A Redefinition of Arguments in Defeasible Logic Programming

Ignacio Viglizzo and **Fernando Tohmé** and **Guillermo Simari**
Universidad Nacional del Sur, Bahía Blanca, Argentina
{viglizzo,ftohme}@criba.edu.ar, grs@cs.uns.edu.ar

Abstract

Defeasible Logic Programming (DELP) is a formalism that extends declarative programming to capture defeasible reasoning. Its inference mechanism, upon a query on a literal in a program, answers by indicating whether or not it is warranted in an argumentation process. While the properties of DELP are well known, some of its basic elements can be redefined in order to shed light on some of the subtleties of the warrant process. We will discuss these alternative definitions and the cases in which they provide a better performance.

Introduction

The inference mechanism known as Defeasible Logic Programming (from now on DELP) has been studied for some time now, and applied to many fields. Many extensions have been considered[1]. This paper intends to take a step back to give a rigorous look at its formal foundations and suggest some possible improvements.

We present here a redefinition of the basic elements of DELP (García and Simari 2004) in order to put this framework in line with recent developments in defeasible reasoning. This version, by considering alternative characterizations of those elements, provides a enhanced formal approach to DELP.

In DELP the goal is to determine whether literals in a logic program including defeasible rules are warranted or not. It proceeds by constructing *arguments* for and against any given literal and drawing different *argumentation lines* (i.e. sequences of arguments, in which each one attacks the previous one). The whole family of argumentation lines for a literal defines a *dialectical tree*. A marking procedure, finally, determines if the literal becomes warranted or not.

The basic issues in DELP involve the definition of argument, the condition of *defeat* among them as well as the admissibility conditions for the construction of argumentation lines. In the original version arguments regarded only defeasible rules, but this was a source for some problems in the characterization of defeat, as we noted in (Viglizzo, Tohmé, and Simari 2008). In the current version, these problems disappear, thanks to slight changes in the definitions. We have chosen to introduce first our new system and left the comparison with the existing one for the last section.

In the next section we present the motivations of the present work. In the following one we introduce the new definitions and then we compare our new proposal with the existing framework. The main point is that this new approach provides a more flexible formal model, able to handle some finer distinctions among arguments.

Preliminaries: A New Look at DeLP

A central topic in this paper is a new definition of argument in DELP to include more structure. That will allow us to enhance the notion of *warrant*. Nevertheless, we will maintain the intuitions behind the original presentation (García and Simari 2004; García 2000; García and Simari 1999) of DELP. The motivation of the present work can be found in some developments (Viglizzo, Tohmé, and Simari 2008; Falappa, Kern-Isberner, and Simari 2002; Stolzenburg et al. 2003; Chesñevar et al. 2003; Chesñevar and Simari 2001) that show that a finer, more structured, version of DELP could enhance the possibilities of applying this framework in Multi-Agent Systems.

The general approach to reasoning in DELP starts with a defeasible logic program and takes a dialectical view of the process of inference. In classical logic, proofs are undisputed. A well-formed formula, *wff*, L is said to be a consequence of a set $\mathcal{S}$ of wffs if and only if there exists a sequence $L_1, L_2, \ldots, L_n$ of wffs such that L is L_n and, for each i either L_i is an axiom or L_i is in $\mathcal{S}$, or L_i is a direct consequence by some rule of inference of some of the preceding wffs in the sequence. The sequence $L_1, L_2, \ldots, L_n$ is called a *proof*.

In DELP, some of the rules are *defeasible* and can be challenged. A literal L will be *warranted* if there exists a non-defeated *argument* supporting L. An argument $\mathcal{A}$ for a literal L is a minimal and consistent set of rules that allows to infer L. In order to establish whether the argument $\mathcal{A}$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for the original argument are considered and compared using some preference criterion, to decide if they are better than the attacked argument. Since counter-arguments are arguments, there may exist de-

[1] See (Tohmé and Simari 2004; Capobianco, Chesñevar, and Simari 2005; Alsinet et al. 2008), among others.

featers for them, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called *argumentation line* appears where each argument defeats its predecessor in the line. In general, each argument has more than one defeater and therefore more than one argumentation line exists. Considering all argumentation lines, a tree with arguments as nodes is constructed. In that tree, called *dialectical tree*, the root is the starting argument and each path from the root to a leaf represents an argumentation line. Finally, a *dialectical analysis* of this tree is used to decide whether the initial argument is defeated or not. In case that argument $\mathcal{A}$ is undefeated, $\mathcal{A}$ is called a *warrant* and the literal L that supports is said to be *warranted*. The set of warranted literals represents the semantics of the defeasible logic program.

Basic Definitions

In this section, we build a refinement on the definitions for DELP as presented in García and Simari, 2004.

Definition 1 (Atoms, Literals, and Complements) Let At be a set. The elements of At will be called atoms. If x is an atom, $\sim x$ will be called a *negated atom*. Given a set $X \subseteq$ At of atoms, $\sim X$ is the set $\{\sim x : x \in X\}$. The set Lit is the set of all literals in At$\cup \sim$At. The complement $\bar{l}$ of a literal $l \in$ Lit is $\sim x$ if l is an atom x and x if l is a negated atom $\sim x$.

