Expert Systems with Applications xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems

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

Artificial Intelligence Research and Development Laboratory, Department of Computer Science and Engineering, Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca, Argentina

ARTICLE INFO

Keywords:
Knowledge-based systems
Explanation support
Abstract argumentation
Structured argumentation
Defeasible Logic Programming

ABSTRACT

The concept of explanation has received attention from different areas in Computer Science, particularly in the knowledge-based systems and expert systems communities. At the same time, argumentation has evolved as a new paradigm for conceptualizing commonsense reasoning, resulting in the formalization of different argumentation frameworks and the development of several real-world argument-based applications. Although the notions of explanation and argument for a claim share many common elements in knowledge-based systems their interrelationships have not yet been formally studied in the context of the current argumentation research in Artificial Intelligence. This article explores these ideas by providing a new perspective on how to formalize dialectical explanation support for argument-based reasoning. To do this, we propose a formalization of explanations for abstract argumentation frameworks with dialectical constraints where different emerging properties are studied and analyzed. As a concrete example of the formalism introduced we show how it can be fleshed out in an implemented rule-based argumentation system.

1. Introduction and motivations

The notion of *explanation* has received attention in different related areas of Computer Science, such as from the knowledge-based systems and expert systems communities (Guida & Zanella, 1997; Lacave & Diez, 2004; Ye & Johnson, 1995). An interesting review about explanations in heuristic expert systems is given in Lacave and Diez (2004), which offers the following appealing 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 the last two decades, argumentation (Bench-Capon & Dunne, 2007; Besnard & Hunter, 2008; Chesñevar, Maguitman, & Loui, 2000; Prakken & Vreeswijk, 2002; Rahwan & Simari, 2009) has evolved as an attractive paradigm for conceptualizing commonsense reasoning, resulting in the formalization of different argumentation frameworks and the development of several real-world argument-based applications. Argumentation, as a research subarea of knowledge representation and reasoning, is of particular importance mainly because it allows to obtain conclusions from repositories containing often inconsistent, incomplete, and uncertain information thus becoming suitable to handle reasoning tasks in knowledge-based systems. During the last decades, the use of argumentation has expanded at increasing pace, driven in part by theoretical advances but also by successful demonstrations of a substantial number of practical applications domains, such as legal reasoning (Prakken & Sartor, 2002), knowledge engineering (Carbogim, Robertson, & Lee, 2000), multiagent systems (Amgoud, Maudet, & Parsons, 2002; Parsons, Sierrra, & Jennings, 1998), and e-government (Atkinson, Bench-Capon, & McBurney, 2005), among many others.

Moulin, Irandoust, Bélanger, and Desbordes (2002), present a review of the literature of explanation systems and argumentation systems. The authors describe the work of researchers leading to the enhancement of the explanation capabilities of knowledge-based and decision support systems, while other researchers developed argumentative techniques for software systems.

They remark that "*it would be interesting for the researchers belonging to these different communities to share their experiences and to develop systems that take advantage of the advances gained in each domain.*"; concluding emphatically that "*argumentation and explanation facilities in knowledge-based systems should be investigated in conjunction.*" However, to the best of our knowledge, current state of the art of argumentation research in Computer Science, does not reflect the goal of studying these two notions of *explanation* and *argument* together as a research goal.

* Corresponding author. Tel.: +54 2914595135; fax: +54 2914595136.
E-mail addresses: ajg@cs.uns.edu.ar (A.J. García), cic@cs.uns.edu.ar (C.I. Chesñevar), ndr@cs.uns.edu.ar (N.D. Rotstein), grs@cs.uns.edu.ar (G.R. Simari).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.12.036

This article represents an effort to fill this gap by providing a new perspective on formalizing *dialectical explanation support* for argument-based reasoning. With that intent, we introduce the concept of *dialectical explanation* that will denote δ-explanation; following the terminology of Lacave and Diez (2004), δ-explanations will be used to *explain* why an argument has a particular warranting status by *exposing* the context under which that argument is considered and the analysis carried out in order to obtain such status.

Hence, one of the contributions of this paper is to introduce a formalization of δ-explanations for a class abstract argumentation frameworks where dialectical constraints are included; for this formalization, our presentation explores and analyzes different emerging properties of δ-explanations. As another interesting contribution, the proposed explanation formalisms is also applied to an implemented rule-based argumentation system. Thus, as we will show below, δ-explanations can enhance both abstract and rule-based argumentation systems. In abstract argumentation, δ-explanations can be a useful tool to comprehend and analyze the interactions among arguments that are under consideration. On the other hand, in rule-based argumentation systems where arguments are endowed with structure (Besnard & Hunter, 2001; Bondarenko, Dung, Kowalski, & Toni, 1997; Chesñevar, Simari, Alsinet, & Godo, 2004; García & Simari, 2004; Prakken, 2010; Simari & Loui, 1992), δ-explanations have the additional capability of aiding in the understanding of how knowledge should be represented and debugging of the underlying knowledge base.

Intuitively, an argument can be seen as a piece of reasoning that supports a claim from certain evidence. Argumentation systems define a way for establishing which claims (or arguments) can be accepted as warranted. In this context, abstract argumentation frameworks (Dung, 1995) have played a major role as a way of understanding argument-based inference, resulting in different semantics for argumentation (Baroni & Giacomin, 2009). The goal of an argumentation semantics is to characterize the set of arguments that should be rationally accepted as justified or warranted. However, the information about *why* an argument, or its claim for that matter, is warranted remains usually hidden within the process.

Consider for example the following scenario where an agent has to decide about having an opera night. Bob happens to be an aficionado to the opera and there is an opera show tonight (argument $\mathcal{O}_{show}$). Besides, today is Bob's birthday and he usually gets together with friends ($\mathcal{O}_{get}$). Bob usually goes to the opera house with friends ($\mathcal{O}_{friends}$). However, today Bob's best friend is coming with her baby to celebrate his birthday, and is not a good idea to bring a baby to the opera ($\mathcal{O}_{baby}$). Observe that the argument $\mathcal{O}_{show}$ supports the claim "go to the show" that is defeated (or attacked) by $\mathcal{O}_{get}$ and vice versa. The argument $\mathcal{O}_{get}$ is in turn defeated by $\mathcal{O}_{friends}$ ("opera with friends"), thus reinstating the argument $\mathcal{O}_{show}$. Note also that the argument $\mathcal{O}_{baby}$ ("friend with baby") defeats both $\mathcal{O}_{get}$ and $\mathcal{O}_{friends}$.

As has been mentioned, an argumentation semantics is needed to obtain the set of accepted arguments (*e.g.*, $\{\mathcal{O}_{baby}, \mathcal{O}_{get}\}$). Fig. 1 depicts a directed graph usually used for abstract argumentation frameworks where nodes are labeled with the arguments and directed edges, represented as arrows, the defeat relation. Although the graph shows all existing arguments and how arguments defeat other arguments, this type of graph does not explain the analysis performed to establish if an argument can be accepted. Even if we annotate the graph telling which arguments are accepted, there would be no indication regarding the analysis that produced the different statuses of these arguments.

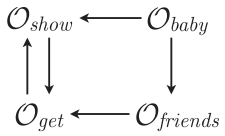

Fig. 1. Graph with argument labels and the defeat relation.

In our formalization, a δ-explanation will be a structure that shows why a particular claim (or argument) is considered warranted. For instance, following the example above, the explanation will show why Bob finally decides not to go to the opera when faced to the described scenario. Thus, an explanation will provide a useful tool to comprehend, analyze, develop, and debug argumentation systems. In the context of the terminology used by Lacave and Diez (2004), we *explain* why an argument supporting a claim is warranted through *exposing* or presenting the whole set of arguments that have been considered and how these arguments are interrelated. We postulate that this information is *understandable* from the receiver's point of view, because all the arguments considered, their statuses (*i.e.*, defeated/undefeated), and their interrelations are explicitly shown. This type of information should be more *satisfactory* and useful for the receiver, because it contains all the elements at stake in the analysis that supports the answer.

Recently, in Walton (2004), Walton introduced a philosophical analysis of explanations. He states that "*the purpose of an explanation is to get the hearer to understand something that he already accepts as a fact*", and that "*an explanation is seen as a transfer of understanding from a respondent to a questioner in a dialogue.*" In accordance with that point of view, in our approach the purpose of an explanation is to transfer the understanding of how the warrant status of a particular argument has been obtained from a given argumentation framework.

Some preliminary work related to this paper has been reported in two workshop papers (García, Chesñevar, Rotstein, & Simari, 2007; García, Rotstein, Chesñevar, & Simari, 2009) and previously in García, Rotstein, and Simari (2007); nevertheless, the work presented here extends them in several ways. The abstract formalization of δ-explanations is completely redefined and extended to cope with both abstract and rule-based argumentation frameworks. A deeper analysis of the formalism is performed and several interesting properties are introduced. The use of the formalism is shown through examples both in abstract frameworks and in an implemented argumentation system.

The rest of this paper is structured as follows. Next, in Section 2 we will present the basic ideas of an abstract argumentation framework with dialectical constraints, which includes several concepts common to most argument-based formalisms. Section 3 introduces the formalization of δ-explanations and shows several properties. Section 4 presents a concrete reification of the proposed abstract framework based on Defeasible Logic Programming (DELP). Finally, Section 5 discusses related work and conclusions.

2. Abstract argumentation frameworks with dialectical constraints

In this section we recall argumentation theories as introduced in Chesñevar and Simari (2007) and Chesñevar, Simari, and Godo (2005); they are built extending of an abstract argumentation framework (Dung, 1995) with a set of dialectical constraints. This formal construct will be used in the following section for the definition of *dialectical explanations* (δ-explanations).

It is important to note that the aim of this paper is not to offer an alternative approach to the semantics of argumentation. Instead, an argumentation theory provides an structure with two slots: one for an argumentation framework and another for a set of dialectical constraints. The focus of the paper lies in the explanation formalism that will be applied to different argumentation

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

systems together with a chosen associated semantics, just by defining an appropriate set of dialectical constraints.

Abstract argumentation frameworks are formalisms for modeling defeasible argumentation in which some components remain unspecified (Dung, 1995; Jakobovits & Vermeir, 1999; Rahwan & Simari, 2009); in them an argument is considered an abstract entity without any specific structure. Roughly speaking, an argument is anything that may attack to or be attacked by another argument (Baroni & Giacomin, 2009). Thus, formally an argumentation framework Φ is a pair $\langle \mathsf{Args}, \mathsf{R} \rangle$, where Args is a finite set of arguments and R is a binary relation between arguments such that $\mathsf{R} \subseteq \mathsf{Args} \times \mathsf{Args}$. The notation $(\mathcal{A}, \mathcal{B}) \in \mathsf{R}$ (or, equivalently, $\mathcal{A}\ \mathsf{R}\ \mathcal{B}$) means that $\mathcal{A}$ *attacks* $\mathcal{B}$, or $\mathcal{A}$ *defeats* $\mathcal{B}$ (Dung, 1995).

Example 2.1. Consider the argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1} \rangle$, where: $\mathsf{Args}_{2.1} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{B}_1, \mathcal{B}_2, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$ and $\mathsf{R}_{2.1} = \{(\mathcal{A}_1, \mathcal{A}_5), (\mathcal{A}_1, \mathcal{A}_3), (\mathcal{A}_2, \mathcal{A}_5), (\mathcal{A}_3, \mathcal{A}_5), (\mathcal{A}_4, \mathcal{A}_1), (\mathcal{B}_1, \mathcal{B}_2), (\mathcal{B}_2, \mathcal{B}_1), (\mathcal{A}_2, \mathcal{C}_1), (\mathcal{A}_3, \mathcal{C}_2), (\mathcal{C}_1, \mathcal{C}_2), (\mathcal{C}_2, \mathcal{C}_1), (\mathcal{C}_1, \mathcal{C}_3), (\mathcal{C}_3, \mathcal{C}_4), (\mathcal{C}_4, \mathcal{C}_5), (\mathcal{C}_5, \mathcal{C}_3)\}$.

An argumentation framework $\langle \mathsf{Args}, \mathsf{R} \rangle$ is graphically depicted as a directed graph; in it every node stands for an argument in Args, and there is an arc $(\mathcal{A}_i, \mathcal{A}_j)$ whenever $(\mathcal{A}_i, \mathcal{A}_j) \in \mathsf{R}$. For instance, Fig. 2 depicts the argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1} \rangle$.

In the literature of abstract argumentation frameworks, several semantics have been defined and studied (Amgoud et al., 2002; Baroni & Giacomin, 2009; Baroni, Giacomin, & Guida, 2005; Jakobovits & Vermeir, 1999). In particular, in this paper we will focus on skeptical semantics for argumentation that is based on a dialectical analysis of argumentation trees. Dialectical analysis in argumentation involves the exploration of a search space 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 search space by generating tree-like structures which are referred to as argument trees (Besnard & Hunter, 2001) or as dialectical trees (Chesñevar et al., 2004; García & Simari, 2004) in the literature.

Given an argumentation framework $\langle \mathsf{Args}, \mathsf{R} \rangle$, to establish if one particular argument $\mathcal{A} \in \mathsf{Args}$ is warranted, each argument $\mathcal{B}$ that defeats $\mathcal{A}$ must be considered. It is clear that the warrant status of $\mathcal{A}$ will depend on the status of these $\mathcal{B}$s. Since each $\mathcal{B}$ can be also defeated, and its defeater can be defeated, and so on, a sequence of arguments can arise where, apart from the last one of the sequence, each one is defeated by the subsequent. For instance, in Example 2.1 $\mathcal{C}_5$ is defeated by $\mathcal{C}_4$, which in turn is defeated by $\mathcal{C}_3$, $\mathcal{C}_1$ defeats $\mathcal{C}_3$, and $\mathcal{A}_2$ defeats $\mathcal{C}_1$. Therefore, the acceptance status of $\mathcal{C}_5$ will depend on the acceptance status of $\mathcal{C}_4, \mathcal{C}_3, \mathcal{C}_1$, and $\mathcal{A}_2$. Before introducing argumentation theories, we will present the notion of a sequence of defeating arguments and other auxiliary definitions.

Definition 2.1 (*Argumentation line*). Let $\Phi = \langle \mathsf{Args}, \mathsf{R} \rangle$ be an abstract argumentation framework. An *argumentation line* rooted in $\mathcal{A}_1$ is either a singleton sequence $[\mathcal{A}_1]$ ($\mathcal{A}_1 \in \mathsf{Args}$) or any finite sequence of arguments $[\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ ($n \geqslant 2$), such that $(\mathcal{A}_i, \mathcal{A}_{i-1}) \in \mathsf{R}$, for $1 < i \leqslant n$. We will write $\mathfrak{Lines}_\Phi$ to denote the set of all argumentation lines in Φ.

Consider the argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1} \rangle$ from Example 2.1. Different argumentation lines rooted in $\mathcal{A}_5$ can be obtained, namely: $\lambda_a = [\mathcal{A}_5, \mathcal{A}_1, \mathcal{A}_4]$, $\lambda_b = [\mathcal{A}_5, \mathcal{A}_2]$, $\lambda_c = [\mathcal{A}_5, \mathcal{A}_3]$, $\lambda_d = [\mathcal{A}_5, \mathcal{A}_1]$ and $\lambda_e = [\mathcal{A}_5, \mathcal{A}_3, \mathcal{A}_1, \mathcal{A}_4]$. Note that although $(\mathcal{A}_1, \mathcal{A}_5) \in \mathsf{R}_{2.1}$, in $\lambda_e \mathcal{A}_1$ attacks $\mathcal{A}_3$, and hence defends $\mathcal{A}_5$. An infinite number of argumentation lines rooted in $\mathcal{B}_1$ can be obtained: $\lambda_1 = [\mathcal{B}_1]$, $\lambda_2 = [\mathcal{B}_1, \mathcal{B}_2]$, $\lambda_3 = [\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_1]$, $\lambda_4 = [\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_1, \mathcal{B}_2]$, *etc.*; this is because $(\mathcal{B}_1, \mathcal{B}_2)$ and $(\mathcal{B}_2, \mathcal{B}_1)$ are members of $\mathsf{R}_{2.1}$ thus introducing a cycle in the associated graph (see Fig. 2). Argumentation lines define a domain over which different constraints can be defined. As such constraints are related to sequences which resemble an argumentation dialog between two parties, we call them *dialectical constraints*.

