Analyzing the Defeat Relation in Observation-based Defeasible Logic Programming

M. Capobianco[1] mc@cs.uns.edu.ar
C. I. Chesñevar[1,2] cic@eps.udl.es
G. R. Simari[1] grs@cs.uns.edu.ar

[1] Dpto. de Cs. e Ing. de la Computación – Universidad Nacional del Sur – Argentina
[2] Departament of Computer Science – Universitat de Lleida – España

Abstract

In the last decade several ways to formalize defeasible reasoning have been studied. A particular approach, *defeasible argumentation*, has been particularly successful to achieve this goal.

The inference process of argument-based systems is based on the interaction of arguments for and against certain conclusions. The relations of attack and defeat among arguments are key elements in these inference process. Usually a preference criterion is used to calculate the defeat relation to decide, in case of conflict, which argument is preferred over its contender.

Specificity is a domain independent principle that has been used in several formalisms. In this work we analyze the problem of incorporating specificity to characterize defeat in a particular argumentative framework, called *Observation Based Defeasible Logic Programming*. Since efficiency is an important issue in ODeLP, we have devised a new version of this criterion, that optimizes the computation of the defeat relation. We also present a formal proof to show that this new version is equivalent to the old one.

Keywords: knowledge representation, defeasible reasoning, argumentation.

1 Introduction

In the last decade several ways to formalize defeasible reasoning have been studied[6, 13, 19, 1, 18]. A particular approach, *defeasible argumentation*[5, 15], has been particularly successful to achieve this goal. Defeasible argumentation is built on the notions of arguments, counterarguments, attack and defeat. The inference process is based on the interaction of arguments for and against certain conclusions.

The relations of attack and defeat among arguments are key elements in argument-based frameworks. Attack (also called counterargument) denotes conflict among arguments. Evaluating conflicting pairs of arguments (that is, determining whether an attack among arguments is successful) is the function of the defeat relation. Usually a preference criterion is used to analyze which argument is preferred over its contender.

The field of defeasible argumentation has not yet reached a full maturity, and researchers disagree on many issues, such as which preference criterion should be used to choose among

competing arguments. Some authors believe that general principles for measuring arguments do not exist, and therefore rely on user-defined criteria, dependent on particular domains [15].

Specificity is a domain independent principle that has been used in several formalisms [12, 17, 7]. Specificity prefers arguments which are *more direct* or *more informed* (i.e., contain more specific information). It has been argued by some researchers that this criterion cannot be used as a general principle of common sense reasoning [15]. Nevertheless, specificity is the only known syntactic-based principle for deciding between conflicting arguments.

In this work we analyze the problem of incorporating specificity to characterize defeat in a particular argumentative framework, called *Observation Based Defeasible Logic Programming*, ODeLP [2]. Efficiency is an important issue in ODeLP, since this framework is designed for representing the knowledge of intelligent agents in real world applications. Computing specificity using domain knowledge is a demanding operation. Thus, we have devised a new version of this criterion, that optimizes the computation of the defeat relation. We present a formal proof to show that this new version is equivalent to the old one. We also analyze the role of the preference criterion in the context of argumentative frameworks.

The rest of the paper is structured as follows. First, in Section 2 we detail the main elements of the ODeLP formalism, particularly the defeat relationship. Section 3 discusses the proposed alternative for computing specificity in ODeLPand Section 4 contains the formal equivalence results between the traditional criterion and the new one. Finally Section 5 presents some conclusions and outlines future work.

2 The ODeLP Framework

Defeasible Logic Programming (DeLP) [7] provides capabilities for knowledge representation and reasoning that uses *defeasible argumentation* to decide between contradictory conclusions through a *dialectical analysis*. Codifying the knowledge base of the agent by means of a DeLP program provides a good trade-off between expressivity and implementability. DeLP has been used to model the behavior of a single intelligent agent in a *static* scenario (e.g. clustering algorithms [10] and intelligent web search [3]), but lacks the appropriate mechanisms to represent knowledge in dynamic environments, where agents must be able to perceive the changes in the world and integrate them into its existing beliefs [11].

The ODeLP framework aims at solving this problem by modeling perception and optimizing the inference system, to cope with time restrictions of dynamic environments. The language of ODeLP is based on the language of logic programming. Concepts like signature, alphabet and atoms are used in their description with their usual meaning. Literals are atoms that may be preceded by the symbol "$\sim$" denoting *strict* negation, as in ELP [9]. ODeLP programs are formed by *observations* and *defeasible rules*. Observations correspond to facts in the context of logic programming, and represent the knowledge an agent has about the world. *Defeasible rules* provide a way of performing tentative reasoning as in other argumentation formalisms [17, 14].