Definition 2 (Fact) A *Fact* is a literal, *i. e.* an atom, or a negated atom.

Definition 3 (Strict Rule) A *Strict Rule* is an ordered pair, denoted "$Head \leftarrow Body$", whose first member, *Head*, is a literal, and whose second member, *Body*, is a finite, non-empty set of literals. A strict rule with the head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as: $L_0 \leftarrow L_1, \ldots, L_n$ $(n > 0)$ or $L_1 \wedge \ldots \wedge L_n \rightarrow L_0$.

Definition 4 (Defeasible Rule) A *Defeasible Rule* is an ordered pair, denoted "$Head \prec Body$", whose first member, *Head*, is a literal, and whose second member, *Body*, is a finite non-empty set of literals. A defeasible rule with head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as: $L_0 \prec L_1, \ldots, L_n$ $(n > 0)$, or $L_1 \wedge \ldots \wedge L_n \succ L_0$.

Definition 5 (Defeasible Logic Program) A *Defeasible Logic Program* $\mathcal{P}$, abbreviated *delp*, is the disjoint union of three possibly infinite sets: a set of facts Π_f, a set of strict rules Π_r and a set of defeasible rules Δ. When required, we will denote $\mathcal{P}$ as (Π, Δ), where $\Pi = \Pi_f \cup \Pi_r$.

Example 1 The following program $\mathcal{P}$=(Π,Δ), written in the traditional notation as:
Π = { $r \leftarrow s$; p; s}
Δ = { $q \prec p$; $\sim q \prec p, r$}
will be written as:
$\Pi_f = \{p, s\}$
$\Pi_r = \{s \rightarrow r\}$
$\Delta = \{p \succ q, p \wedge r \succ \sim q\}$

Definition 6 (Defeasible and Strict Derivation) Let $\mathcal{P}$ be a *delp* and L a ground literal. A *derivation* of L from $\mathcal{P}$, denoted $\mathcal{P} \mid\sim L$, consists of a finite sequence $L_1, L_2, \ldots, L_n = L$ of ground literals, and each literal L_i is in the sequence because:

(i) $L_i \in \Pi_f$, or
(ii) there exists a rule $R_i \in \Pi_r \cup \Delta$ with head L_i and body $B_1, B_2, \ldots, B_k$ and every literal of the body is an element L_j of the sequence appearing before L_i, $j < i$.

If all the rules used in the derivations come from the set Π_r, then we say that the sequence is a *strict derivation*. In this case we will write $\mathcal{P} \vdash L$. If at least one of the rules comes from Δ, we will say it is a *defeasible derivation*.

Definition 7 (Annotated Derivation) Let $\mathcal{P}$ be a *delp* and L a ground literal. An *annotated derivation* of L from $\mathcal{P}$, consists of a finite sequence of rules and facts $[R_1, R_2, \ldots, R_n]$, where L is the fact R_n or the head of the rule R_n and if a rule R_i is in the sequence, then its body $B_1^i \wedge \ldots \wedge B_{k_i}^i$, is such that for all $B_j^i, 1 \leq j \leq k_i$, B_j^i is a fact or it appears as the head L_m, for some rule R_m with $1 \leq m < i$.

Remark 1 Notice that facts could be added to the sequence at any point prior to their appearance in the body of a rule. Even so, we might endow the set of facts and rules with a total order so that there is a way to choose one of all the annotated defeasible derivations of a literal that contain the same rules, by using a lexicographical order on those sequences.

Definition 8 (Annotated Sub-derivation) Let $\mathcal{P}$ be a *delp* and L a ground literal. Let D be an annotated derivation of L from $\mathcal{P}$. We will say that an annotated derivation D' of L' from $\mathcal{P}$ is a *sub-derivation* of D if every member of D' is a also a member of D.

Example 2 From the program $\mathcal{P}$:
$\Pi_f = \{p, s\}$
$\Pi_r = \{s \rightarrow r\}$
$\Delta = \{p \succ q, p \wedge r \succ \sim q\}$,
the next annotated derivation D for $\sim q$ can be constructed:

$$[s, s \rightarrow r, p, p \wedge r \succ \sim q]$$

and the annotated derivation D' = $[s, s \rightarrow r]$ is a sub-derivation of D.

From the point of view of the analysis of the process of reasoning, interesting derivations must be concise.

Definition 9 (Minimal Derivation) Let $\mathcal{P}$ be a *delp* and L a ground literal. Let D be an annotated derivation of L from $\mathcal{P}$, we will say that D is a *minimal* annotated derivation if there is no proper sub-derivation D' of D such that D' is also an annotated derivation of L.

Remark 2 Every derivation D for L contains at least a minimal sub-derivation D' for that literal L.

As we will see, introducing unnecessary elements in a defeasible derivation weakens its ability to support its conclusion. For that reason, we will only consider minimal derivations.