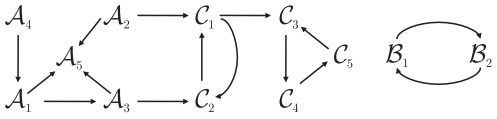

Fig. 2. Argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1} \rangle$ (Example 2.1).

Definition 2.2 (*Dialectical constraint*). Let $\Phi = \langle \mathsf{Args}, \mathsf{R} \rangle$ be an abstract argumentation framework. A *dialectical constraint* $\mathbf{C}$ *in the context of* Φ is any function $\mathbf{C} : \mathfrak{Lines}_\Phi \mapsto \{\mathsf{True}, \mathsf{False}\}$. A given argument sequence $\lambda \in \mathfrak{Lines}_\Phi$ satisfies $\mathbf{C}$ in Φ when $\mathbf{C}(\lambda) = \mathit{True}$.

Then, as we already mentioned, an argumentation theory is defined by combining an argumentation framework with a particular set of dialectical constraints. Formally:

Definition 2.3 (*Argumentation theory*). An *argumentation theory T* (or just a *theory*) is a pair $(\Phi, \mathbf{DC})$, where Φ is an abstract argumentation framework, and $\mathbf{DC} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$ is a finite (possibly empty) set of *dialectical constraints*.

Therefore, in an argumentation theory, any set of dialectical constraints can be used (*e.g.*, the empty set). Thus, different argumentation systems can be obtained (with different semantics) by defining the appropriate set of dialectical constraints. It must be noted that a full formalization for dialectical constraints is outside the scope of this work. As they may vary from one particular argumentation framework to another, they are included as a parameter in an argumentation theory.[1] Below, some illustrative examples of dialectical constraints are shown. For instance, dialectical constraints can be used to impose conditions on argumentation lines to be considered rationally *acceptable*. Such conditions are usually defined by disallowing certain moves which might lead to fallacious situations. For example, it is forbidden to repeat the same argument in an argumentation line (Example 2.2), or it is not allowed that parties contradict themselves when advancing arguments (Example 2.3), or it is required that the attack relation has to be such that there is no pair of arguments that attack each other, achieving in this way that no argument in an argumentation line can be attacked by its successor (see Example 2.4). Several examples follow.

Example 2.2 (*Dialectical constraint* $\mathbf{C}_{nc}$*: non-circularity*). A dialectical constraint for preventing a circular argumentation line can be defined as follows. Consider $\lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$, then $\mathbf{C}_{nc}(\lambda) = \mathsf{True}$ iff $n = 1$ or, for each $\mathcal{A}_k$ in λ ($2 \leqslant k \leqslant n$), it holds that $\mathcal{A}_k$ does not appear in $[\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{k-1}]$; conversely, $\mathbf{C}_{nc}(\lambda) = \mathsf{False}$ iff there exist $\mathcal{A}_v$ ($2 \leqslant v \leqslant n$) such that $\mathcal{A}_v = \mathcal{A}_i$ ($1 \leqslant i \leqslant v - 1$). Thus, the function $\mathbf{C}_{nc}$ returns True iff every argument appears only once in an argumentation line (and false otherwise). Consider for instance the argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1} \rangle$ of Example 2.1. In that case, as already remarked above, an infinite number of argumentation lines rooted in $\mathcal{B}_1$ can be obtained, *e.g.*, $\lambda_1 = [\mathcal{B}_1]$, $\lambda_2 = [\mathcal{B}_1, \mathcal{B}_2]$, $\lambda_3 = [\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_1]$, and $\lambda_4 = [\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_1, \mathcal{B}_2]$. Nevertheless, only the first two satisfy $\mathbf{C}_{nc}$: $\mathbf{C}_{nc}(\lambda_1) = \mathsf{True}$, $\mathbf{C}_{nc}(\lambda_2) = \mathsf{True}$, and $\mathbf{C}_{nc}(\lambda_3) = \mathsf{False}$, and $\mathbf{C}_{nc}(\lambda_4) = \mathsf{False}$.

Example 2.3 (*Dialectical constraint* $\mathbf{C}_{co}$*: commitment*). Consider a line $\lambda = [\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_n]$; in it $\mathcal{A}_3$ is attacking $\mathcal{A}_2$ thus giving support for $\mathcal{A}_1$. Therefore, $\mathcal{A}_3$ should not contradict $\mathcal{A}_1$. The same holds for $\mathcal{A}_2$ and $\mathcal{A}_4$, and for $\mathcal{A}_1$ and $\mathcal{A}_5$. Observe that arguments in odd positions represent *supporting* arguments for $\mathcal{A}_1$, and that arguments in even positions represent *interfering* arguments

[1] A similar approach is adopted in Kakas and Toni (1999), where different characterizations of constraints give rise to different Logic Programming semantics.

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

for $\mathcal{A}_1$ (indirect attack). A dialectical constraint can be defined to impose that supporting (respectively interfering) arguments should not attack each other. Consider the sets $\lambda_S = \{\mathcal{A}_i \in \lambda \mid \mathcal{A}_i$ appears in an odd position in $\lambda\}$ and $\lambda_I = \{\mathcal{A}_j \in \lambda \mid \mathcal{A}_j$ appears in an even position in $\lambda\}$. We define $\mathbf{C}_{co}(\lambda) = \mathsf{True}$ iff for any pair of arguments $\mathcal{A}_k$ and $\mathcal{A}_t$ of λ_S, $(\mathcal{A}_k, \mathcal{A}_t) \notin \mathsf{R}$ and for any pair of arguments $\mathcal{A}_g$ and $\mathcal{A}_h$ of $\lambda_I, (\mathcal{A}_g, \mathcal{A}_h) \notin \mathsf{R}$. Otherwise, $\mathbf{C}_{co}(\lambda) = \mathsf{False}$. Consider the argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1}\rangle$ of Example 2.1 where several lines rooted in $\mathcal{A}_5$ can be obtained: $\lambda_a = [\mathcal{A}_5, \mathcal{A}_1, \mathcal{A}_4]$ and $\lambda_e = [\mathcal{A}_5, \mathcal{A}_3, \mathcal{A}_1, \mathcal{A}_4]$. Note that $\mathbf{C}_{co}(\lambda_a) = \mathsf{True}$ and $\mathbf{C}_{co}(\lambda_d) = \mathsf{False}$ since $(\mathcal{A}_1, \mathcal{A}_5) \in \mathsf{R}_{2.1}$.

Example 2.4 (*Dialectical constraint* $\mathbf{C}_{prop}$*: proper attack*). Consider the argumentation line $\lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ from an argumentation framework $\langle \mathsf{Args}, \mathsf{R}\rangle$. We define $\mathbf{C}_{prop}(\lambda) = \mathsf{True}$ iff $n = 1$ or, for each $\mathcal{A}_k$ in λ $(2 \leqslant k \leqslant n)$, it holds that $(\mathcal{A}_{k-1}, \mathcal{A}_k) \notin \mathsf{R}$; conversely, $\mathbf{C}_{prop}(\lambda) = \mathsf{False}$ iff there exist $\mathcal{A}_v$ $(2 \leqslant v \leqslant n)$ such that $(\mathcal{A}_{v-1}, \mathcal{A}_v) \in \mathsf{R}$;. Thus, the function $\mathbf{C}_{prop}$ returns True if every argument in the line attacks is predecessor but no argument in the line attacks its successor.

The definition of acceptable argumentation line corresponds to the use of these criteria, *i.e.*, an argumentation line will be acceptable when complies with the constrains the theory requires.

Definition 2.4 (*Acceptable argumentation line*). Let $T = (\Phi, \mathbf{DC})$ be an argumentation theory. An argumentation line $\lambda \in \mathfrak{Lines}_\Phi$ is *acceptable* with respect to T iff for every $\mathbf{C} \in \mathbf{DC}$, $\mathbf{C}(\lambda) = \mathsf{True}$.

Example 2.5. Consider the argumentation framework $\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1}\rangle$ defined in Example 2.1 and the dialectical constraints $\mathbf{C}_{nc}$ and $\mathbf{C}_{co}$ of Examples 2.2 and 2.3. Then, with these elements we can define the argumentation theory $T_{2.5} = (\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1}\rangle, \{\mathbf{C}_{nc}, \mathbf{C}_{co}\})$. The argumentation line $\lambda_a = [\mathcal{A}_5, \mathcal{A}_1, \mathcal{A}_4]$ is acceptable with respect to the argumentation theory $T_{2.5}$ because $\mathbf{C}_{nc}(\lambda_a) = \mathsf{True}$ and $\mathbf{C}_{co}(\lambda_a) = \mathsf{True}$. Observe that $\lambda_b = [\mathcal{B}_1, \mathcal{B}_2]$ is acceptable whereas $\lambda_c = [\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_1]$ is not acceptable with respect to $T_{2.5}$ because $\mathbf{C}_{nc}(\lambda_c) = \mathsf{False}$. The line $\lambda_d = [\mathcal{A}_5, \mathcal{A}_3, \mathcal{A}_1, \mathcal{A}_4]$ is not acceptable with respect to $T_{2.5}$ since $\mathbf{C}_{co}(\lambda_d) = \mathsf{False}$ because $(\mathcal{A}_1, \mathcal{A}_5) \in \mathsf{R}_{2.1}$. Consider now the argumentation theory $T_{2.5b} = (\langle\{\mathcal{A}, \mathcal{B}, \mathcal{C}\}, \{(\mathcal{B}, \mathcal{A}), (\mathcal{C}, \mathcal{A}), (\mathcal{B}, \mathcal{C}), (\mathcal{C}, \mathcal{B})\}\rangle, \{\mathbf{C}_{nc}, \mathbf{C}_{co}\})$. Here, $\mathcal{B}$ and $\mathcal{C}$ attack each other, and both attack $\mathcal{A}$. The lines $[\mathcal{A}, \mathcal{B}, \mathcal{C}]$ and $[\mathcal{A}, \mathcal{C}, \mathcal{B}]$ are not acceptable with respect to $T_{2.5b}$ because they do not satisfy $\mathbf{C}_{co}$. Nevertheless, $[\mathcal{A}, \mathcal{B}]$ and $[\mathcal{A}, \mathcal{C}]$ are both acceptable with respect to $T_{2.5b}$.

Remark 2.1. Observe that given a theory T, if an argumentation line λ satisfies $\mathbf{C}_{nc}$ then any subsequence of λ also satisfies $\mathbf{C}_{nc}$ and the same holds for $\mathbf{C}_{co}$. The justification can be found in Appendix A.

As we have stated before, in this paper we will focus on skeptical semantics for argumentation that is based on a dialectical analysis of argumentation trees. Hence, we will adapt the definition of dialectical tree (García, Chesñevar, & Simari, 1993; Simari, Chesñevar, & García, 1994) to this abstract framework.

Definition 2.5 (*Dialectical tree*). Let $T = (\Phi, \mathbf{DC})$ be an argumentation theory, where $\Phi = \langle \mathsf{Args}, \mathsf{R}\rangle$, and let $\mathcal{A}_1$ be an argument in Args. Let $Acc(\mathcal{A}_1) \subseteq \mathfrak{Lines}_\Phi$ be the set of all the argumentation lines rooted in $\mathcal{A}_1$ that are acceptable with respect to T. A dialectical tree for $\mathcal{A}_1$ in T (denoted $\mathcal{T}_{\mathcal{A}_1}$) is a tree of arguments from Args such that:

1. The root of the tree is labeled with $\mathcal{A}_1$.
2. Let N be node of the tree labeled $\mathcal{A}_n$, and $[\mathcal{A}_1, \ldots, \mathcal{A}_n]$ be the sequence of labels of the path from the root to N. Let $Attackers = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k\}$ be the set of all the arguments such that $(\mathcal{B}_i, \mathcal{A}_n) \in \mathsf{R}$ $(1 \leqslant i \leqslant k)$. For each argument $\mathcal{B}_i$ such that the argumentation line $\lambda_i = [\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{B}_i]$ is acceptable with respect to the **DC** in T (*i.e.*, $\lambda_i \in Acc(\mathcal{A}_1)$), the node N has a child N_i labeled $\mathcal{B}_i$. If $Attackers = \emptyset$ or there is no $\mathcal{B}_i$ such that $\lambda_i \in Acc(\mathcal{A}_1)$, then N is a leaf.

Consider a dialectical tree $\mathcal{T}_{\mathcal{A}_1}$ in a theory T, and let $paths(\mathcal{T}_{\mathcal{A}_1})$ be the set of all paths from the root of $\mathcal{T}_{\mathcal{A}_1}$ to a leaf. Observe that every element of $paths(\mathcal{T}_{\mathcal{A}_1})$ is an acceptable argumentation line $\lambda_i = [\mathcal{A}_1, \ldots, \mathcal{A}_n]$ w.r.t. **DC** in T. Also note that λ_i is exhaustive, in the sense that there cannot exist an argumentation line $\lambda_i' = [\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}]$ acceptable w.r.t. **DC** in T, because in that case $\mathcal{A}_n$ will not be a leaf in that tree.

Definition 2.6 (*Exhaustive acceptable argumentation line*). Let $T = (\Phi, \mathbf{DC})$ be an argumentation theory. An argumentation line $\lambda \in \mathfrak{Lines}_\Phi$ is *exhaustive* with respect to T iff λ is acceptable with respect to **DC** in T and there is no acceptable argumentation line $\lambda' \in \mathfrak{Lines}_\Phi$ extending λ.

Thus, dialectical trees can be characterized by the set of exhaustive acceptable argumentation lines; this set will be called *exhaustive bundle set* and it corresponds to all the paths from the root to a leaf in $\mathcal{T}_{\mathcal{A}_1}$.

Definition 2.7 (*Exhaustive bundle set*). Let $T = (\Phi, \mathbf{DC})$ be an argumentation theory, where $\Phi = \langle \mathsf{Args}, \mathsf{R}\rangle$, $\mathcal{A} \in \mathsf{Args}$. The set $ebundle(\mathcal{A}) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of all the argumentation lines rooted in $\mathcal{A}$ that are acceptable with respect to **DC** in T, is called an *exhaustive bundle set* for $\mathcal{A}$ in T if and only if all λ_i $(1 \leqslant i \leqslant n)$ are exhaustive.

Example 2.6. Consider $T_{2.5} = (\langle \mathsf{Args}_{2.1}, \mathsf{R}_{2.1}\rangle, \{\mathbf{C}_{nc}, \mathbf{C}_{co}\})$ (introduced in Example 2.5). A dialectical tree for $\mathcal{A}_5$ is shown in Fig. 3(a). Observe that $ebundle(\mathcal{A}_5) = \{[\mathcal{A}_5, \mathcal{A}_1, \mathcal{A}_4], [\mathcal{A}_5, \mathcal{A}_3], [\mathcal{A}_5, \mathcal{A}_2]\}$.

Definition 2.5 shows how to build a dialectical tree. 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. For instance, Fig. 3(c) and (d) shows two different dialectical tree that are also associated with the bundle set $ebundle(\mathcal{A}_5)$ of Example 2.6. This observation is formalized by introducing the following relation which can be trivially shown to be an equivalence relation.

