,ANS):- warrants(P,C,Q,WSQ),
             complement(Q,NQ), warrants(P,C,NQ,WSNQ),
             get_trees(WSQ,WSNQ,EX), get_answer(Q,WSQ,WSNQ,ANS).
warrants(Q,WS):- findall((Q,TREES),warrant(Q,TREES),WS).
get_answer(_,WSQ,WSNQ,yes):- WSQ \= [].
get_answer(_,WSQ,WSNQ,no):- WSNQ \= [].
get_answer(Q,_,_,unknown):- not_in_signature(Q).
get_answer(_,_,_,undecided).
```

Predicate warrant/2 takes a query and attempts to warrant it; it does so by building dialectical trees. In case the query is warranted, the dialectical trees built are 'saved' along with the query. Different instances of a query can be obtained via backtracking. Predicate warrants/2 takes a query Q and returns all its warranted instances (along with their corresponding trees) within a list. Predicate get_trees/3 retrieves the dialectical trees information from the warranted instances for both Q and $\sim$Q. Finally, predicate get_answer/4 takes the query, both lists of warranted instances (for Q and $\sim$Q), and returns the answer.

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 ($\delta$-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 $\delta$-Explanation returned in support of the answer $A$. Then $E$ justifies (Definition 6) $A$.*

**Lemma 2 ($\delta$-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 $\delta$-Explanation returned in support of the answer $A$. Then $E$ contains all the possible justifications (Definition 6) for any instance of $A$.*

## 4 Related Work

A very thorough survey relating explanation and argumentation capabilities can be found in [4]. Although the authors are mainly concerned about negotiation/persuasion, and interactive/collaborative explanations, the discussion Section of that article poses really interesting issues about the integration of explanation and argumentation; for instance, whether the same knowledge base can be used to generate both explanatory and argumentative information. In our approach, we do extract all the information from the given knowledge base (*i.e.*, the DELP-program) to return both kinds of information.

In [4], the authors claim that these two areas (*i.e.*, argumentation and explanation facilities in knowledge-base systems) should be "investigated in conjunction". Our paper tries to move forward in that direction, providing means to "better understand the mechanisms underlying the activities of explanation and argumentation".

Recently, Douglas Walton [10] has offered a dialogue theory of explanation. In that work a successful explanation is defined as transfer of understanding in a dialogue system where a questioner and a respondent take part. The questioner begins by asking a question seeking to understand some piece of information and the respondent gives a reply that conveys understanding of that information to the questioner. His approach follows a different path than ours, focussing in the distinction between *explanation* and *argument* and defining an explanation as a new speech act.

Our approach handles $\delta$-Explanations within argumentation systems through a graphical representation of dialectical trees. Visualization in argumentation has been addressed in [11]. In that paper, the objective is to provide a visual tool that does not require the reader to understand logic to be able to follow the argumentative process shown by the system. To achieve this, they use an animated argumentation space: arguments are introduced one by one in the process to allow for a more comprehensive visualization. They also allow to see this space in a static manner. Both ways give the user the possibility to navigate the space at will, or in auto-pilot mode. Every element taking part of the argumentation process is represented graphically: conflicts are highlighted and arguments are tagged with the role they are playing in the whole process.

Although the article by Schroeder uses argumentation trees in a similar way as we do, we focus on explanations; that is, we are concerned with providing the whole context corresponding to the query. Our explanations are represented in such a way that they are useful to both humans and software agents.

## 5 Conclusions and Future Work

Future work includes further research about additional information that can be attached to the current form of the $\delta$-Explanations. In particular, we are currently formalizing the notion of *discarded arguments*. These arguments are discarded by the system in the sense that their introduction into an acceptable argumentation line renders it fallacious. At the moment, we have singled out two reasons for an argument $\mathcal{A}$ to be discarded: (1) Non-attacking arguments: when $\mathcal{A}$ conflicts with the last argument in the line, but does not attack it (*i.e.*, the last argument is better than $\mathcal{A}$ wrt. the comparison criterion); (2) Double-blocking arguments: when the final argument in the line $\mathcal{A}_n$ is a blocking defeater of the preceding argument $\mathcal{A}_{n-1}$, and $\mathcal{A}$ is, in turn, a blocking defeater for $\mathcal{A}_n$. More dialectical constraints can be considered thus adding more types of discarded arguments. It is interesting to show discarded arguments within a $\delta$-Explanation, because the user has the possibility of analyzing why a particular argument has not been included into the explanation. Sometimes, it is not clear why these situations occur.

We have addressed the problem, not often considered, of providing explanation capabilities to an argumentation system. We have defined a concrete argument system 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 uses the $\delta$-Explanation system as subsystem.

## References


1. Lacave, C., Diez, F.J.: A review of explanation methods for heuristic expert systems. Knowl. Eng. Rev. 19(2), 133–146 (2004)
2. Ye, L.R., Johnson, P.E.: The impact of explanation facilities on user acceptance of expert systems advice. MIS Q. 19(2), 157–172 (1995)
3. Guida, G., Zanella, M.: Bridging the gap between users and complex decision support systems: the role of justification. In: ICECCS '97: Proc. 3rd IEEE International Conference on Engineering of Complex Computer Systems, Washington, pp. 229–238. IEEE Computer Society Press, Los Alamitos (1997)
4. Moulin, B., Irandoust, H., Bélanger, M., Desbordes, G.: Explanation and argumentation capabilities: Towards the creation of more persuasive agents. Artif. Intell. Rev. 17(3), 169–222 (2002)
5. Falappa, M.A., Kern-Isberner, G., Simari, G.R.: Explanations, belief revision and defeasible reasoning. Artif. Intell. 141(1), 1–28 (2002)
6. DeLP: `http://lidia.cs.uns.edu.ar/delp`
7. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming 4(1), 95–138 (2004)
8. Lifschitz, V.: Foundations of logic programs. In: Brewka, G. (ed.) Principles of Knowledge Representation, pp. 69–128. CSLI Pub. (1996)
9. Stolzenburg, F., García, A., Chesñevar, C.I., Simari, G.R.: Computing Generalized Specificity. Journal of Non-Classical Logics 13(1), 87–113 (2003)
10. Walton, D.: A new dialectical theory of explanation. Philosophical Explorations 7(1), 71–89 (2004)
11. Schroeder, M.: Towards a visualization of arguing agents. Future Generation Computer System 17(1), 15–26 (2000)