There are other ways in which derivations can be minimal. Consider the following example:

Example 3 Let $\mathcal{P}$ be the program composed of $\Pi_f = \{a, b\}$; $\Pi_r = \{a \wedge b \rightarrow c,\ b \rightarrow e,\ d \rightarrow g,\ g \rightarrow f\}$; $\Delta = \{c \succ f,\ a \succ d,\ e \succ f\}$.
We have the following minimal derivations for the literal f:

$$[a, b, a \wedge b \rightarrow c, c \succ f]$$

$$[a, a \succ d, d \rightarrow g, g \rightarrow f]$$

$$[b, b \rightarrow e, e \succ f]$$

While the three of them are minimal in the sense defined above, we can observe that the first two are longer than the third one, and that the first one uses both facts a and b, while the second one uses only a.

One may also want to avoid using defeasible rules as much as possible. This adds yet another dimension to the study of these derivations.

We can also observe that all three are minimal subderivations for f of the annotated derivation:

$$[a, b, a \wedge b \rightarrow c, c \succ f, a \succ d, d \rightarrow g, g \rightarrow f, b \rightarrow e, e \succ f]$$

One of the properties that a piece of reasoning should have as such is internal consistency. The following definitions characterize that property.

Definition 10 (Consequence Operators) For a given program $\mathcal{P}$, and sets $X \subseteq \text{Lit}$, $\mathcal{R} \subseteq \Pi_r$ and $\mathcal{D} \subseteq \Delta$, $C(X, \mathcal{R}, \mathcal{D})$ is the set of all literals derivable from $X \cup \mathcal{R} \cup \mathcal{D}$. For any subprogram $\mathcal{A}$ of $\mathcal{P}$, we let $C_{\mathcal{P}}(\mathcal{A}) = C(\Pi_f, \Pi_r, \mathcal{A} \cap \Delta)$, while with $C_{\mathcal{A}}(\mathcal{A})$, or just $C(\mathcal{A})$, we denote the set $C(\Pi_f \cap \mathcal{A}, \Pi_r \cap \mathcal{A}, \Delta \cap \mathcal{A})$.

Remark 3 The idea behind having two different operators $C_{\mathcal{P}}$ and C is that we will use the first one to determine the acceptable derivations from a program, but we will afterwards look only at the derivations on their own to compare two of them.

Definition 11 (Contradictory and Consistent Sets) Given a set $X \subseteq \text{Lit}$, let $X^+ = X \cap \mathsf{At}$ and $X^- = \{a \in \mathsf{At} :\sim a \in X\}$. X is said to be *contradictory* if $X^+ \cap X^- \neq \emptyset$. A subprogram $\mathcal{A}$ of $\mathcal{P}$ is *contradictory* if $C(\mathcal{A})$ is contradictory, and *inconsistent* if $C_{\mathcal{P}}(\mathcal{A})$ is contradictory. If $\mathcal{A}$ is not inconsistent, it will be called *consistent*.

It is clear that a contradictory program is inconsistent, since the consequence operator is monotonic.

Example 4 In the program of example 2, $C_{\mathcal{P}}(\mathcal{P}) = C(\Pi_f, \Pi_r, \Delta) = \{p, s, r, q, \sim q\}$ is contradictory.

We assume that for all programs, the set Π of facts and strict rules is not contradictory, that is, that $C_{\mathcal{P}}(\emptyset) = C(\Pi_f, \Pi_r, \emptyset)$ is not contradictory.

Definition 12 (Consistent Derivation) Let D be an annotated derivation. We will say that D is *consistent* if the set of all facts and rules used in D is consistent.

Definition 13 (Brief) We call a minimal, consistent annotated derivation a *brief*.

Definition 14 (Argument) Let $\mathcal{P}$ be a *delp*, and let D be a brief for a literal L from $\mathcal{P}$. We will say that the set $\mathcal{A}$ of facts and rules, strict and defeasible, contained in the derivation D is an *argument for L constructed from $\mathcal{P}$*.

So an argument for L constructed from $\mathcal{P}$ is the set obtained from applying a forgetful operator Arg to a brief. If D is a brief we can write the argument obtained as $\mathcal{A} = \mathsf{Arg}(D)$.

Remark 4 From the definitions above, we notice that given an annotated derivation D, constructed from a program $\mathcal{P}$, the set of rules and facts involved in that derivation represents a subprogram of $\mathcal{P}$, and therefore an argument fo a literal L is a minimal, consistent subprogram $\mathcal{A}$ of $\mathcal{P}$ from which L can derived.

Notation: We will use the notation $\langle \mathcal{A}, L \rangle_{\mathcal{P}}$ to represent the fact that $\mathcal{A}$ is an argument for L constructed from $\mathcal{P}$. Usually, when the reference to the defeasible logic program $\mathcal{P}$ is clearly understood, we will simplify the notation for arguments as $\langle \mathcal{A}, L \rangle$. Unless it is necessary, we will not mention the defeasible program $\mathcal{P}$. Given a defeasible logic program $\mathcal{P}$, we will denote the set of all arguments $\langle \mathcal{A}, L \rangle$ that can be constructed from $\mathcal{P}$ as $\mathcal{A}rgs(\mathcal{P})$.