Definition 2.8. 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 $paths(\mathcal{T}_\mathcal{A}) = paths(\mathcal{T}'_\mathcal{A}) = ebundle(\mathcal{A})$.

As shown in the next proposition, a dialectical tree for any argument $\mathcal{A}$ can be proven to be unique up to $\equiv_\tau$-equivalence.

Proposition 2.1. *Let T be an argumentation theory. For any argument $\mathcal{A}$ in T there is a unique dialectical tree $\mathcal{T}_\mathcal{A}$ in T up to $\equiv_\tau$-equivalence.*

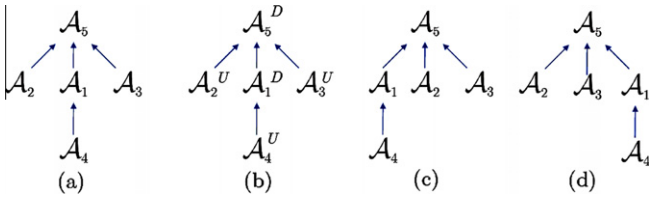

Fig. 3. (a) Dialectical tree $\mathcal{T}_{\mathcal{A}_5}$ of Example 2.6; (b)$\mathcal{T}_{\mathcal{A}_5}$ after applying and–or marking; (c, d) two other exhaustive dialectical trees belonging to the same equivalence class of $\mathcal{T}_{\mathcal{A}_5}$.

Proof. See Appendix A. □

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* or *labeling* criterion provides a definition of such acceptance criterion. Connections between defeat status assignments and extensions in Dung's argumentation frameworks have been firstly investigated by Verheij (1996); see also Caminada (2006), Verheij (2007) and Baroni and Giacomin (2009), Baroni, Caminada, and Giacomin (2011) for an extensive account of labeling and the corresponding references. Our approach is based in similar ideas presented in García et al. (1993), Simari et al. (1994) and García and Simari (2004). Formally:

Definition 2.9 (*Marking criterion*). Let T be an argumentation theory and $\mathfrak{Tree}_T$ all the dialectical trees that can be obtained from a theory T. Let $\mathfrak{Tree}^*_T$ be the set of all the dialectical trees where each node is marked D (defeated) or U (undefeated). A marking criterion for T is a function $marking : \mathfrak{Tree}_T \mapsto \mathfrak{Tree}^*_T$ such that given a dialectical tree $\mathcal{T}_\mathcal{A}$ for an argument $\mathcal{A}$, it returns a marked dialectical tree, denoted $\mathcal{T}^*_\mathcal{A}$, where each node of $\mathcal{T}_\mathcal{A}$ has been assigned a mark D or U. The associated function $rootmark : \mathfrak{Tree}^*_T \mapsto \{D, U\}$ returns the mark of the root node of $\mathcal{T}^*_\mathcal{A}$.

Example 2.7. A marking criterion can be defined as follows. Given a dialectical tree, leaves are marked U. Then, for every inner node N of the tree such that all its children have been marked, the mark for N is D if there is at least one child of N marked as U, and the mark for N is U if all its children are marked as D. Fig. 3(b) shows the dialectical tree $\mathcal{T}_{\mathcal{A}_5}$ of Example 2.6 after applying this marking criterion. This is a natural criterion for marking a tree and it corresponds to the grounded semantics defined for abstract argumentation frameworks (Dung, 1995).

The following definition captures the idea of an argument being ultimately undefeated in an argumentation theory. In the literature, arguments that obtain that status usually are said to be accepted, justified, or warranted.

Definition 2.10 (*Warrant*). Let T be an argumentation theory and $marking(\cdot)$ a marking criterion for T. An argument $\mathcal{A}$ is a *warrant* for its claim with respect to the marking criterion iff the marked dialectical tree $\mathcal{T}^*_\mathcal{A} = marking(\mathcal{T}_\mathcal{A})$ is such that $rootmark(\mathcal{T}^*_\mathcal{A}) = U$. If an argument $\mathcal{A}$ is a warrant for its claim, then we will also say that its claim is warranted (or that there is a warrant for its claim), and that the marked dialectical tree $\mathcal{T}^*_\mathcal{A}$ exhibits the warrant status of $\mathcal{A}$'s claim.

In the following section, marked dialectical trees will be the basis of the notion of *dialectical explanation* in abstract argumentation theories. We will use the marking criterion introduced in Example 2.7 in the examples.

3. Explanations in abstract argumentation frameworks

In this work, given that an argument has obtained the status of being warranted in an argumentation theory, the purpose of a dialectical explanation will be to *transfer the understanding* of how an argument obtains that status. Consequently, a δ-explanation will consist of a structure that reflects the analysis that was carried out in order to obtain such status, and it will contain those arguments and counterarguments that are considered as part of this analysis.

Suppose that there is a known set of arguments, and we receive the information that a particular argument in that set is warranted whereas others are not; we may want to know *why* that conclusion was drawn. For instance, consider again the opera example presented in Section 1 where argument $\mathcal{O}_{show}$ represents "Bob is an opera aficionado and there is an opera show tonight"; $\mathcal{O}_{get}$ represents "today is Bob's birthday and he usually gets together with friends"; $\mathcal{O}_{friends}$ represents "Bob usually goes to the opera house with friends"; and $\mathcal{O}_{baby}$ represents "today Bob's best friend is coming with her baby because it's his birthday, and is not a good idea to attend the opera with a baby". If a system returns the information that $\mathcal{O}_{get}$ is a warranted argument or that $\mathcal{O}_{show}$ is not warranted, then we may want to know the reasons supporting that these conclusions were drawn.

In this section, we will define δ-explanation that will provide all the information (arguments) considered by the system for warranting a claim and the analysis (dialectical trees) that has performed. An argumentation theory with the arguments of the opera scenario is introduced in the example below.

Example 3.1. Consider the set of arguments $\mathsf{Args}_{3.1} = \{\mathcal{O}_{show}, \mathcal{O}_{get}, \mathcal{O}_{friends}, \mathcal{O}_{baby}\}$, where $\mathcal{O}_{baby}$ defeats (or attacks) both $\mathcal{O}_{show}$ and $\mathcal{O}_{friends}, \mathcal{O}_{friends}$ in turn defeats $\mathcal{O}_{get}$, and $\mathcal{O}_{get}$ and $\mathcal{O}_{show}$ defeat each other. The following defeat relation reflects these observations $\mathsf{R}_{3.1} = \{(\mathcal{O}_{get}, \mathcal{O}_{show}), (\mathcal{O}_{show}, \mathcal{O}_{get}), (\mathcal{O}_{friends}, \mathcal{O}_{get}), (\mathcal{O}_{baby}, \mathcal{O}_{friends}), (\mathcal{O}_{baby}, \mathcal{O}_{show})\}$. The argumentation theory $T_{3.1} = (\langle \mathsf{Args}_{3.1}, \mathsf{R}_{3.1} \rangle, \mathbf{DC})$ can be defined where $\mathbf{DC} = \{\mathbf{C}_{nc}, \mathbf{C}_{co}\}$ are the dialectical constraints defined in Examples 2.2 and 2.3, respectively. Fig. 4 depicts (a) the graph of the argumentation framework $\langle \mathsf{Args}_{3.1}, \mathsf{R}_{3.1} \rangle$ and (b)–(e) the marked dialectical trees for arguments in $\mathsf{Args}_{3.1}$. Here, the marking criterion of Example 2.7 is used.

Although Fig. 4 (a) shows all the existing arguments and the defeat relation defined over the set of arguments, the graph does not show the analysis performed to state if an argument has to be accepted as warranted. For example, $\mathcal{O}_{get}$ is defeated by $\mathcal{O}_{friends}$ but $\mathcal{O}_{friends}$ is in turn defeated by $\mathcal{O}_{baby}$, thus, reinstating $\mathcal{O}_{get}$. The dialectical process for warranting a claim involves finding the arguments that either support or interfere with that claim. As we discussed in the previous section, these arguments are connected through the defeat relation and their interrelationships in terms of this defeat

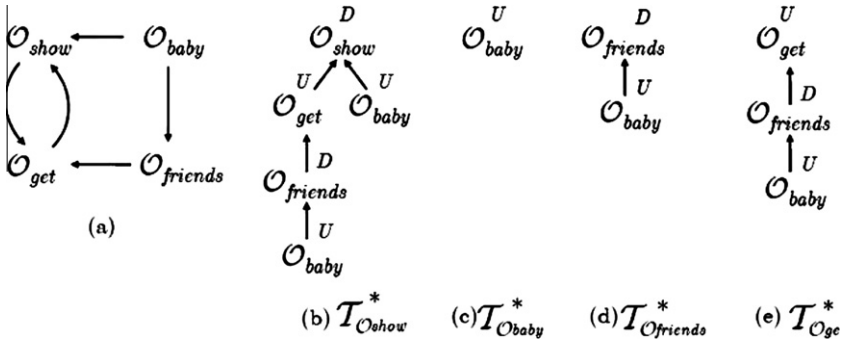

Fig. 4. (a) Argumentation framework of Example 3.1. (b–e) Marked dialectical trees.

relation, together with the dialectical constraints, can be conceptualized through dialectical trees.

Remark 3.1. In the examples of this paper we will use the marking criterion introduced in Example 2.7, akin to grounded semantics in abstract argumentation. Using this criterion to mark a dialectical tree, an argument mark depends on the position in which such argument appears in an argumentation line. Since it is possible that an argument appears in more than one argumentation line, if this argument appears in two different lines, the mark for each occurrence could be different.

Conceptually, an argument supports a claim by reasoning from certain premises. The formalism for abstract argumentation described in the previous section reduces arguments to abstract entities without any reference to the corresponding claim. However, for concrete applications it is necessary to recover the idea that every argument has an associated claim, accordingly we will assume a function $\mathsf{claim}(\mathcal{A})$ that will provide the claim of the argument $\mathcal{A}$. For instance, in the scenario described above, $\mathsf{claim}(\mathcal{O}_{show})$ is "go to the opera" and $\mathsf{claim}(\mathcal{O}_{baby})$ is "do not go to the opera". If an argument $\mathcal{A}$ is shown to be a warrant for a claim Q through the dialectical tree $\mathcal{T}^*_{\mathcal{A}}$ (Definition 2.10), then we will also say that Q is warranted (or that there is a warrant for Q), and that $\mathcal{T}^*_{\mathcal{A}}$ exhibits $\mathcal{A}$ as a warrant for claim Q.

Remark 3.2. The notation $\overline{Q}$ will represent a claim that contradicts the claim Q. For instance, $\overline{Q}$ can contradict Q if $\overline{Q}$ is the complement of Q with respect to strict negation. In the opera example, if Q represents "go to the opera", then $\overline{Q}$ represents "do not go to the opera". That is, $\mathsf{claim}(\mathcal{O}_{show})$ and $\mathsf{claim}(\mathcal{O}_{baby})$ contradict each other. We assume that $\overline{\overline{Q}} = Q$, that is, if $\overline{Q}$ contradicts Q then it also holds that Q contradicts $\overline{Q}$.

In the argumentation process, when a claim is being analyzed it is necessary to consider the arguments that support the claim but also is required to ponder the arguments contradicting it. For instance, in Example 3.1 arguments for going to the opera and arguments for not going to the opera should be considered to decide whether to go or not to go. It is clear that there could be more than one argument that supports a particular claim and also several arguments against the same claim. Therefore, to understand why an argument for a claim receives a particular status is not enough to look at the analysis performed over the arguments supporting the claim; it is necessary also to consider the analysis done over the ones that contradict it.

In our approach an explanation will consist of a structure that includes those marked dialectical trees that are considered for establishing such status. If only one argument $\mathcal{A}$ for a claim Q exists, then there will be a unique tree $\mathcal{T}^*_{\mathcal{A}}$ (Proposition 2.1) and this tree will be included in the explanation of the warrant status of Q. Nevertheless, given a claim Q there could be several different arguments $\mathcal{A}_1, \ldots, \mathcal{A}_n$ supporting Q, and each $\mathcal{A}_i$ will generate its corresponding (different) dialectical tree. Hence, in this case an explanation will include several dialectical trees. The following proposition establish that there is as many exhaustive dialectical trees related to a claim Q as different arguments with Q as their claim. Clearly, its proof is straightforward from Proposition 2.1.

Proposition 3.1. *Let T be an argumentation theory and Q a claim. If there are n arguments $\mathcal{A}_1, \ldots, \mathcal{A}_n$ for claim Q then there exist n different exhaustive dialectical trees $\mathcal{T}^*_{\mathcal{A}_1}, \ldots, \mathcal{T}^*_{\mathcal{A}_n}$ (up to equivalences with respect to $\equiv_\tau$ as introduced in Definition 2.8).*

Proof. If there are n arguments for Q then each argument has a unique dialectical tree (Proposition 2.1). It is clear that for each pair of arguments $\mathcal{A}_i$ and $\mathcal{A}_j$ $(1 \leqslant i, j \leqslant n)$ the dialectical trees $\mathcal{T}^*_{\mathcal{A}_i}$ and $\mathcal{T}^*_{\mathcal{A}_j}$ are different because their roots are different. Therefore, there are n different dialectical trees. □

The following example illustrates the general case where there are several arguments for the same claim Q and several arguments supporting $\overline{Q}$. Each one of these arguments will generate a different dialectical tree and some of these dialectical trees may have the root marked *U*.

Example 3.2. Consider the argumentation theory $T_{3.2} = (\langle \mathsf{Args}_{3.2}, \mathsf{R}_{3.2} \rangle, \mathbf{DC})$, where the set of arguments is $\mathsf{Args}_{3.2} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{X}_1, \mathcal{B}_2, \mathcal{F}_1, \mathcal{C}_1, \mathcal{D}_1, \mathcal{D}_2, \mathcal{E}_1, \mathcal{E}_2\}$, the defeat relation is $\mathsf{R}_{3.2} = \{(\mathcal{X}_1, \mathcal{A}_1), (\mathcal{X}_1, \mathcal{A}_2), (\mathcal{A}_4, \mathcal{X}_1), (\mathcal{B}_2, \mathcal{A}_2), (\mathcal{F}_1, \mathcal{A}_3), (\mathcal{C}_1, \mathcal{F}_1), (\mathcal{D}_1, \mathcal{D}_2), (\mathcal{D}_2, \mathcal{D}_1), (\mathcal{E}_2, \mathcal{E}_1)\}$, and the dialectical constraints $\mathbf{DC} = \{\mathbf{C}_{nc}, \mathbf{C}_{co}\}$ are the ones defined in Examples 2.2 and 2.3, respectively. Assume that $\mathsf{claim}(\mathcal{A}_1) = \mathsf{claim}(\mathcal{A}_2) = \mathsf{claim}(\mathcal{A}_3) = \mathsf{claim}(\mathcal{A}_4) = a$, $\mathsf{claim}(\mathcal{X}_1) = \bar{a}$, $\mathsf{claim}(\mathcal{B}_2) = b$, $\mathsf{claim}(\mathcal{F}_1) = \bar{f}$, $\mathsf{claim}(\mathcal{C}_1) = \bar{c}$, $\mathsf{claim}(\mathcal{D}_1) = d$, $\mathsf{claim}(\mathcal{D}_2) = \bar{d}$, $\mathsf{claim}(\mathcal{E}_1) = e$, and $\mathsf{claim}(\mathcal{E}_2) = \bar{e}$. In this theory, there are four arguments for claim a, one argument for claims $\bar{a}, \bar{f}, d, \bar{d}, e$, and $\bar{e}$, and no argument for claim f. Fig. 5 shows the marked dialectical trees $\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_2}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}$ and $\mathcal{T}^*_{\mathcal{X}_1}$. The root of $\mathcal{T}^*_{\mathcal{A}_2}$ is marked *D*, whereas the roots of $\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}$ and $\mathcal{T}^*_{\mathcal{A}_4}$ are marked *U* and each one provides a warrant for claim a. Observe that the only argument with claim $\bar{a}$ is $\mathcal{X}_1$ and the root of $\mathcal{T}^*_{\mathcal{X}_1}$ is marked *D*. Hence, there is no warrant for $\bar{a}$. Fig. 6 shows the marked dialectical trees for the arguments $\mathcal{D}_1, \mathcal{D}_2, \mathcal{E}_1, \mathcal{E}_2, \mathcal{F}_1$ and $\mathcal{C}_1$.