Remark 5 While we keep the notation above, we observe that the literal L is redundant as the next proposition shows.

Proposition 1 Given an argument $\mathcal{A}$, there is a unique literal L so that the facts and rules in a set $\mathcal{A}$ can be listed as a brief for L.

Proof. Assume that there are two such literals L_1 and L_2. Let $[R_1, R_2, \ldots, R_n]$ be a brief for L_1. Assume that L_2 is the head of rule R_i. If $i < n$, then $[R_1, \ldots, R_i]$ is a shorter brief for L_2, and this contradicts the minimality of $\mathcal{A}$. Therefore $R_i = R_n$ and $L_1 = L_2$. □

Definition 15 (Sub-Argument) Let $\mathcal{P}$ be a *delp*, and let $\mathcal{A}$ be an argument for a literal L constructed from a brief D produced from $\mathcal{P}$. If D_1 is a brief for a literal $L_1 \in C(\mathcal{A})$ and a sub-derivation of D, then $\mathcal{A}_1 = \mathsf{Arg}(D_1)$ is a *sub-argument* of $\mathcal{A}$. It is clear that $\mathcal{A}_1 \subseteq \mathcal{A}$ and we will also use the notation $\langle \mathcal{A}_1, L_1 \rangle \subseteq \langle \mathcal{A}, L \rangle$ to reflect this.

Proposition 2 Let $\mathcal{P}$ be a *delp*, and let $\mathcal{A}$ be an argument for a literal L constructed from a brief D produced from $\mathcal{P}$. Then, $L_1 \in C(\mathcal{A})$ if and only if there is a *unique* sub-argument $\langle \mathcal{A}_1, L_1 \rangle$ of $\langle \mathcal{A}, L \rangle$.

Proof. Let L_1 be a literal in $C(\mathcal{A})$. Then, there is a derivation of L_1 using the rules and facts in $\mathcal{A}$. Furthermore, there is a brief D_1 for L_1 that uses only those facts and is therefore a sub-derivation of the brief that gives rise to the argument $\mathcal{A}$. Then $\mathcal{A}_1 = \mathsf{Arg}(D_1)$ is a sub-argument of $\mathcal{A}$. To see that it is unique in these conditions, assume that there are two different ones. Then there are in $\mathcal{A}$ redundant rules for the derivation of L_1, which contradicts the minimality of $\mathcal{A}$.
On the other direction, if $\langle \mathcal{A}_1, L_1 \rangle$ is a sub-argument of $\langle \mathcal{A}, L \rangle$, then there is a derivation for L_1 using the facts and rules of $\mathcal{A}$, that is, $L_1 \in C(\mathcal{A})$. □

Remark 6 Let $\mathcal{P}$ be a defeasible logic program. The relation of sub-argumentation is a partial order relation in the set of all possible arguments that can be constructed from $\mathcal{P}$, *i. e.* ($\mathcal{A}rgs(\mathcal{P})$; $\subseteq$) is a poset.

Definition 16 (Disagreement) Let $\mathcal{P}$ be a *delp*. We say that two literals L_1 and L_2 *disagree* if and only if the set $C_{\mathcal{P}}(\{L_1, L_2\})$ is contradictory.

Example 5 Let $\mathcal{P}$ be the program $\Pi_f = \{a, c\}, \Pi_r = \{d \rightarrow\sim b\}, \Delta = \{a \succ b, c \succ d\}$.
The literals d and b disagree, since $\sim b$ is a consequence of d using a strict rule of the program.

Definition 17 (Counter-argument) Let $\langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_2, L_2 \rangle$ be two arguments. We say that $\langle \mathcal{A}_1, L_1 \rangle$ *counter-argues*, *rebuts*, or *attacks* $\langle \mathcal{A}_2, L_2 \rangle$ at literal L if and only if $L \in C(\mathcal{A}_2)$ and literals L and L_1 *disagree*.
We put in this case $\langle \mathcal{A}_2, L_2 \rangle R_L \langle \mathcal{A}_1, L_1 \rangle$. Thus, for each literal L we have a binary relation R_L and we can define the *attacking relation* as $R = \cup_{L \in \mathrm{Lit}} R_L$.