A dialectical explanation for a claim Q will contain dialectical trees for arguments that support Q and also dialectical trees for arguments that support $\overline{Q}$ (*i.e.*, arguments for a claim that is in direct contradiction with Q). Observe that there can be either several arguments for Q, only one argument for Q, or no arguments for Q; and the same is true for $\overline{Q}$. Hence, nine combinations arise. For instance, it may happen that there is no argument for a claim Q but there are several arguments for $\overline{Q}$. In this case, although there is no warrant for Q, there exists an explanation that will include the dialectical trees for those arguments that support $\overline{Q}$. Example 3.2 introduces an argumentation theory that shows some of the combinations mentioned above.

The following definition characterizes two distinguished sets of marked dialectical trees that will be used in an explanation.

Definition 3.1 (*Dialectical tree sets*). Let $T = (\langle \mathsf{Args}, \mathsf{R} \rangle, \mathbf{DC})$ be an argumentation theory and Q a claim. Let $\mathbb{T}^*(Q) = \{\mathcal{T}^*_{\mathcal{A}} | \mathcal{A} \in \mathsf{Args} \text{ and } \mathsf{claim}(\mathcal{A}) = Q\}$, the sets $\mathbb{T}^*_U(Q) \subseteq \mathbb{T}^*(Q)$, and $\mathbb{T}^*_D(Q) \subseteq \mathbb{T}^*(Q) \cup \mathbb{T}^*(\overline{Q})$ are defined as follows:

$$\mathbb{T}^*_U(Q) = \{\mathcal{T}^* \in \mathbb{T}^*(Q) \mid \mathit{rootmark}(\mathcal{T}^*) = U\},$$

$$\mathbb{T}^*_D(Q) = \{\mathcal{T}^* \in (\mathbb{T}^*(Q) \cup \mathbb{T}^*(\overline{Q})) \mid \mathit{rootmark}(\mathcal{T}^*) = D\}.$$

That is, the set $\mathbb{T}^*_U(Q)$ includes all the dialectical trees for arguments that support Q where the root is marked as undefeated. Therefore, $\mathbb{T}^*_U(Q)$ contains all the dialectical trees that provide a warrant for Q. The set $\mathbb{T}^*_U(Q)$ is empty if no warrant for Q exists. Observe that the set $\mathbb{T}^*_U(\overline{Q})$ will include all the dialectical trees that provide a warrant for $\overline{Q}$ (if there are any). The set $\mathbb{T}^*_D(Q)$ includes all the dialectical trees for Q and $\overline{Q}$ whose roots are marked as

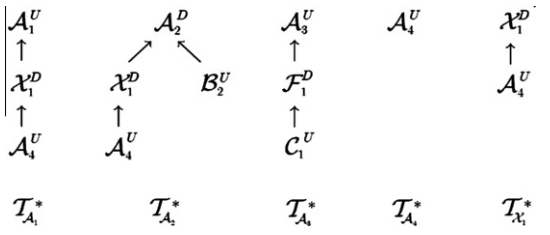

Fig. 5. Marked dialectical trees for arguments of Example 3.2.

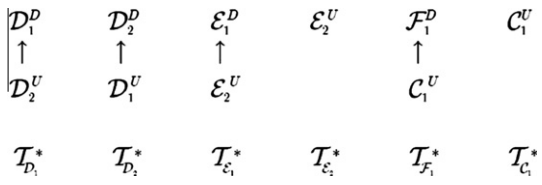

Fig. 6. Marked dialectical trees for arguments of Example 3.2.

defeated. That is, $\mathbb{T}^*_D(Q)$ contains those dialectical trees that show which arguments for Q and $\overline{Q}$ do not provide a warrant. For instance, in Example 3.2 there are four arguments that support claim a, and the dialectical tree of three of them are marked U (see Fig. 5). Hence, $\mathbb{T}^*_U(a) = \{\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}\}$. Since there is only one argument ($\mathcal{X}_1$) that supports claim $\bar{a}$ and its dialectical tree is marked D, then $\mathbb{T}^*_D(a) = \{\mathcal{T}^*_{\mathcal{A}_2}, \mathcal{T}^*_{\mathcal{X}_1}\}$.

As stated above, the purpose of an explanation is to transfer the understanding of how the warrant status of a particular argument can be obtained from a given argumentation theory. Therefore, an explanation will consist of a structure that reflects the analysis carried out in order to obtain such status, and it will contain those arguments and counterarguments that are considered by this analysis.

Definition 3.2 (δ-*explanation*). Let T be an argumentation theory and Q a claim. A δ-explanation for Q in T is the tuple $\mathbb{E}_T(Q) = (\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q))$.

That is, a δ-explanation $\mathbb{E}_T(Q)$ for a claim Q from a theory T is a triplet $(\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q))$, where the first component is a (possible empty) set of those marked dialectical trees from T that provide a warrant for Q. The second component of $\mathbb{E}_T(Q)$ is a (possibly empty) set of marked dialectical trees that provide a warrant for $\overline{Q}$. Finally, the third component $(\mathbb{T}^*_D(Q))$ is a (possibly empty) set that contains those marked dialectical trees for Q or for $\overline{Q}$ that provide no warrant, *i.e.*, their roots are marked D (defeated).

Example 3.3. Consider the argumentation theory $T_{3.2}$ of Example 3.2 where there are four arguments ($\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, and $\mathcal{A}_4$) that support claim a and one argument ($\mathcal{X}_1$) that supports claim $\bar{a}$. The δ-explanation for claim a from $T_{3.2}$ is $\mathbb{E}_{T_{3.2}}(a) = (\{\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}\}, \emptyset, \{\mathcal{T}^*_{\mathcal{A}_2}, \mathcal{T}^*_{\mathcal{X}_1}\})$, whereas the δ-explanation for $\bar{a}$ is $\mathbb{E}_{T_{3.2}}(\bar{a}) = (\emptyset, \{\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}\}, \{\mathcal{T}^*_{\mathcal{A}_2}, \mathcal{T}^*_{\mathcal{X}_1}\})$. Fig. 5 shows all the dialectical trees of $\mathbb{E}_{T_{3.2}}(a)$ and $\mathbb{E}_{T_{3.2}}(\bar{a})$. Observe that the δ-explanation from $T_{3.2}$ for the claim e is $\mathbb{E}_{T_{3.2}}(e) = (\emptyset, \{\mathcal{T}^*_{\mathcal{E}_2}\}, \{\mathcal{T}^*_{\mathcal{E}_1}\})$. The marked dialectical trees of $\mathbb{E}_{T_{3.2}}(e)$ are shown in Fig. 6.

If the first element of a δ-explanation $\mathbb{E}_T(Q)$ is not empty, then there exist at least one argument that provides a warrant for the claim Q. Below we will show that based on a δ-explanation it is possible to define the notion of *answer* for a query (a claim posed to an argumentation theory). But first we will analyze some properties of the proposed formalism and we will introduce the notion of coherent argumentation theory.

Proposition 3.2. *Let* $T = (\langle \mathsf{Args}, \mathsf{R}\rangle, \mathbf{DC})$ *be an argumentation theory and let* $\mathbb{E}_T(Q) = \left(\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q)\right)$ *be a* δ*-explanation. The sets* $\mathbb{T}^*_U(Q)$, $\mathbb{T}^*_U(\overline{Q})$, *and* $\mathbb{T}^*_D(Q)$ *are disjoint.*

Proof. By Proposition 2.1, for each argument in the set *Args* there is a unique exhaustive dialectical tree. Therefore, all the dialectical trees for Q and for $\overline{Q}$ are different, and hence, $\mathbb{T}^*_U(Q)$ and $\mathbb{T}^*_U(\overline{Q})$ are disjoint. Recall that the mark of the root of a dialectical tree is unique by definition, and thus, the set $\mathbb{T}^*_D(Q)$ has no trees whose root is marked U. Therefore, $\mathbb{T}^*_D(Q)$ and $\left(\mathbb{T}^*_U(Q) \cup \mathbb{T}^*_U(\overline{Q})\right)$ are disjoint sets. □

From Proposition 3.2 it holds that $\mathbb{T}^*_D(Q) = (\mathbb{T}^*(Q) \cup \mathbb{T}^*(\overline{Q}))\backslash \left(\mathbb{T}^*_U(Q) \cup \mathbb{T}^*_U(\overline{Q})\right)$. Note that in Example 3.3, the third component of the δ-explanations $\mathbb{E}_{T_{3.2}}(a)$ and $\mathbb{E}_{T_{3.2}}(\bar{a})$ are the same, and the first two are the same but in different order. The following proposition shows that this situation will hold in general for explanations of contradictory claims. Its proof is straightforward from Definitions 3.2 and 3.1.

Proposition 3.3. *Let* T *be an argumentation theory and let* Q *be a claim, then* $\mathbb{E}_T(Q) = (X, Y, Z)$ *iff* $\mathbb{E}_T(\overline{Q}) = (Y, X, Z)$.

Proof. Since $\mathbb{E}_T(Q) = \left(\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q)\right)$ and we assume that $\overline{\overline{Q}} = Q$, then $\mathbb{E}_T(\overline{Q}) = \left(\mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_U(Q), \mathbb{T}^*_D(Q)\right)$. □

The previous examples have shown how it is possible that some of the components of a δ-explanation could be empty. The first component of the explanation $\mathbb{E}_T(Q)$ will be empty if: (a) There exist no argument for Q, or (b) All the dialectical trees corresponding to arguments for Q are marked D. In both cases there is no warrant for Q, but in each one the associated δ-explanation is different. In case (a) $\mathbb{T}^*_D(Q)$ will contain no dialectical trees corresponding to arguments for Q, whereas in case (b) $\mathbb{T}^*_D(Q)$ will contain dialectical trees for arguments that support Q (*e.g.*, $\mathbb{E}_{T_{3.2}}(\bar{a}) = (\emptyset, \{\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}\}, \{\mathcal{T}^*_{\mathcal{A}_2}, \mathcal{T}^*_{\mathcal{X}_1}\})$). The second component of $\mathbb{E}_T(Q)$ will be empty if there is no warrant for $\overline{Q}$, and the third component will be empty when there is no dialectical tree for Q nor $\overline{Q}$ whose root is marked D. Finally, as the next proposition establishes, observe that there are cases where the three components can be empty, *e.g.*, $\mathbb{E}_{T_{3.2}}(Q) = (\emptyset, \emptyset, \emptyset)$.

Proposition 3.4. *Let* T *be an argumentation theory. Then* $\mathbb{E}_T(Q) = (\emptyset, \emptyset, \emptyset)$ *iff no arguments for* Q *nor* $\overline{Q}$ *exist.*

Proof. Assume that there is at least one argument for Q (respectively $\overline{Q}$) then this argument will have exactly one dialectical tree and the root of this tree will be marked either U or D. If the mark is U then $\mathbb{T}^*_U(Q)$ (respectively $\mathbb{T}^*_U(\overline{Q})$) will be non-empty, whereas if the mark is D then $\mathbb{T}^*_D(Q)$ will be non-empty. □

Observe that Definition 3.2 impose no condition over the sets $\mathbb{T}^*_U(Q)$ and $\mathbb{T}^*_U(\overline{Q})$, and thus their elements will depend only on the particular defeat relation and the associated dialectical constraints of the given argumentative theory T. For instance, there can be an argumentative theory T where for a claim Q, both Q and $\overline{Q}$ are warranted from T, *e.g.*, consider the argumentation theory $(\langle\{\mathcal{A}, \mathcal{B}\}, \emptyset\rangle, \emptyset)$, where $\mathsf{claim}(\mathcal{A}) = Q$ and $\mathsf{claim}(\mathcal{B}) = \overline{Q}$. In this theory, no argument defeats the other and therefore both arguments are trivially warranted and so are their claims Q and $\overline{Q}$. If such a situation occurs, then $\mathbb{T}^*_U(Q) \neq \emptyset$ and $\mathbb{T}^*_U(\overline{Q}) \neq \emptyset$. In light of this situation, we propose the following definitions, where the first one states that an argumentative theory will be coherent when for any claim Q, if Q is warranted then $\overline{Q}$ is not warranted.

Definition 3.3 (*Coherent argumentation theory*). An argumentation theory $T = (\langle \mathsf{Args}, \mathsf{R}\rangle, \mathbf{DC})$ is coherent iff for any claim Q such that there exists an argument $\mathcal{A} \in \mathsf{Args}$ with $\mathsf{claim}(\mathcal{A}) = Q$ that is warranted, then there is no argument $\mathcal{B} \in \mathsf{Args}$ with $\mathsf{claim}(\mathcal{B}) = \overline{Q}$ that is warranted.

Answers from coherent argumentation theories are of special importance.

Definition 3.4 (*Coherent* δ*-explanation*). Given a claim Q, a δ-explanation $\mathbb{E}_T(Q) = \left(\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q)\right)$ for Q from an argumentation theory T is said to be coherent iff $\mathbb{T}^*_U(Q) \neq \emptyset$ implies $\mathbb{T}^*_U(\overline{Q}) = \emptyset$.

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

The relation between coherent theories and coherent δ-explanations is thus established.

Theorem 3.1. *Let $T = (\langle \text{Args}, \text{R}\rangle, \mathbf{DC})$ be an argumentation theory. If T is coherent, then for any claim Q its δ-explanation $\mathbb{E}_T(Q) = (\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q))$ is coherent.*

Proof. Consider a δ-explanation where $\mathbb{T}^*_U(Q) \neq \emptyset$; then, by definition of $\mathbb{T}^*_U(Q)$, there exists an argument $\mathcal{A}$ for claim Q where the root of $\mathcal{T}^*_{\mathcal{A}}$ is marked U, and therefore $\mathcal{A}$ is warranted. Since T is coherent, there will be no warranted argument $\mathcal{B}$ in *Args* such that $claim(\mathcal{B}) = \overline{Q}$. Then, for every argument $\mathcal{B}$ with claim $\overline{Q}$, the root of $\mathcal{T}^*_{\mathcal{B}}$ will be marked D. Therefore, $\mathbb{T}^*_U(\overline{Q}) = \emptyset$. □

Note that a δ-explanation $\mathbb{E}_T(Q) = (\emptyset, \emptyset, \mathbb{T}^*_D(Q))$ where $\mathbb{T}^*_D(Q) \neq \emptyset$ is possible and means that although there are arguments for Q or $\overline{Q}$, there is no warrant for either literal (*e.g.*, $\mathbb{E}_{T_{3.2}}(f)$ in Example 3.2).

Many applications of argumentative systems (*e.g.*, expert systems) are query-based. Therefore, it is useful to define two concepts related to an explanation: *query* and *answer* for a query. For instance, following the scenario of Example 3.1, the query "go to the opera show" can be formulated and the answer for that query should be "no". In our proposal, a query Q, called *T-query*, is a potential claim of an argument to be found in an argumentation theory T. Next, we will define T-answers for T-queries, in terms of coherent δ-explanations.

Definition 3.5 (*T-answer*). Let T be a coherent argumentation theory and Q a T-query. Let $\mathbb{E}_T(Q) = (\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q))$ be a δ-explanation for Q obtained from T. The T-answer for Q is:

- YES, if $\mathbb{T}^*_U(Q) \neq \emptyset$.
- NO, if $\mathbb{T}^*_U(\overline{Q}) \neq \emptyset$.
- UNDECIDED, if $\mathbb{T}^*_D(Q) \neq \emptyset$, $\mathbb{T}^*_U(Q) = \emptyset$ and $\mathbb{T}^*_U(\overline{Q}) = \emptyset$.
- UNKNOWN, if $\mathbb{E}_T(Q) = (\emptyset, \emptyset, \emptyset)$.

Proposition 3.5. *Let T be a coherent argumentation theory and Q a T-query, then the T-answer that is obtained from T is unique.*

Proof. Let $\mathbb{E}_T(Q) = \left(\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q)\right)$ be the T-answer for Q in T. If the answer is UNKNOWN, *i.e.*, $\mathbb{E}_T(Q) = (\emptyset, \emptyset, \emptyset)$, then for Proposition 3.4 there is no argument for either Q or its complement $\overline{Q}$ and the only possible answer is UNKNOWN. If the T-answer for the T-query Q is YES then there is at least one argument $\mathcal{A} \in \text{Args}$ with $\text{claim}(\mathcal{A}) = Q$ that is warranted, since the theory T is coherent there is no warranted argument $\mathcal{B} \in \text{Args}$ with $\text{claim}(\mathcal{B}) = \overline{Q}$, *i.e.*, $\mathbb{T}^*_U(\overline{Q}) = \emptyset$ and the answer cannot be NO. Analogously, exchanging the roles of Q and $\overline{Q}$, for the case that the T-answer is NO. In both cases, the answer cannot be UNDECIDED nor UNKNOWN because, depending the case, $\mathbb{T}^*_U(Q)$ or $\mathbb{T}^*_U(\overline{Q})$ is non empty. If the T-answer Q is UNDECIDED, then there is no warranted argument for Q or for $\overline{Q}$, therefore the T-answer cannot be neither YES nor NO; and since $\mathbb{T}^*_D(Q) \neq \emptyset$, the answer cannot be UNKNOWN. □

To understand why a T-query has a particular T-answer, it is essential to examine which arguments have been considered and the existing connections among them. It is important to note that in argumentation systems where the proof procedure is based on the construction of dialectical trees, δ-explanations play a central role. The δ-explanations show the reasoning carried out by the system, and they allow to visualize the support for a given answer. It is clear that without this information it would be very difficult to understand the returned answer.

Returning to Example 3.1 where arguments $\mathcal{O}_{show}$ and $\mathcal{O}_{friends}$ support the claim "go to the opera" (go) and $\mathcal{O}_{baby}$ and $\mathcal{O}_{get}$ support the contrary, "do not go to the opera" ($\overline{go}$). The δ-explanation for go is $\mathbb{E}(go) = (\emptyset, \{\mathcal{O}_{baby}, \mathcal{O}_{get}\}, \{\mathcal{O}_{show}, \mathcal{O}_{friends}\})$ Therefore, the answer for go is NO and the explanation shows why all arguments that support go were considered defeated (dialectical trees were depicted in Fig. 4(b)–(e)). It is important to notice at this point that the examples presented are necessarily small given the space available, but it is easy to see that as the number of arguments grows the size of the trees will tend also to grow. The visualization of the answers will become more important in these cases, that surely are common in real applications.

Example 3.4. Consider again argumentation theory $T_{3.2}$ of Example 3.2. Recall that the dialectical explanation for a is $\mathbb{E}_{T_{3.2}}(a) = (\{\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}\}, \emptyset, \{\mathcal{T}^*_{\mathcal{A}_2}, \mathcal{T}^*_{\mathcal{X}_1}\})$; then, the answer for a is YES and the answer for $\bar{a}$ is NO. Observe that $\mathbb{E}_{T_{3.2}}(d) = (\emptyset, \emptyset, \{\mathcal{T}^*_{\mathcal{D}_1}, \mathcal{T}^*_{\mathcal{D}_2}\})$, therefore the answer for d (and also for $\bar{d}$) is UNDECIDED. Finally, since $\mathbb{E}_{T_{3.2}}(h) = (\emptyset, \emptyset, \emptyset)$, the answer for h is UNKNOWN.

For some particular applications, a δ-explanation may represent too much information and a more concise explanation could be needed. For instance, in order to show that there is a warrant for a claim Q, a single marked dialectical tree whose root is marked U can be shown. Its definition is introduced next.

Definition 3.6 (*Concise explanation*). Let T be a coherent argumentation theory and Q a T-query. A concise explanation for Q, denoted by $\mathbb{CE}_T(Q)$, is a single marked dialectical tree $\mathcal{T}^*$ obtained as follows:

- if the T-answer for Q is YES, then $\mathcal{T}^* = \gamma(\mathbb{T}^*_U(Q))$.
- if the T-answer for Q is NO, then $\mathcal{T}^* = \gamma\left(\mathbb{T}^*_U(\overline{Q})\right)$.
- if the *T-answer for* Q is UNDECIDED, then $\mathcal{T}^* = \gamma(\mathbb{T}^*_D(Q))$.
- if the T-answer for Q is UNKNOWN, then $\mathcal{T}^*$ is a null tree called τ.

Where γ is a selection function that returns exactly one element from a set of marked dialectical trees.

The definition of γ depends on a particular criterion for selecting the most representative dialectical tree, possibly reflecting some set of attributes characterizing the domain of the problem. For instance, to return the tree that has less nodes, the tree with more nodes, or the one with the longest argumentation line.

If a preference criterion between arguments can be defined, then this criterion can be used for selecting the most preferred representative tree. For example, assume that $\mathcal{A} \succ \mathcal{B}$ means $\mathcal{A}$ is preferred over $\mathcal{B}$, then the dialectical tree whose root is best with respect to $\succ$ can be selected Note finally that γ should be defined in such a way that it returns a single dialectical tree, if there are more than one tree that satisfy the adopted criterion, one of them can be randomly selected.

Example 3.5. Consider argumentation theory $T_{3.2}$ of Example 3.2 where there are four arguments ($\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, and $\mathcal{A}_4$) that support claim a and one argument ($\mathcal{X}_1$) that supports claim $\bar{a}$. Recall that $\mathbb{T}^*_U(a) = \{\mathcal{T}^*_{\mathcal{A}_1}, \mathcal{T}^*_{\mathcal{A}_3}, \mathcal{T}^*_{\mathcal{A}_4}\}$ and $\mathbb{T}^*_U(\bar{a}) = \emptyset$, and that the T-answer for a is YES. Consider that γ selects the tree that has less nodes, then $\mathbb{CE}_{3.2}(a) = \mathcal{T}^*_{\mathcal{A}_4}$.

4. Answers and δ-explanations in DELP: a reification

The formal setup of abstract argumentation frameworks has allowed the study of the different semantics emerging from the interaction of arguments through the attack relation. For argumen-

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

tation systems where arguments are provided an internal structure (Besnard & Hunter, 2001; Bondarenko et al., 1997; Chesñevar et al., 2004; García & Simari, 2004; Prakken, 2010; Simari, Chesñevar, & García, 1994; Simari & Loui, 1992), δ-explanations provide the additional capability of aiding in the understanding of how knowledge should be represented and of supporting the debugging process of the underlying knowledge base. See Bryant and Krause (2008) for an in-depth review of implementation of defeasible reasoning systems.

In this section we will demonstrate the capabilities of the formalism defined above by reifying the abstract concepts introduced so far in our formalism. Among the possible options above mentioned, we have chosen a well-known and implemented argumentation system called DELP (Defeasible Logic Programming). The representation language of DELP (García & Simari, 2004) is an extension of the language of Logic Programming. The system defines arguments, the notion of conflict and defeat between arguments (attack in the terminology of abstract argumentation frameworks), and uses a dialectical proof procedure to obtain the warranted arguments. DELP is fully implemented and available online (DeLP-home-page, 2007). Although this decision is based on the possibility of experimenting directly with the implementation to refine our research, it is possible to apply similar constructions to any of the remaining systems mentioned before (Besnard & Hunter, 2001; Bondarenko et al., 1997; Prakken, 2010).

In the last years DELP has been used for knowledge representation and reasoning in different domains. For instance, in Black and Hunter (2009) DELP was adapted and used for a Generative Inquiry Dialogue System; in Chesñevar, Maguitman, and Simari (2006) an argumentation recommender system based on DELP was introduced; in Ferretti, Errecalde, García, and Simari (2007) an application of Defeasible Logic Programming to Decision Making in a robotic environment was proposed; in Gómez, Chesñevar, and Simari (2008) DELP was used for the specification of scripts for reasoning about form fields; and in Thimm and Kern-Isberner (2008) a distributed argumentation framework using Defeasible Logic Programming was defined.

DELP has the declarative capability of representing weak information in the form of *defeasible rules*, and provides a defeasible argumentation inference mechanism for warranting the entailed conclusions. A defeasible rule ($\alpha \prec \beta$) is used to represent tentative information that may be used if nothing can be posed against it (the rule can be read as "reasons to believe in β provide reasons to believe in α"). Below we will provide the essential elements of Defeasible Logic Programming as support for the rest of the presentation; the terminology involved is an extension of the used in Logic Programming. The reader is directed to García and Simari (2004) where full details can be found.

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 strong negation "$\sim$", *e.g.*, *chicken*(*little*) or $\sim$*scared*(*little*). *Strict Rules*, 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 represent non-defeasible information, *e.g.*, *bird* $\leftarrow$ *chicken* or $\sim$*innocent* $\leftarrow$ *guilty*. *Defeasible Rules*, 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 represent tentative information, *e.g.*, $\sim$*flies* $\prec$ *chicken* or *flies* $\prec$ *chicken*,*scared*.

When required, a DELP-program will be denoted (Π,Δ) distinguishing the subset Π of facts and strict rules, and the subset Δ of defeasible rules (see Example 4.1). *Strong negation* is allowed in the head of rules, and hence may be used to represent contradictory knowledge. Given a program (Π,Δ), contradictory literals could be derived. Nevertheless, the set Π (which is used to represent non-defeasible information) must possess certain internal coherence, and therefore no pair of contradictory literals can be derived from Π.

Example 4.1. Consider the DELP-program $(\Pi_{4.1},\Delta_{4.1})$ that represents the scenario introduced in Example 3.1. In this program, *go*, *show_tonight*, *birthday*, *baby*, and *friends* stand for 'go to the opera show', 'there is an opera show tonight', 'today is Bob's birthday', 'a friend is coming with her baby', and 'get together with friends', respectively.

$$\Pi_{4.1} = \{show\_tonight, birthday, baby\},$$
$$\Delta_{4.1} = \left\{ \begin{array}{ll} go \prec showTonight & go \prec showTonight, friends \\ friends \prec birthday & \sim go \prec showTonight, friends, baby \\ \sim go \prec friends & \end{array} \right\}.$$

The program $(\Pi_{4.1},\Delta_{4.1})$ has five defeasible rules representing tentative information, and three facts representing that there is an opera show tonight, today is Bob's birthday, and a friend is coming with her baby. The first defeasible rule states that if there is an opera show tonight then there is a good reason to go to the opera house. The second defeasible rule represents that on his birthday Bob usually gets together with friends. The third one states that, if friends are coming, then Bob would probably stay at home. However, the fourth rule establishes that the fact that there is an opera show tonight and being in the company of friends provide good reasons to go to the opera. The last rule states that the situation changes if a friend brings a baby.

In the previous section, arguments were considered as abstract entities; however, in DELP, arguments are obtained from a particular program, they have a concrete structure, and they must satisfy certain formal restrictions. Next, we introduce the definition of argument and then, in Example 4.2, different arguments that can be obtained from the program $(\Pi_{4.1},\Delta_{4.1})$ are shown.

Definition 4.1 (*Argument structure*). Let (Π,Δ) be a DELP-program, $\langle\mathcal{A},L\rangle$ is an *argument structure*, or simply an *argument*, for a literal L obtained from (Π,Δ) if $\mathcal{A}$ is set of defeasible rules ($\mathcal{A}\subseteq\Delta$) such that: (1) there exists a derivation for L from $\Pi\cup\mathcal{A}$; (2) the set $\Pi\cup\mathcal{A}$ is non-contradictory (*i.e.*, no pair of contradictory literals can be derived); and (3) There is no $\mathcal{A}'\subsetneq\mathcal{A}$ that satisfies (1) and (2) (*i.e.*, $\mathcal{A}$ is a minimal subset satisfying (1) and (2)).

As shown in Example 4.2, in DELP it is possible to build arguments for contradictory literals, *e.g.*, from $(\Pi_{4.1},\Delta_{4.1})$ there are arguments for *go* and $\sim$*go*.

Example 4.2. From the DELP-program $(\Pi_{4.1},\Delta_{4.1})$ introduced in Example 4.1 the following arguments can be obtained:

$\langle\mathcal{O}_1, go\rangle = \langle\{go \prec showTonight\}, go\rangle$
$\langle\mathcal{O}_2, \sim go\rangle = \langle\{(\sim go \prec friends), (friends \prec birthday)\}, \sim go\rangle$
$\langle\mathcal{O}_3, go\rangle = \langle\{(go \prec showTonight, friends), (friends \prec birthday)\}, go\rangle$
$\langle\mathcal{O}_4, \sim go\rangle = \langle\{(\sim go \prec showTonight, friends, baby), (friends \prec birthday)\}, \sim go\rangle$

In DELP an argument $\langle\mathcal{D},M\rangle$ *attacks* $\langle\mathcal{A},L\rangle$ if $\langle\mathcal{D},M\rangle$ is a *proper* or *blocking defeater* for $\langle\mathcal{A},L\rangle$. A defeater for $\langle\mathcal{A},L\rangle$ is an argument $\langle\mathcal{D},M\rangle$ such that $\langle\mathcal{A},L\rangle$ is not preferred over $\langle\mathcal{D},M\rangle$, and L and M are in conconflict, *i.e.*, a pair of contradictory literals can be derived from $\Pi\cup\{L,M\}$. A defeater can be *proper* ($\langle\mathcal{D},M\rangle$ is preferred over $\langle\mathcal{A},L\rangle$) or *blocking* (neither is preferred over the other).

It is important to note that in DELP the argument comparison criterion is modular, and it is possible to select the most appropriate criterion for the domain in question. By default the system provides a syntactic criterion which extends classic specificity and in the examples below we will use it. The criterion, called *generalized*

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

specificity (Stolzenburg, García, Chesñevar, & Simari, 2003), favors two aspects in an argument: it prefers (1) a *more precise* argument (*i.e.*, with greater informational content), and (2) a *more concise* argument (*i.e.*, with less use of rules). Using this criterion, in Example 4.2, argument $\langle \mathcal{O}_3, go\rangle$ is preferred over $\langle \mathcal{O}_2, \sim go\rangle$ (more precise), $\langle \mathcal{O}_4, \sim go\rangle$ is preferred over $\langle \mathcal{O}_3, go\rangle$ and $\langle \mathcal{O}_4, \sim go\rangle$ is preferred over $\langle \mathcal{O}_1, go\rangle$. Note that $\langle \mathcal{O}_2, \sim go\rangle$ is not preferred over $\langle \mathcal{O}_1, go\rangle$ and vice versa. Therefore, $\langle \mathcal{O}_2, \sim go\rangle$ is a blocking defeater for $\langle \mathcal{O}_1, go\rangle$ (and vice versa), $\langle \mathcal{O}_3, go\rangle$ is a proper defeater for $\langle \mathcal{O}_2, \sim go\rangle$, and $\langle \mathcal{O}_4, go\rangle$ is a proper defeater for $\langle \mathcal{O}_3, \sim go\rangle$ and also for $\langle \mathcal{O}_3, \sim go\rangle$.