Example 6 Let $\mathcal{P}$ be the program: $\Pi_f = \{a, c\}, \Pi_r = \{d \rightarrow\sim b\}, \Delta = \{a \succ b, c \succ d\}$ and let $\mathcal{A} = \{a, a \succ b\}$, $\mathcal{B} = \{c, c \succ d\}, \mathcal{C} = \{c, d \rightarrow\sim b, c \succ d\}$.
We can depict the attacking relation with arrows labeled by the literals at which the attack occurs.

$$\langle \mathcal{B}, d \rangle \underset{b}{\overset{d}{\leftrightarrows}} \langle \mathcal{A}, b \rangle \underset{d, \sim b}{\overset{b}{\leftrightarrows}} \langle \mathcal{C}, \sim b \rangle$$

Given a pair of arguments it is interesting to have a comparison criterion that would allow to decide which one is better in some sense. We will assume there is a binary relation $\prec$ that is a subset of the attacking relation R. We call this relation pre-defeating, since we will extend it as follows:

Definition 18 (Defeaters) $\langle \mathcal{A}_1, L_1 \rangle$ is a *proper defeater* of $\langle \mathcal{A}_2, L_2 \rangle$ (at literal L) if $\langle \mathcal{A}_2, L_2 \rangle R_L \langle \mathcal{A}_1, L_1 \rangle$ and the unique sub-argument $\langle \mathcal{A}, L \rangle$ of $\langle \mathcal{A}_2, L_2 \rangle$ is such that $\langle \mathcal{A}, L \rangle \prec \langle \mathcal{A}_1, L_1 \rangle$. We denote this by $\langle \mathcal{A}_2, L_2 \rangle < \langle \mathcal{A}_1, L_1 \rangle$.
$\langle \mathcal{A}_1, L_1 \rangle$ is a *blocking defeater* of $\langle \mathcal{A}_2, L_2 \rangle$ (at literal L) if $\langle \mathcal{A}_2, L_2 \rangle R_L \langle \mathcal{A}_1, L_1 \rangle$ and the unique sub-argument $\langle \mathcal{A}, L \rangle$ of $\langle \mathcal{A}_2, L_2 \rangle$ is such that $\langle \mathcal{A}, L \rangle \not\prec \langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_1, L_1 \rangle \not\prec \langle \mathcal{A}, L \rangle$. Whenever this is the case we use the notation $\langle \mathcal{A}_2, L_2 \rangle \approx \langle \mathcal{A}_1, L_1 \rangle$, but we must keep in mind this is not necessarily a symmetric relation.
We call *defeaters* of an argument to all proper and blocking defeaters. If $\langle \mathcal{A}_2, L_2 \rangle$ is a defeater of $\langle \mathcal{A}_1, L_1 \rangle$ we write $\langle \mathcal{A}_1, L_1 \rangle \leq \langle \mathcal{A}_2, L_2 \rangle$.

For the sake of completeness we present next the definitions, taken from García and Simari, 2004 that conform the DELP system, although we have not made significant changes on them.

Definition 19 (Argumentation Line) Let $\mathcal{P}$ be a *delp* and $\langle \mathcal{A}_0, L_0 \rangle$ an argument obtained from $\mathcal{P}$. An *argumentation line* for $\langle \mathcal{A}_0, L_0 \rangle$ is a sequence of arguments from $\mathcal{P}$, denoted $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \langle \mathcal{A}_3, L_3 \rangle, \ldots]$, where each element of the sequence $\langle \mathcal{A}_i, L_i \rangle$, $i > 0$, is a defeater of its predecessor $\langle \mathcal{A}_{i-1}, L_{i-1} \rangle$.

Definition 20 (Supporting and Interfering Arguments) Let $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_2, L_2 \rangle \langle \mathcal{A}_3, L_3 \rangle, \ldots]$ be an argumentation line. We define the set of *supporting arguments* as $\Lambda_S = \{\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \ldots\}$, and the set of *interfering arguments* $\Lambda_I = \{\langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_3, L_3 \rangle, \ldots\}$.

Definition 21 (Concordance) Let $\mathcal{P}$ be a *delp*. Two arguments $\langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_2, L_2 \rangle$ are *concordant* if and only if the set $C_{\mathcal{P}}(\mathcal{A}_1 \cup \mathcal{A}_2)$ is non-contradictory. More generally, a set of arguments $\{\langle \mathcal{A}_i, L_i \rangle\}_{i=1}^{n}$ is concordant if and only if $C_{\mathcal{P}}(\bigcup_{i=1}^{n} \mathcal{A}_i)$ is non-contradictory.

Definition 22 (Acceptable Argumentation Line) An argumentation line $\Lambda = [\langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle]$ is an *acceptable argumentation line* if and only if:

1. Λ is a finite sequence.
2. The sets Λ_S and Λ_I, of supporting and interfering arguments are concordant.
3. No argument $\langle \mathcal{A}_k, L_k \rangle$ in Λ is a sub-argument of an argument $\langle \mathcal{A}_i, L_i \rangle$ appearing earlier in Λ ($i < k$.)
4. For all i such that the argument $\langle \mathcal{A}_i, L_i \rangle$ is a blocking defeater for $\langle \mathcal{A}_{i-1}, L_{i-1} \rangle$, if $\langle \mathcal{A}_{i+1}, L_{i+1} \rangle$ exists, then it is a proper defeater for $\langle \mathcal{A}_i, L_i \rangle$.