In DELP an argumentation line will be considered acceptable when the line satisfies $\mathbf{C}_{co}$ (commitment, see Example 2.3), $\mathbf{C}_{nc}$ (non-circularity, see Example 2.2), and a third constraint $\mathbf{C}_{bb}$ that we will introduce next. Consider the argumentation line $\lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$. We define $\mathbf{C}_{bb}(\lambda)$ = False if there are three consecutive arguments in the line such that $\mathcal{A}_k$ is a blocking defeater for $\mathcal{A}_{k-1}$, and $\mathcal{A}_{k+1}$ is a blocking defeater for $\mathcal{A}_k$ $(2 \leqslant k \leqslant n-1)$. Otherwise, $\mathbf{C}_{bb}(\lambda)$ = True.

Example 4.3. Consider the arguments described in Example 4.2. The argumentation lines $[\langle \mathcal{O}_1, go\rangle, \langle \mathcal{O}_4, \sim go\rangle]$ and $[\langle \mathcal{O}_1, go\rangle, \langle \mathcal{O}_2, \sim go\rangle, \langle \mathcal{O}_3, go\rangle]$ are acceptable argumentation lines in DELP, whereas $[\langle \mathcal{O}_1, go\rangle, \langle \mathcal{O}_2, \sim go\rangle, \langle \mathcal{O}_1, go\rangle]$ is not.

DELP's marked (or labeled) dialectical trees will be depicted as a tree where the nodes represent the arguments as triangles with their associated labels ("*U*" or "*D*"), and the edges connecting them denote the defeat relation (see Fig. 7). A bidirectional arrow edge represents a blocking defeat, whereas a unidirectional arrow represents a proper defeat. An argument $\langle \mathcal{A}, L\rangle$ will be pictorially depicted as a triangle, where its upper vertex points to the conclusion *L*, and the name of the set of defeasible rules $\mathcal{A}$ is associated with the triangle itself. At the right of each node the associated mark representing its status ("*U*" or "*D*") will be shown. For instance, Fig. 7 shows four marked dialectical trees obtained from the DELP-program $(\Pi_{4.1}, \Delta_{4.1})$ and associated with the query *go*. Two of them ($\mathcal{T}^*_{\langle \mathcal{O}_1, go\rangle}$ and $\mathcal{T}^*_{\langle \mathcal{O}_3, go\rangle}$) have their roots marked as "*D*", whereas the other two ($\mathcal{T}^*_{\langle \mathcal{O}_2, \sim go\rangle}$ and $\mathcal{T}^*_{\langle \mathcal{O}_4, \sim go\rangle}$) have their roots marked as "*U*". Hence, the literal '∼*go*' is warranted.

We will turn now to the consideration of *explanations* and *answers* for *ground queries* in DELP (called DELP-queries). Later, in Section 4.2, we will generalize explanations and answers for *schematic queries*.

4.1. Queries, answers, and δ-explanations in DELP

As introduced above, a DELP-*query* is a ground literal that DELP will try to warrant. The dialectical process for warranting a DELP-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 DELP-query *Q*, there could exist different arguments that support *Q*, and each argument will generate a different dialectical tree. As we will show below, the returned answer for *Q* will only be, metaphorically speaking, 'the tip of the iceberg' of a set of several dialectical trees that have been explored to support the resulting answer.

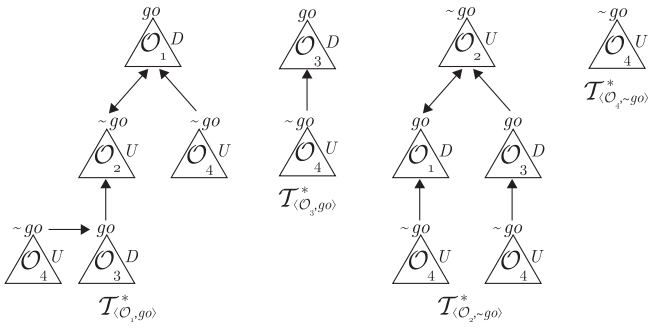

Fig. 7. Dialectical trees of Example 4.1.

To understand why a DELP-query receives a particular answer, it is essential to consider the arguments that have been generated and the existing connections among them. Therefore, an explanation for a DELP-query *Q* will consist of a structure that includes those marked dialectical trees that are considered for establishing the warrant status of *Q*, that is, not only the marked dialectical trees for arguments that support *Q* but also marked dialectical trees for arguments that support $\overline{Q}$. In DELP, $\overline{Q}$ means the complement of *Q* with respect to strong negation "∼" (*i.e.*, $\bar{a} = \sim a$ and $\overline{\sim a} = a$). Based on the formalism presented in the previous section, below we will introduce DELP's corresponding notions of dialectical tree sets and DELP-δ-explanations as particular cases of Definitions 3.1 and 3.2 respectively.

Definition 4.2 (*DELP dialectical tree sets*). Let $\mathcal{P}$ be a DELP-program and *Q* a DELP-query. Let $\mathbb{T}^*(Q) = \{\langle \mathcal{A}_0, Q\rangle, \ldots, \langle \mathcal{A}_n, Q\rangle\}$ be the set of all arguments for *Q* from $\mathcal{P}$, and $\mathbb{T}^*(\overline{Q}) = \{\langle \mathcal{B}_0, \overline{Q}\rangle, \ldots, \langle \mathcal{B}_m, \overline{Q}\rangle\}$ the set of all arguments for $\overline{Q}$ from $\mathcal{P}$. The sets $\mathbb{T}^*_U(Q) \subseteq \mathbb{T}^*(Q)$, and $\mathbb{T}^*_D(Q) \subseteq \mathbb{T}^*(Q) \cup \mathbb{T}^*(\overline{Q})$ are defined as follows:

$$\mathbb{T}^*_U(Q) = \{\mathcal{T}^* \in \mathbb{T}^*(Q) \mid \text{Mark}(\mathcal{T}^*) = U\},$$
$$\mathbb{T}^*_D(Q) = \{\mathcal{T}^* \in (\mathbb{T}^*(Q) \cup \mathbb{T}^*(\overline{Q})) \mid \text{Mark}(\mathcal{T}^*) = D\}.$$

In DELP, given an argument $\mathcal{A}$ there is a unique associated dialectical tree, therefore, a similar result to Proposition 3.2 holds: given a DELP $\mathcal{P}$ and a query *Q*, the sets $\mathbb{T}^*_U(Q)$, $\mathbb{T}^*_U(\overline{Q})$, and $\mathbb{T}^*_D(Q)$ are disjoint. Thus, it also holds that $\mathbb{T}^*_D(Q) = (\mathbb{T}^*(Q) \cup \mathbb{T}^*(\overline{Q})) \setminus (\mathbb{T}^*_U(Q) \cup \mathbb{T}^*_U(\overline{Q}))$.

Definition 4.3 (*DELP-δ-explanation*). Let $\mathcal{P}$ be a DELP-program and *Q* a DELP-query. A DELP-δ-explanation for *Q* from $\mathcal{P}$ is the tuple $\mathbb{E}_{\mathcal{P}}(Q) = (\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q))$.

That is, a DELP-δ-explanation $\mathbb{E}_{\mathcal{P}}(Q)$ for a claim *Q* from a DELP-program $\mathcal{P}$ is a triplet where the first component is a (possibly empty) set of marked dialectical trees from $\mathcal{P}$ that provide a warrant for *Q*. The second component of $\mathbb{E}_{\mathcal{P}}(Q)$ is a (possibly empty) set of marked dialectical trees that provide a warrant for $\overline{Q}$, whereas the third component ($\mathbb{T}^*_D(Q)$) is a (possibly empty) set that contains the marked dialectical trees for *Q* or for $\overline{Q}$ that provide no warrant, *i.e.*, their roots are marked *D* (defeated).

Example 4.4. Consider the DELP-program $\mathcal{P}_{4.1}$ of Example 4.1. The DELP-queries '*go*' and '∼*go*' have the following DELP-δ-explanations:

$$\mathbb{E}_{\mathcal{P}_{4.1}}(go) = (\emptyset, \{\mathcal{T}^*_{\langle \mathcal{O}_4, \sim go\rangle}\}, \{\mathcal{T}^*_{\langle \mathcal{O}_1, go\rangle}, \mathcal{T}^*_{\langle \mathcal{O}_2, \sim go\rangle}, \mathcal{T}^*_{\langle \mathcal{O}_3, go\rangle}\});$$
$$\mathbb{E}_{\mathcal{P}_{4.1}}(\sim go) = (\{\mathcal{T}^*_{\langle \mathcal{O}_4, \sim go\rangle}\}, \emptyset, \{\mathcal{T}^*_{\langle \mathcal{O}_1, go\rangle}, \mathcal{T}^*_{\langle \mathcal{O}_2, \sim go\rangle}, \mathcal{T}^*_{\langle \mathcal{O}_3, go\rangle}\}).$$

Fig. 7 shows all the marked trees included in these explanations. Observe that the δ-explanation $\mathbb{E}_{\mathcal{P}_{4.1}}(sleep) = (\emptyset, \emptyset, \emptyset)$.

Propositions 3.3 and 3.4 also hold for DELP-δ-explanations, that is, given a DELP-program $\mathcal{P}$ and a DELP-query *Q*:

- $\mathbb{E}_{\mathcal{P}}(Q) = (X, Y, Z)$ iff $\mathbb{E}_{\mathcal{P}}(\overline{Q}) = (Y, X, Z)$, and
- $\mathbb{E}_{\mathcal{P}}(Q) = (\emptyset, \emptyset, \emptyset)$ iff there are no arguments for *Q* nor $\overline{Q}$.The corresponding proofs for DELP are straightforward from Definitions 4.2 and 4.3, following analogous steps as the ones used for Propositions 3.3 and 3.4.

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

Recall that a δ-explanation is coherent if $\mathbb{T}^*_U(Q) \neq \emptyset$ implies $\mathbb{T}^*_U(\overline{Q}) = \emptyset$ (Definition 3.5). That is, given a query Q that is warranted, there cannot be warranted arguments that support $\overline{Q}$. The following proposition shows that DELP-δ-explanations have the same property.

Proposition 4.1. *DELP-δ-explanations are coherent.*

Proof. Given a DELP-program $\mathcal{P}$, there cannot be warrants for a query Q and for $\overline{Q}$. Hence, according to Definition 3.3 DELP is coherent. Therefore, by Theorem 3.1 DELP-δ-explanations are coherent. □

Since DELP-δ-explanations are coherent, the answers for DELP-queries can be defined in a similar way as *T*-answers. In DELP, the answer to a query Q is YES when Q is warranted, NO when $\overline{Q}$ is warranted, UNDECIDED if neither Q nor $\overline{Q}$ is warranted, and UNKNOWN when there is no argument for Q. Next, we will define DELP-*answers* for DELP-queries in terms of a DELP-δ-explanation.

Definition 4.4 (*DELP-answer*). Let $\mathcal{P}$ be a DELP-program and Q a DELP-query. Let $\mathbb{E}_{\mathcal{P}}(Q) = \left(\mathbb{T}^*_U(Q), \mathbb{T}^*_U(\overline{Q}), \mathbb{T}^*_D(Q)\right)$ be a DELP-δ-explanation for Q. The DELP-answer for Q is:

- YES, if $\mathbb{T}^*_U(Q) \neq \emptyset$.
- NO, if $\mathbb{T}^*_U(\overline{Q}) \neq \emptyset$.
- UNDECIDED, if $\mathbb{T}^*_D(Q) \neq \emptyset$, $\mathbb{T}^*_U(Q) = \emptyset$ and $\mathbb{T}^*_U(\overline{Q}) = \emptyset$.
- UNKNOWN, if $\mathbb{E}_{\mathcal{P}}(Q) = (\emptyset, \emptyset, \emptyset)$.

In DELP strict and defeasible rules are ground. However, following the usual convention introduced in Lifschitz (1996), we will use "schematic rules" with variables. Each schematic rule represents several ground rules where variables are replaced (instantiated) by ground elements. To distinguish variables, as usual, an initial uppercase letter is used.

Example 4.5. Consider the DELP-program $(\Pi_{4.5}, \Delta_{4.5})$ where '*flies*' is abbreviated as '*f*':

$$\Pi_{4.5} = \left\{ \begin{array}{l} bird(X) \leftarrow chicken(X) \\ chicken(little) \\ chicken(tina) \\ scared(tina) \\ bird(rob) \end{array} \right\}, \quad \Delta_{4.5} = \left\{ \begin{array}{l} f(X) \prec bird(X) \\ f(X) \prec chicken(X), scared(X) \\ \sim f(X) \prec chicken(X). \end{array} \right\}$$

As mentioned before, a DELP-query is a ground literal and arguments are built using ground rules. From the program $(\Pi_{4.5}, \Delta_{4.5})$ the following three arguments can be obtained (where ground instances of schematic rules were used):

$\langle \mathcal{A}_1, f(tina) \rangle = \langle \{f((tina)) \prec bird(tina)\}, f((tina)) \rangle$,
$\langle \mathcal{A}_2, \sim f(tina) \rangle = \langle \{\sim f(tina) \prec chicken(tina)\}, \sim f(tina) \rangle$, and
$\langle \mathcal{A}_3, f(tina) \rangle = \langle \{f(tina) \prec chicken(tina), scared(tina)\}, f(tina) \rangle$.

Here, $\langle \mathcal{A}_2, \sim f(tina) \rangle$ defeats $\langle \mathcal{A}_1, f(tina) \rangle$ and $\langle \mathcal{A}_3, f(tina) \rangle$ defeats $\langle \mathcal{A}_2, \sim f(tina) \rangle$. Observe that the DELP-$\delta$-explanation for DELP-query '*f*(*tina*)' is $(\{\mathcal{T}^*_{\langle \mathcal{A}_1 f(tina) \rangle}, \mathcal{T}^*_{\langle \mathcal{A}_3 f(tina) \rangle}\}, \emptyset \{\mathcal{T}^*_{\langle \mathcal{A}_2, \sim f(tina) \rangle}\})$, and therefore, the answer for this query is YES (Fig. 8 shows the marked dialectical trees). Note that the DELP-δ-explanation for '$\sim$*f*(*tina*)' is $(\emptyset, \{\mathcal{T}^*_{\langle \mathcal{A}_1 f(tina) \rangle}, \mathcal{T}^*_{\langle \mathcal{A}_3 f(tina) \rangle}\}, \{\mathcal{T}^*_{\langle \mathcal{A}_2, \sim f(tina) \rangle}\})$, whose answer is NO. Finally, observe that the answer for '*walks*(*tina*)' is UNKNOWN, because no argument could be formed for that query.

Note that from the point of view of a knowledge engineer of knowledge programmer, DELP-δ-explanations give a global idea of the interactions among arguments within the context of a query. Thus, DELP-δ-explanation can also be used as a debugging tool while programming: whenever an unexpected answer is obtained, the programmer can resort to these explanations to detect errors in the program. It might also happen that the answer appear to be correct but the explanation will reveal a mistake in the knowledge representation. For instance, if in Example 4.5 the rule "$f(X) \prec chicken(X)$" is mistakenly used instead of "$\sim f(X) \prec chicken(X)$', there will be no argument either for $\sim f(little)$ or $\sim f(tina)$. Similar problems may arise in case a literal is misspelled (*e.g.*, "chickem" or "chickn" instead of "chicken"). In these situations, explanations can help the debugging task by showing that some expected arguments are missing or that some unexpected arguments arise.

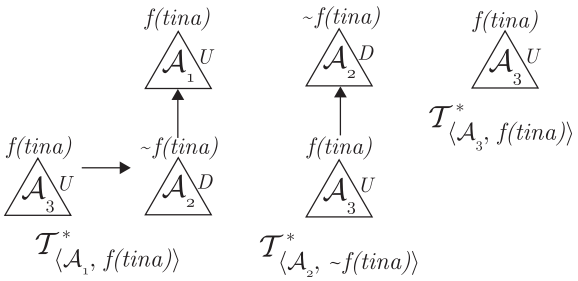

Fig. 8. Dialectical trees of Example 4.5.

In the first part of this section we have considered DELP-δ-explanations for ground queries (*e.g.*, *go*, $\sim f(tina)$ or *bird*(*little*)); nevertheless, sometimes is useful to have a more general type of query. With that in mind, we will introduce *schematic queries* that allow variables and represent a set of individual queries. For instance, in Example 4.5, instead of submitting several ground queries (*f*(*tina*), *f*(*little*), *f*(*rob*), etc.), to know "if it is warranted that some individual flies", a single schematic query can be used. Below, we will extend the notions of DELP-δ-explanations and DELP-answers for this new type of query.

4.2. Schematic queries and generalized DELP-δ-explanations

A schematic query is a literal that has at least one variable; it represents the set of DELP-queries that unify (in the Logic Programming sense (Lloyd, 1987)) with it and only have constants from the program signature. For instance, in the DELP-program of Example 4.5, the schematic query *f*(*X*) will refer to *f*(*tina*), *f*(*rob*), and *f*(*little*). To accommodate this possibility, we will extend the definition of DELP-δ-explanations to include schematic queries. Observe that the schematic query *f*(*X*) actually has infinite terms that unify with the variable *X*; however, all queries with terms that are not in the program signature (*e.g.*, *f*(*mac*) in Example 4.5) 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 a generalized DELP-δ-explanation will refer only to those instances of DELP-queries that contain constants from the program signature.

Schematic queries give us the possibility of asking more general questions than ground queries do. 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 a more complex form of reasoning as we may pose a query, gather the warranted instances, and continue the reasoning process with the individuals involved in these warrants. Thus, an explanation for a schematic query will contain the explanations for the elements of the set of individual DELP-queries that it represents.

Definition 4.5 (*Generalized DELP-δ-explanation*). Let $\mathcal{P}$ be a DELP-program and Q a schematic query. Let $\{Q_1, \ldots, Q_n\}$ be all the ground instances of Q with respect to the signature of $\mathcal{P}$. Let $\mathbb{E}_{\mathcal{P}}(Q_i)$ be the

DELP-δ-explanation for Q_i $(1 \leqslant i \leqslant n)$ from program $\mathcal{P}$. Then, the *generalized* DELP-δ-*explanation* for Q in $\mathcal{P}$ is $\mathbb{GE}_{\mathcal{P}}(Q) = \{\mathbb{E}_{\mathcal{P}}(Q_1), \ldots, \mathbb{E}_{\mathcal{P}}(Q_n)\}$.

Observe that a DELP-δ-explanation (Definition 4.3) is a particular case of a generalized DELP-δ-explanation, where the set $\mathbb{GE}_{\mathcal{P}}(Q)$ is a singleton. Consider for instance the DELP-program $(\Pi_{4.5}, \Delta_{4.5})$ and suppose that we want to know whether or not from this program it can be warranted that a certain individual does not fly. If we make the query for $\sim f(X)$, the answer is YES, because there is a warranted instance: $\sim f(little)$. The supporting argument is: $\langle \mathcal{B}_1, \sim f(little)\rangle = \langle \{\sim f(little) \prec chicken(little)\}, \sim f(little)\rangle$. Note that the answer for the schematic query $f(X)$ is also YES, but with a different set of warranted instances: $f(tina)$ and $f(rob)$. The supporting argument for instance '$X = rob$' is the undefeated argument: $\langle \mathcal{C}_1, f(rob)\rangle = \langle \{f(rob) \prec bird(rob)\}, f(rob)\rangle$.

The generalized DELP-δ-explanation for $f(X)$ from $\mathcal{P}_{4.5} = (\Pi_{4.5}, \Delta_{4.5})$ is:

$$\mathbb{GE}_{\mathcal{P}_{4.5}}(f(X)) = \{(\{\mathcal{T}^*_{\langle \mathcal{A}_1 f(tina)\rangle}, \mathcal{T}^*_{\langle \mathcal{A}_3 f(tina)\rangle}\}, \emptyset, \{\mathcal{T}^*_{\langle \mathcal{A}_2, \sim f(tina)\rangle}\}), (\emptyset, \{\mathcal{T}^*_{\langle \mathcal{B}_2, \sim f(little)\rangle}\}, \{\mathcal{T}^*_{\langle \mathcal{B}_1 f(little)\rangle}\}), (\{\mathcal{T}^*_{\langle \mathcal{C}_1 f(rob)\rangle}\}, \emptyset, \emptyset)\}.$$

The dialectical trees of $\mathbb{GE}_{\mathcal{P}_{4.5}}(f(X))$ are shown in Fig. 9.

Consider a schematic query $Q(X)$ and a DELP-program $\mathcal{P}$. Suppose that $Q(X)$ represents five (ground) DELP-queries: $Q(a)$, $Q(b)$, $Q(c)$, $Q(d)$, and $Q(e)$. As the reader may have noticed from the example above, it may happen that from $\mathcal{P}$ the answer for $Q(a)$ is YES, the answer for $Q(b)$ is NO, the answer for $Q(c)$ is YES, the answer for $Q(b)$ is UNDECIDED, and the answer for $Q(a)$ is NO. Next, we define the answer for a schematic query taking into consideration the individual answers for each ground instance.

Definition 4.6 (*DELP-answer for a schematic query*). Let Q be a schematic query and $\mathcal{P}$ be a DELP-program. Let $\{Q_1, \ldots, Q_n\}$ be all the ground instances of Q w.r.t. the signature of $\mathcal{P}$. The answer for the schematic query Q is:

- YES, if there exists an instance $Q_i \in \{Q_1, \ldots, Q_n\}$ such that the answer for the DELP-query Q_i is YES (*i.e.*, $\mathbb{T}^*_U(Q_i) \neq \emptyset$).
- NO, if for every instance $Q_i \in \{Q_1, \ldots, Q_n\}$, the answer for Q_i is NO, (*i.e.*, $\mathbb{T}^*_U(\overline{Q_i}) \neq \emptyset$ for all Q_i).
- UNDECIDED, if there is no instance $Q_i \in \{Q_1, \ldots, Q_n\}$ such that the answer for the DELP-query Q_i is YES, and there exists an instance $Q_k \in \{Q_1, \ldots, Q_n\}$ such that the answer for Q_k is UNDECIDED.
- UNKNOWN, if $\{Q_1, \ldots, Q_n\} = \emptyset$

Consider, for example, that for a given program $\mathcal{P}$ the following answers are obtained for these ground literals: $p(a)$ YES, $p(b)$ NO, $p(c)$ UNDECIDED, $t(a)$ NO, $t(b)$ NO, and $t(c)$ UNDECIDED. Therefore, the answer for $p(X)$ is YES, and the answer for $t(X)$ is UNDECIDED.

Example 4.6. Consider the DELP-program $\mathcal{P}_{4.6} = (\Pi_{4.6}, \Delta_{4.6})$

$$\Pi_{4.6} = \left\{ \begin{array}{l} adult(peter) \\ adult(annie) \\ unemployed(peter) \\ student(annie) \end{array} \right\}, \quad \Delta_{4.6} = \left\{ \begin{array}{l} car(X) \prec adult(X) \\ \sim car(X) \prec unemployed(X) \\ \sim car(X) \prec student(X) \end{array} \right\}.$$

From $(\Pi_{4.6}, \Delta_{4.6})$ the following arguments can be built:

$\langle \mathcal{N}_1, car(annie)\rangle = \langle \{car(annie) \prec adult(annie)\}, car(annie)\rangle$,
$\langle \mathcal{N}_2, \sim car(annie)\rangle = \langle \{\sim car(annie) \prec student(annie)\}, \sim car(annie)\rangle$,
$\langle \mathcal{P}_1, car(peter)\rangle = \langle \{car(peter) \prec adult(peter)\}, car(peter)\rangle$, *and*
$\langle \mathcal{P}_2, \sim car(peter)\rangle = \langle \{\sim car(peter) \prec unemployed(peter)\}, \sim car(peter)\rangle$.

Using the adopted comparison criterion, $\langle \mathcal{N}_1, car(annie)\rangle$ and $\langle \mathcal{N}_2, \sim car(annie)\rangle$ are blocking defeaters of each other and the same situation occurs for $\langle \mathcal{P}_1, car(peter)\rangle$ and $\langle \mathcal{P}_2, \sim car(peter)\rangle$.

The generalized DELP-δ-explanation for '$car(X)$' is:

$$\mathbb{GE}_{\mathcal{P}_{4.6}}(car(X)) = \{(\emptyset, \emptyset, \{\mathcal{T}^*_{\langle \mathcal{N}_1, car(annie)\rangle}, \mathcal{T}^*_{\langle \mathcal{N}_2, \sim car(annie)\rangle}\}), (\emptyset, \emptyset, \{\mathcal{T}^*_{\langle \mathcal{P}_1, car(peter)\rangle}, \mathcal{T}^*_{\langle \mathcal{P}_2, \sim car(peter)\rangle}\})\}.$$

Fig. 10 shows the mentioned dialectical trees. The explanation shows that neither '$car(annie)$' nor '$\sim car(annie)$' are warranted, and the same holds for '$car(peter)$' and '$\sim car(peter)$'. Therefore, there are no warranted arguments and the DELP-answer for the schematic query $car(X)$ is UNDECIDED.

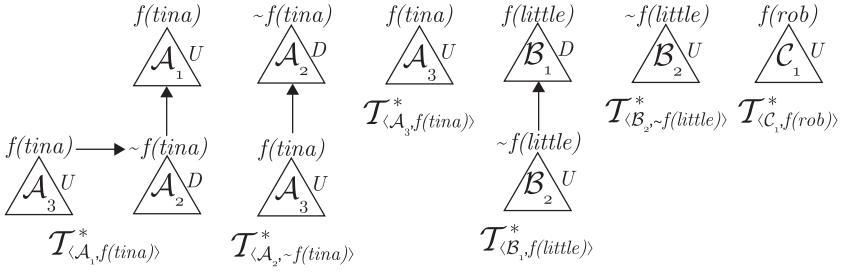

Fig. 9. Dialectical trees associated with $\mathbb{GE}_{\mathcal{P}_{4.5}}(f(X))$.

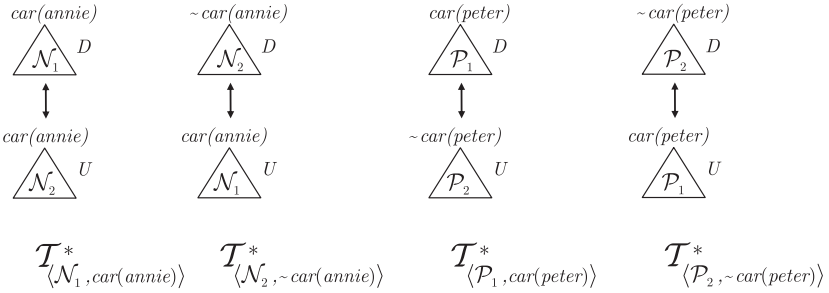

Fig. 10. Dialectical trees of $\mathbb{GE}_{\mathcal{P}_{4.6}}(car(X))$.

4.3. Implementation details

A prototype implementation of generalized DELP-δ-explanation has been developed, integrated with two different implementations of DELP. One of these implementations is available online (DeLP-home-page, 2007). There, DELP-δ-explanations are generated and written into an XML file. It is clear that the translation from δ-explanation to XML is rather trivial and therefore, it is not included here.

DELP-δ-explanations were also used in the implementation of a DELP-server (García, Rotstein, Tucat, & Simari, 2007). A DELP-server is a reasoning service that can be consulted simultaneously by several agents from different remote hosts. The server receives queries and, along with the answers it returns a DELP-δ-explanation in XML format. Therefore, software agents may parse DELP-δ-explanations in order to use this information for their own purposes (*e.g.*, show the explanation to a user, inspect which knowledge was used for obtaining the answer, etc.). Thus, DELP-δ-explanations are represented in such a way that they are useful to both humans and software agents.

5. Conclusions and related work

Recommender systems (Ricci et al., 2011) represent an interesting field where the need for improved explanation facilities has been recognized. In Friedrich and Zanker (2011) explanations in recommender systems are characterized by two properties. First, an explanation should give information about the recommendation. Second, they should be aligned with the design objectives of the system; these objectives are analyzed in Tintarev and Masthoff (2011). An important goal, already highlighted here, is reassuring the user about the recommendation by providing information about the rationale behind it; this information should be given in such a way that the user can validate the mechanism that produced the recommendation.

In the work presented here, we have addressed the problem of providing explanation capabilities to argumentation systems. As stated in the introduction, this is an important and yet undeveloped field in the area of argumentation in Computer Science. Our focus was put on argumentation systems based on a dialectical proof procedure, and we have defined *dialectical explanations* for both, abstract argumentation frameworks and a Logic Programming based argumentation system that is implemented.

One of the contributions of this paper is to introduce a formalization of δ-explanations for abstract argumentation frameworks with dialectical constraints; for this formalization, different properties were proposed and analyzed. We have shown that in abstract argumentation, δ-explanations are a useful tool to comprehend and analyze the interactions among arguments that are under consideration. As another contribution, the proposed explanation formalism was applied for providing explanation capabilities to Defeasible Logic Programming (DELP). Hence, the answer for a query can be explained in terms of the interactions of all the arguments that DELP considered to give that answer. From the user point of view, the answer for a query is explained presenting the whole set of dialectical trees related to the query, and from a DELP programmer point of view, explanations give a global idea of the interactions among arguments within the context of a query. Using these characteristics, DELP-δ-explanation can be also used as a debugging tool while programming: whenever unexpected behavior arises, the programmer can use this type of explanations to detect errors in the program.

An empirical analysis about the impact of different types of explanations in the context of expert systems is given in Ye and Johnson (1995). 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.

A thorough survey relating explanation and argumentation capabilities can be found in Moulin et al. (2002). The authors are mainly concerned about negotiation/persuasion, and interactive/collaborative explanations and they discuss 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. As it was shown, in our proposed approach, we extract all the information from the given knowledge base (*e.g.*, the DELP-program) to return both kinds of information.

We also agree with (Moulin et al., 2002), 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. As it was shown in the examples given in our proposal, our approach provides a higher-level explanation in a way that the whole context of a query can be revealed.

Our approach handles δ-explanations within argumentation systems through a graphical representation of dialectical trees. Visualization in argumentation has been addressed by Schroeder (2000). 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. Schroeder (2000) uses argumentation trees in a similar way as we do, although we focus on their applicability to model explanations (that is, we are concerned with providing the whole context corresponding to the query). As stated in Section 4.3, a δ-explanation can be represented in XML, therefore explanations can be represented in such a way that they can be used by both humans and software agents. Since the translation from a δ-explanation to XML is rather trivial we have not included it in the paper.

Appendix A. Proofs

This appendix includes proofs and auxiliary definitions that are used in the proofs.

Remark 2.1. Observe that given a theory T, if an argumentation line λ satisfies $\mathbf{C}_{nc}$ then any subsequence of λ also satisfies $\mathbf{C}_{nc}$. The same holds for $\mathbf{C}_{co}$.