Definition 23 (Dialectical Tree) A dialectical tree for an argument constructed from a program $\mathcal{P}$ for $\langle \mathcal{A}, L \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}, L \rangle}$, is formed taking all the acceptable argumentation lines that start with $\langle \mathcal{A}, L \rangle$: the line $[\langle \mathcal{A}, L \rangle]$ is the root of the tree and a line $[\langle \mathcal{A}, L \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle]$ is the child of $[\langle \mathcal{A}, L \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_{n-1}, L_{n-1} \rangle]$ for all $n \geq 1$.

Procedure 1 (Marking of a Dialectical Tree) Let $\mathcal{T}_{\langle \mathcal{A}, L \rangle}$ be a dialectical tree for $\langle \mathcal{A}, L \rangle$. The corresponding marked dialectical tree, denoted $\mathcal{T}^*_{\langle \mathcal{A}, L \rangle}$, will be obtained marking every node in $\mathcal{T}_{\langle \mathcal{A}, L \rangle}$ as follows:

1. All leaves in $\mathcal{T}_{\langle \mathcal{A}, L \rangle}$ are marked as "U"s in $\mathcal{T}^*_{\langle \mathcal{A}, L \rangle}$.
2. Let Λ be an inner node of $\mathcal{T}_{\langle \mathcal{A}, L \rangle}$. Then Λ will be marked as "U" in $\mathcal{T}^*_{\langle \mathcal{A}, L \rangle}$ if and only if every one of its children is marked as "D". Otherwise, Λ will be marked as "D".

Definition 24 (Warranted Literal) A literal L is *warranted* if and only if for some argument $\langle \mathcal{A}, L\rangle$, the root of $\mathcal{T}^*_{\langle \mathcal{A},L\rangle}$ is marked as "U". We will say that $\mathcal{A}$ is a *warrant* for L.

The definition of *warrant* ensures the non-monotonicity of DELP. If a new rule is added to a program $\mathcal{P}$, the dialectical tree for $\langle \mathcal{A}, L\rangle$ may change and consequently, instead of being marked "U", its root may become marked as "D":

Example 7 Let $\mathcal{P}$ be the program: $\Pi_f = \{p\}, \Pi_r = \{p \rightarrow b\}, \Delta = \{b \succ f\}$ and let $\mathcal{A} = \{p, p \rightarrow b\}$, $\mathcal{B} = \{p, p \rightarrow b, b \succ f\}$.

It follows trivially that f is warranted by $\mathcal{B}$.

Now consider $\mathcal{P}'$, in which $\Pi'_f = \Pi_f$, $\Pi'_r = \Pi_r$ while $\Delta' = \Delta \cup \{p \succ \sim f\}$. Then, a new argument obtains, $\mathcal{C} = \{p, p \succ \sim f\}$. If furthermore, $\langle \mathcal{B}, f\rangle \prec \langle \mathcal{C}, \sim f\rangle$, then $\langle \mathcal{B}, f\rangle < \langle \mathcal{C}, \sim f\rangle$ so in $\mathcal{P}'$, $\mathcal{T}^*_{\langle \mathcal{B},f\rangle}$ is such that f is no longer warranted.

Comparison with the previous framework

In this section we want to compare the newly introduced system with the one originally presented in Simari and Loui, 1992 and García and Simari, 2004. In order to do this, we must distinguish the arguments as defined in Definition 14 of this paper from the original ones, which we'll call *d-arguments*. Here the letter d stands to emphazise that only the defeasible rules are regarded in this kind of arguments.

Definition 25 (d-arguments) Let $\mathcal{P}$=(Π, Δ) be a *delp*. A *d-argument* is a pair $\langle \mathcal{A}, L\rangle$ such that there is a brief D for L and $\mathcal{A} = \Delta \cap \mathsf{Arg}(D)$.

The definition above is not the original one but one can readily see it is equivalent to it, and uses the machinery introduced in this paper.

Now we can be recover d-arguments from our arguments.

Definition 26 (Equi-defeasibility) Let $\mathcal{P}$=(Π, Δ) be a *delp* and $\mathcal{A}rgs(\mathcal{P})$ the corresponding set of arguments. Let $\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle \in \mathcal{A}rgs(\mathcal{P})$. We will say that $\mathcal{A}_1$ and $\mathcal{A}_2$ are equi-defeasible if an only if $\mathcal{A}_1 \cap \Delta = \mathcal{A}_2 \cap \Delta$. We will denote this relation as $\mathcal{A}_1 \equiv \mathcal{A}_2$.

That is, $\mathcal{A}_1$ and $\mathcal{A}_2$ are equi-defeasible if they coincide on the set of defeasible rules they use. It is easy to see that equi-defeasibility is an equivalence relation.

There is a correspondence between arguments $\mathcal{A}$ supporting a literal L and d-arguments $\langle [\mathcal{A}], L\rangle$, where $[\mathcal{A}]$ denotes the equivalence class of $\mathcal{A}$ under the equi-defeasibility relation. We will identify the class of an argument $\mathcal{A}$, $[\mathcal{A}]$ with the set $\mathcal{A} \cap \Delta$. Together with d-arguments, we have the corresponding definitions of d-sub-argument, d-attack and d-defeating relation:

Definition 27 Let $d\mathcal{A}rgs(\mathcal{P})$ be the set of all d-arguments of a program $\mathcal{P}$. The d-argument $\langle \mathcal{A}_1, h\rangle$ is a d-sub-argument of $\langle \mathcal{A}_2, h'\rangle$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$.

Definition 28 For each literal h we define the binary relation R^d_h on $d\mathcal{A}rgs(\mathcal{P})$ by $\langle \mathcal{A}_1, h_1\rangle R^d_h \langle \mathcal{A}_2, h_2\rangle$ iff there exists $\langle \mathcal{A}, h\rangle \in d\mathcal{A}rgs(\mathcal{P})$ such that $\mathcal{A} \subseteq \mathcal{A}_1$ and h and h_2 disagree. We say in this case that the argument $\langle \mathcal{A}_1, h_1\rangle$ is d-attacked by $\langle \mathcal{A}_2, h_2\rangle$ at h or that $\langle \mathcal{A}_2, h_2\rangle$ *d-attacks* or *d-rebuts* $\langle \mathcal{A}_1, h_1\rangle$ at h. We also say that $\langle \mathcal{A}_2, h_2\rangle$ is a *counter-d-argument* of $\langle \mathcal{A}_1, h_1\rangle$.

Definition 29 $\langle \mathcal{A}_1, L_1\rangle$ is a *proper d-defeater* of $\langle \mathcal{A}_2, L_2\rangle$ (at literal h) if $\langle \mathcal{A}_2, L_2\rangle R^d_h \langle \mathcal{A}_1, L_1\rangle$ and there exists a sub-d-argument $\langle \mathcal{A}, L\rangle$ of $\langle \mathcal{A}_2, L_2\rangle$ such that $\langle \mathcal{A}, L\rangle \prec \langle \mathcal{A}_1, L_1\rangle$. We denote this by $\langle \mathcal{A}_2, L_2\rangle <^d \langle \mathcal{A}_1, L_1\rangle$.

$\langle \mathcal{A}_1, L_1\rangle$ is a *blocking d-defeater* of $\langle \mathcal{A}_2, L_2\rangle$ (at literal h) if $\langle \mathcal{A}_2, L_2\rangle R^d_h \langle \mathcal{A}_1, L_1\rangle$ and there exists a sub-d-argument $\langle \mathcal{A}, L\rangle$ of $\langle \mathcal{A}_2, L_2\rangle$ such that $\langle \mathcal{A}, L\rangle \not\prec \langle \mathcal{A}_1, L_1\rangle$ and $\langle \mathcal{A}_1, L_1\rangle \not\prec \langle \mathcal{A}, L\rangle$. Whenever this is the case we use the notation $\langle \mathcal{A}_2, L_2\rangle \approx^d \langle \mathcal{A}_1, L_1\rangle$.

We call *d-defeaters* of an argument to all proper and blocking d-defeaters. If $\langle \mathcal{A}_2, L_2\rangle$ is a d-defeater of $\langle \mathcal{A}_1, L_1\rangle$ we write $\langle \mathcal{A}_1, L_1\rangle \leq^d \langle \mathcal{A}_2, L_2\rangle$.

We now proceed to analyze three cases where we can see the difference between the formalisms in action.

Spurious Attacks

Example 8 Using the program from example 6, we see that the equivalence class of arguments $\mathcal{B}$ and $\mathcal{C}$ is the same so the d-arguments are $\langle [\mathcal{A}], b\rangle$, $\langle [\mathcal{B}], d\rangle$ and $\langle [\mathcal{B}], \sim b\rangle$. These two last d-arguments are each other's sub-d-arguments as well. This enables that $\langle [\mathcal{A}], b\rangle$ d-attack $\langle [\mathcal{B}], d\rangle$ at the literal $\sim b$, which does not appear in the deduction for d. Since $\sim b \notin C(\mathcal{B})$, this attack does not happen in the new framework.

Proper and blocking defeaters

Remark 7 Notice that our notion of sub-argument is more restrictive than that of (García and Simari 2004), since all that is required for $\mathcal{A}$ to be a sub-d-argument of $\mathcal{B}$ is that $\mathcal{A} \cap \Delta \subseteq \mathcal{B}$, while we now demand that $\mathcal{A} \subseteq \mathcal{B}$. This eliminates some undesirable effects like the one in the following example.