(a) Consider first $\mathbf{C}_{nc}$. Let $\lambda_a = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ be an argumentation line such that $\mathbf{C}_{nc}(\lambda_a) = \mathsf{True}$. Suppose that a subsequence of λ_a, $\lambda_u = [\mathcal{A}_i, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{i+k}]$ $(0 \leqslant i \leqslant n-k)$ is such that $\mathbf{C}_{nc}(\lambda_u) = \mathsf{False}$. Hence, there must be an argument $\mathcal{A}_p$ $(i \leqslant p \leqslant i+k)$ such that $\mathcal{A}_p \in \lfloor \lambda_u \rfloor_{p-i}$. Then, since λ_u is a subsequence of λ_a, it holds that $\mathcal{A}_p \in \lfloor \lambda_a \rfloor_{p-1}$, and thus, since $\mathcal{A}_p$ also belongs to λ_a then $\mathbf{C}_{nc}(\lambda_a) = \mathsf{False}$, which contradicts our initial hypothesis.

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

(b) Consider now $\mathbf{C}_{co}$. Let $\lambda_b = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ be an argumentation line such $\mathbf{C}_{co}(\lambda_b)$ = True. Suppose that a subsequence of λ_b, $\lambda_u = [\mathcal{A}_i, \mathcal{A}_{i+1}, \ldots \mathcal{A}_{i+k}]$ $(0 \leqslant i \leqslant n-k)$ is such that $\mathbf{C}_{co}(\lambda_u)$ = False. Hence, there must be a pair of arguments $\mathcal{A}_p$ and $\mathcal{A}_q$ $(i \leqslant p, q \leqslant i+k)$ such that $\{\mathcal{A}_p, \mathcal{A}_q\} \subseteq \lambda_{uS}$ or $\{\mathcal{A}_p, \mathcal{A}_q\} \subseteq \lambda_{uI}$, and $(\mathcal{A}_p, \mathcal{A}_q) \in \mathrm{R}$. Then, $\mathcal{A}_p$ and $\mathcal{A}_q$ both belong to λ_{bS} or they both belong to λ_{bI}, and therefore, $\mathbf{C}_{nc}(\lambda_b)$ = False, contradicting our initial hypothesis.

A.1. Auxiliary terminology and properties used for the proof of Proposition 2.1

Let $\lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ be an argumentation line. We will write $|\lambda| = n$ to denote that λ has n arguments and that the *length* of λ is n. Then $\lambda' = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k]$ will be called an *initial argumentation segment* in λ of length k, $k \leqslant n$, 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. Given an argumentation line $[\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$, every subsequence $[\mathcal{A}_i, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{i+k}]$ $(1 \leqslant i \leqslant n-k)$ is also an argumentation line. In particular, every initial argumentation segment is also an argumentation line. Let λ and λ' be two argumentation lines in T. We will say that λ' *extends* λ in T iff $\lambda = \lfloor\lambda'\rfloor_k$, for some $k < |\lambda'|$, that is, λ' extends λ iff λ is a proper initial argumentation segment of λ'.

Definition 5.1 (*Bundle set*). Let T be an argumentation theory and $\mathcal{A}$ an argument in T. A set $S_{\mathcal{A}} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of argumentation lines rooted in $\mathcal{A}$, that are all acceptable with respect to T, is called a *bundle set* for $\mathcal{A}$ in T if and only if there is no pair $\lambda_i, \lambda_j \in S_{\mathcal{A}}$ such that λ_i extends λ_j. If every argumentation line in $S_{\mathcal{A}}$ is exhaustive, then $S_{\mathcal{A}}$ is called an *exhaustive bundle* set for $\mathcal{A}$ in T.

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 vice versa.

Definition 5.2 (*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 $\mathbb{T} : \wp(\mathfrak{Lines}_{\mathcal{A}}) \setminus \{\emptyset\} \to \overline{\mathfrak{Tree}_{\mathcal{A}}}$ as $\mathbb{T}(S_{\mathcal{A}}) =_{def} \overline{\mathcal{T}_{\mathcal{A}}}$, 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 $\mathcal{T}_{\mathcal{A}}$.

Proposition 5.1. *For any argument $\mathcal{A}$ in an argumentative theory T, the mapping $\mathbb{T}$ is a bijection.*

Proof. The mapping is well-defined. If $\mathbb{T}(S_{\mathcal{A}}) = \mathcal{T}_{\mathcal{A}} \neq \mathcal{T}'_{\mathcal{A}} = \mathbb{T}(S'_{\mathcal{A}})$ then there must differ in at least one branch. That implies that there is at least one of the acceptable argumentation lines in $S_{\mathcal{A}}$ that it is different from one of the acceptable argumentation lines in $S'_{\mathcal{A}}$. Thus, $S_{\mathcal{A}} \neq S'_{\mathcal{A}}$.

The mapping is surjective. Clearly, given a $\mathcal{T}_{\mathcal{A}}$ the set of all paths from its leaves to the root is a set S of argument lines that satisfies Definiton 2.7.

The mapping is injective. Let us assume that $\mathbb{T}(S_{\mathcal{A}}) = \mathbb{T}(S'_{\mathcal{A}})$ but $S_{\mathcal{A}} \neq S'_{\mathcal{A}}$. Then there is at least one argumentation line that belongs to $S_{\mathcal{A}}$ that does not belong to $S'_{\mathcal{A}}$ (or the other way around). That argumentation line will produce a branch in the $\mathbb{T}(S_{\mathcal{A}})$ that will not appear in $\mathbb{T}(S'_{\mathcal{A}})$. Therefore, the assumption cannot hold and $S_{\mathcal{A}} = S'_{\mathcal{A}}$. □

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 2.1. *Let T be an argumentation theory; for any argument $\mathcal{A}$ in T there is a unique dialectical tree $\mathcal{T}_{\mathcal{A}}$ in T (up to an equivalence with respect to $\equiv_\tau$ as introduced in Definition 2.8).*

Proof. Let us assume that for a given argument $\mathcal{A}$ there exist two different exhaustive dialectical trees $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}'_{\mathcal{A}}$, *i.e.*, $\mathcal{T}_{\mathcal{A}} \neq \mathcal{T}'_{\mathcal{A}}$. Equivalently, given that $\mathbb{S}(\cdot)$ is injective, $\mathbb{S}(\mathcal{T}_{\mathcal{A}}) = S_{\mathcal{A}} \neq \mathbb{S}(\mathcal{T}'_{\mathcal{A}}) = S'_{\mathcal{A}}$, that is, $S_{\mathcal{A}}$ and $S'_{\mathcal{A}}$ are two different sets of exhaustive argumentation lines rooted in $\mathcal{A}$. Since $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}'_{\mathcal{A}}$ are both exhaustive dialectical trees rooted in $\mathcal{A}$, $S_{\mathcal{A}}$ and $S'_{\mathcal{A}}$ each must contain all the exhaustive argumentation lines rooted in $\mathcal{A}$ (Definition 2.5). Therefore, $S_{\mathcal{A}} = S'_{\mathcal{A}}$ and $\mathbb{T}(S_{\mathcal{A}}) = \mathbb{T}(S'_{\mathcal{A}})$, that is, $\mathcal{T}_{\mathcal{A}} = \mathcal{T}'_{\mathcal{A}}$ contradicting the initial assumption. Thus, given an argument $\mathcal{A}$ in T its exhaustive dialectical tree it is unique. □

References

Amgoud, L., Maudet, N., & Parsons, S. (2002). An argumentation-based semantics for agent communication languages. In *Proceedings of the 15th ECAI, Lyon, France* (pp. 38–42).

Atkinson, K., Bench-Capon, T. J. M., & McBurney, P. (2005). Multi-agent argumentation for edemocracy. In *Proceedings of the third European workshop on multi-agent systems, Belgium* (pp. 35–46).

Baroni, P., Caminada, M., & Giacomin, M. (2011). An introduction to argumentation semantics. *The Knowledge Engineering Review, 26*(4), 365–410.

Baroni, P., & Giacomin, M. (2009). Semantics of abstract argument systems. In I. Rahwan & G. R. Simari (Eds.), *Argumentation in artificial intelligence* (pp. 25–44). Springer Verlag. ISBN: 978-0-387-98196-3.

Baroni, P., Giacomin, M., & Guida, G. (2005). SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence, 168*(1-2), 162–210.

Bench-Capon, T. J. M., & Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial Intelligence, 171*(10–15), 619–641.

Besnard, P., & Hunter, A. (2001). A logic-based theory of deductive arguments. *Artificial Intelligence, 1:2*(128), 203–235.

Besnard, P., & Hunter, A. (2008). *Elements of argumentation*. MIT Press.

Black, E., & Hunter, A. (2009). An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems, 19*(2), 173–209.

Bondarenko, A., Dung, P., Kowalski, R., & Toni, F. (1997). An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence, 93*(1–2), 63–101.

Bryant, D., & Krause, P. (2008). A review of current defeasible reasoning implementations. *The Knowledge Engineering Review, 23*(03), 227–260.

Caminada, M. (2006). On the issue of reinstatement in argumentation. In M. Fisher, W. van der Hoek, B. Konev, & A. Lisitsa (Eds.), *JELIA. Lecture notes in computer science* (Vol. 4160, pp. 111–123). Springer.

Carbogim, D., Robertson, D., & Lee, J. (2000). Argument-based applications to knowledge engineering. *The Knowledge Engineering Review, 15*(2), 119–149.

Chesñevar, C., & Simari, G. R. (2007). A lattice-based approach to computing warranted belief in skeptical argumentation frameworks. In *Proceedings of 20th international joint conference on artificial intelligence (IJCAI 2007), Hyberabad, India*.

Chesñevar, C., Simari, G. R., Alsinet, T., & Godo, L. (2004). A logic programming framework for possibilistic argumentation with vague knowledge. In *UAI 2004, Banff, Canada* (pp. 76–84).

Chesñevar, C., Maguitman, A., & Loui, R. (2000). Logical models of argument. *ACM Computing Surveys, 32*(4), 337–383.

Chesñevar, C., Maguitman, A., & Simari, G. R. (2006). Artificial intelligence applications and innovations. In *Argument-based user support systems using defeasible logic programming. IFIP Series* (pp. 61–69). Springer.

Chesñevar, C., Simari, G. R., & Godo, L. (2005). *Computing dialectical trees efficiently in possibilistic defeasible logic programming*. Springer Verlag, pp. 158–171.

DeLP-home-page. (2007). Web page: <http://lidia.cs.uns.edu.ar/delp>.

Dung, P. (1995). 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.

Ferretti, E., Errecalde, M., García, A. J., & Simari, G. R. (2007). An application of defeasible logic programming to decision making in a robotic environment. In *LPNMR. LNAI* (Vol. 4483, pp. 297–302). Springer.

Friedrich, G., & Zanker, M. (2011). A taxonomy for generating explanations in recommender systems. *AI Magazine, 32*(3), 90–98.

García, A. J., Chesñevar, C. I., & Simari, G. R. (1993). Making argument systems computationally attractive. In *Actas XIII conf. internacional de la sociedad chilena para ciencias de la computación, La Serena, Chile*.

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036

García, A. J., Chesñevar, C. I., Rotstein, N. D., & Simari, G. R. (2007). Abstract presentation of dialectical explanations in defeasible argumentation. In *First international workshop on argumentation and non-monotonic reasoning (ArgNMR 07)* (pp. 17–32).

García, A. J., Rotstein, N. D., & Simari, G.R. (2007). Dialectical explanations in defeasible argumentation. In *ECSQARU* (pp. 295–307).

García, A. J., Rotstein, N. D., Chesñevar, C. I., & Simari, G. R. (2009). Explaining why something is warranted in defeasible logic programming. In *IJCAI 2009 workshop on explanation-aware computing (ExaCt 2009)*.

García, A., Rotstein, N., Tucat, M., & Simari, G. R. (2007). An argumentative reasoning service for deliberative agents. In *KSEM. LNAI* (Vol. 4798, pp. 128–139). Springer.

García, A. J., & Simari, G. R. (2004). Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming, 4*(1), 95–138.

Gómez, S. A., Chesñevar, C. I., & Simari, G. R. (2008). Defeasible reasoning in web forms through argumentation. *International Journal of Information Technology & Decision Making, 7*, 71–101.

Guida, G., & Zanella, M. (1997). Bridging the gap between users and complex decision support systems: the role of justification. In *Proceedings of the 3rd IEEE international conference on engineering of complex computer systems* (pp. 229–238).

Jakobovits, H., & Vermeir, D. (1999). Dialectic semantics for argumentation frameworks. In *ICAIL* (pp. 53–62).

Kakas, A., & Toni, F. (1999). Computing argumentation in logic programming. *Journal of Logic Programming, 9*(4), 515–562.

Lacave, C., & Diez, F. J. (2004). A review of explanation methods for heuristic expert systems. *The Knowledge Engineering Review, 19*(2), 133–146.

Lifschitz, V. (1996). Foundations of logic programs. In G. Brewka (Ed.), *Principles of knowledge representation* (pp. 69–128). CSLI Publications.

Lloyd, J. W. (1987). *Foundations of logic programming* (2nd ed.). New York: Springer-Verlag.

Moulin, B., Irandoust, H., Bélanger, M., & Desbordes, G. (2002). Explanation and argumentation capabilities: towards the creation of more persuasive agents. *Artificial Intelligence Review, 17*(3), 169–222.

Parsons, S., Sierrra, C., & Jennings, N. (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation, 8*, 261–292.

Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument & Computation, 1*(2), 93–124.

Prakken, H., & Sartor, G. (2002). The role of logic in computational models of legal argument – a critical survey. In A. Kakas & F. Sadri (Eds.), *Computational logic: logic programming and beyond* (pp. 342–380). Springer.

Prakken, H., & Vreeswijk, G. (2002). Logical systems for defeasible argumentation. In D. Gabbay & F. Guenther (Eds.), *Handbook of philosophical logic* (pp. 219–318). Kluwer Academic Publishers.

Rahwan, I., & Simari, G. R. (2009). *Argumentation in artificial intelligence*. Springer.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (Eds.). (2011). *Recommender systems handbook*. Springer.

Schroeder, M. (2000). Towards a visualization of arguing agents. *Future Generation Computer System, 17*(1), 15–26.

Simari, G. R., Chesñevar, C. I., & García, A. J. (1994). Focusing inference in defeasible argumentation. In *IV Iberoamerican congress on artificial intelligence (IBERAMIA'94), Venezuela*.

Simari, G. R., Chesñevar, C. I., & García, A. J. (1994). The role of dialectics in defeasible argumentation. In *International conference of the Chilenean computer science society* (pp. 111–121).

Simari, G. R., & Loui, R. P. (1992). A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence, 53*, 125–157.

Stolzenburg, F., García, A., Chesñevar, C. I., & Simari, G. R. (2003). Computing generalized specificity. *Journal of Non-Classical Logics, 13*(1), 87–113.

Thimm, M., & Kern-Isberner, G. (2008). A distributed argumentation framework using defeasible logic programming. In *Proceedings of the 2nd international conference on computational models of argument (COMMA'08)* (pp. 381–392). IOS Press.

Tintarev, N., & Masthoff, J. (2011). Designing and evaluating explanations for recommender systems. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 479–510). Springer.

Verheij, B. (1996). Two approaches to dialectical argumentation: admissible sets and argumentation stages. In J. -J. Meyer, & L. van der Gaag (Vol. Eds.), *Proceedings of the eighth Dutch conference on artificial intelligence (NAIC'96), Utrecht, The Netherlands* (pp. 357–368).

Verheij, B. (2007). A labeling approach to the computation of credulous acceptance in argumentation. In M. M. Veloso (Vol. Ed.), *IJCAI* (pp. 623–628).

Walton, D. (2004). A new dialectical theory of explanation. *Philosophical Explorations, 7*(1), 71–89.

Ye, L. R., & Johnson, P. E. (1995). The impact of explanation facilities on user acceptance of expert systems advice. *MIS Quarterly, 19*(2), 157–172.

Please cite this article in press as: García, A. J., et al. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2012.12.036