Example 9 Let $\mathcal{P}_2$ be the program with $\Pi_f = \{a\}$, $\Pi_r = \{b \rightarrow h, c \rightarrow h, b \wedge c \rightarrow h_1\}$, while $\Delta = \{a \succ b, a \succ c, a \succ \sim h\}$. Here we had that the d-argument $\langle \{a \succ \sim h\}, \sim h\rangle$ d-attacks $\langle \{a \succ b, a \succ c\}, h_1\rangle$ at literal h, but we have two sub-d-arguments at which the attack can happen: $\langle \{a \succ b\}, h\rangle$ and $\langle \{a \succ c\}, h\rangle$. If we let $\langle \{a \succ b\}, b\rangle \prec \langle \{a \succ \sim h\}, \sim h\rangle$ then we obtain:

$$\begin{gathered}
\langle \{a \succ b\}, b\rangle <^d \langle \{a \succ \sim h\}, \sim h\rangle \\
\langle \{a \succ b\}, h\rangle <^d \langle \{a \succ \sim h\}, \sim h\rangle \\
\langle \{a \succ b, a \succ c\}, h_1\rangle <^d \langle \{a \succ \sim h\}, \sim h\rangle \\
\langle \{a \succ c\}, c\rangle \approx^d \langle \{a \succ \sim h\}, \sim h\rangle \\
\langle \{a \succ c\}, h\rangle \approx^d \langle \{a \succ \sim h\}, \sim h\rangle \\
\langle \{a \succ b, a \succ c\}, h_1\rangle \approx^d \langle \{a \succ \sim h\}, \sim h\rangle
\end{gathered}$$

So in this case $\langle \{a \succ \sim h\}, \sim h\rangle$ was both a proper and a blocking d-defeater for the same argument and at the same literal.

Under the new definitions presented in this paper, the argument $\{a, a \succ b, a \succ c, b \wedge c \rightarrow h_1\}$ for h_1 has no sub-arguments for h, so the conflict does not occur.

Discerning the facts

When one uses d-arguments, the facts and strict rules used for a derivation are not recorded. This ends up meaning that all of them are implicit in each d-argument. One may as well consider a unique fact $f_{\mathcal{P}}$, logically equivalent to the conjunction of all the facts in a program $\mathcal{P}$, and obtain an equivalent program. With the proposed definition of arguments given here, the information on which facts are used to support a conclusion is recorded, and may later be used for comparing the relative weight of competing arguments. We think this adds to the flexibility of DELP, in particular when applied in the field of interacting agents (Capobianco, Chesñevar, and Simari 2005), where different agents may trust information on facts in different degrees.

Conclusions

This paper introduced a new version of DELP in which the central notion of argument has been modified. It contains now both defeasible *and* facts and strict rules. Problems that arose from restricting arguments to include only defeasible rules disappear. On one hand, now each argument supports a unique literal, so the *attack* relation is more transparently defined. On the other, the derived relation of *defeat* (be it proper or blocking) obtains in an unambiguous way.

These slight modifications of the DELP formalism ensure a leaner non-monotonic inference mechanism, mainly because each argument now supports a unique literal. Furthermore, the information carried by each argument is richer than in previous versions.

The new system has not yet been implemented, so we make no claims about its computational cost. We suspect, however, that it might be less than the classical version of DELP because of the closer correspondence between arguments an literals. This should ease the calculation of the attack and defeat relations. Manipulating lists (as the annotated derivations) may also be easier than dealing with sets.

Finally, the explicitness of the representation promoted here facilitates the interaction of several sources of information. This feature is particularly appropriate in multi-agent environments.

References

Alsinet, T.; Chesñevar, C. I.; Godo, L.; Sandri, S.; and Simari, G. 2008. Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification. *Int. J. Approx. Reasoning* 48(3):711–729.

Capobianco, M.; Chesñevar, C. I.; and Simari, G. R. 2005. Argumentation and the dynamics of warranted beliefs in changing environments. *Autonomous Agents and Multi-Agent Systems* 11(2):127–151.

Chesñevar, C. I., and Simari, G. R. 2001. Formalizing defeasible argumentation using a labelled deductive system. *Journal of Computer Science and Technology* 1(4):18–33.

Chesñevar, C. I.; Dix, J.; Stolzenburg, F.; and Simari, G. R. 2003. Relating defeasible and normal logic programming through transformation properties. *Theoretical Computer Science* 290(1):499–529.

Falappa, M. A.; Kern-Isberner, G.; and Simari, G. R. 2002. Explanations, belief revision and defeasible reasoning. *Artif. Intell.* 141(1/2):1–28.

García, A. J., and Simari, G. R. 1999. Parallel defeasible argumentation. *Journal of Computer Science and Technology Special Issue: Artificial Intelligence and Evolutive Computation. http://journal.info.unlp.edu.ar/* 1(2):45–57.

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

García, A. J. 2000. *Defeasible Logic Programming: Definition, Operational Semantics and Parallelism*. Ph.D. Dissertation, Computer Science and Engineering Department, Universidad Nacional del Sur, Bahía Blanca, Argentina.

Simari, G. R., and Loui, R. P. 1992. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence* 53:125–157.

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

Tohmé, F. A., and Simari, G. R. 2004. Negotiation among ddelp agents. In *Argumentation in Multi-Agent Systems*, 223–233.

Viglizzo, I. D.; Tohmé, F. A.; and Simari, G. R. 2008. An alternative foundation for delp: Defeating relations and truth values. In Hartmann, S., and Kern-Isberner, G., eds., *Foundations of Information and Knowledge Systems, 5th International Symposium, FoIKS 2008, Pisa, Italy, February 11-15, 2008. Proceedings*, number 4932 in Lecture Notes in Computer Science, 42–57.