Definition 1. ([Observation]–[Defeasible Rule]) An *observation* is a ground literal L representing some fact about the world, obtained through the perception mechanism, that the agent believes to be correct. A *defeasible rule* has the form $L_0 \prec L_1, L_2, \ldots, L_k$, where L_0 is a literal and $L_1, L_2, \ldots, L_k$ is a non-empty finite set of literals. ■

Based on these elements, programs in ODeLP are defined as follows:

```
poor_performance(john).
sick(john).
good_performance(peter).
unruly(peter)
suspend(X) -< ~responsible(X).
suspend(X) -< unruly(X).
~suspend(X) -< responsible(X).
~responsible(X) -< poor_performance(X).
responsible(X) -< good_performance(X).
responsible(X) -< poor_performance(X),sick(X).
```

Figure 1: An ODeLP program for assessing the status of employees in a company

Definition 2. ([ODeLP Program])An *ODeLP program* is a pair $\langle \Psi, \Delta \rangle$, where Ψ is a finite set of observations and Δ is a finite set of defeasible rules. In a program $\mathcal{P}$, the set Ψ must be *non-contradictory* (*i.e.*, it is not the case that $Q \in \Psi$ and $\sim Q \in \Psi$, for any literal Q). ■

Example 2.1. Fig. 1 shows an ODeLP program. Observations describe that John has a poor performance at his job, John is currently sick, and Peter also has a good performance, but is unruly. Defeasible rules deal with the evaluation of the employees' performance, according with their responsibility in the job. If a given person is not responsible in his/her job then he/she could be suspended, a responsible person should not be suspended, and a person is hold as responsible (or not responsible) considering his/her performance in the company. ■

Given an ODeLP program $\mathcal{P}$, a query posed to $\mathcal{P}$ corresponds to a ground literal Q which must be supported by an *argument* [17, 7]. Arguments are built on the basis of a *defeasible derivation* computed by backward chaining applying the usual SLD inference procedure used in logic programming. Observations play the role of facts and defeasible rules function as inference rules. In addition to provide a proof supporting a ground literal, such a proof must be non-contradictory and minimal for being considered as an argument in ODeLP. Formally:

Definition 3. ([Argument – Sub-argument])Given a ODeLP program $\mathcal{P}$, an *argument* $\mathcal{A}$ for a ground literal l, also denoted $\langle \mathcal{A}, l \rangle$, is a subset of ground instances of the defeasible rules in $\mathcal{P}$ such that: (1) there exists a defeasible derivation for l from $\Psi \cup \mathcal{A}$, (2) $\Psi \cup \mathcal{A}$ is non-contradictory, and (3) $\mathcal{A}$ is minimal with respect to set inclusion in satisfying (1) and (2). Given two arguments $\langle \mathcal{A}_1, l_1 \rangle$ and $\langle \mathcal{A}_2, l_2 \rangle$, we will say that $\langle \mathcal{A}_1, l_1 \rangle$ is a *sub-argument* of $\langle \mathcal{A}_2, l_2 \rangle$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$. ■

As in most argumentation frameworks, arguments in ODeLP can attack each other. This situation is captured by the notion of *counterargument*. Defeat among arguments is defined combining the counterargument relation and a preference criterion "$\preceq$", that must be a partial order.

Definition 4. ([Counter-argument])An argument $\langle \mathcal{A}_1, l_1 \rangle$ *counter-argues* an argument $\langle \mathcal{A}_2, l_2 \rangle$ at a literal l if and only if there is a sub-argument $\langle \mathcal{A}, l \rangle$ of $\langle \mathcal{A}_2, l_2 \rangle$ such that l_1 and l are complementary literals. ■

Definition 5. ([Defeater])An argument $\langle \mathcal{A}_1, l_1 \rangle$ *defeats* $\langle \mathcal{A}_2, l_2 \rangle$ at a literal l if and only if there exists a sub-argument $\langle \mathcal{A}, l \rangle$ of $\langle \mathcal{A}_2, l_2 \rangle$ such that $\langle \mathcal{A}_1, l_1 \rangle$ counter-argues $\langle \mathcal{A}_2, l_2 \rangle$ at l, and either:

1. $\langle \mathcal{A}_1, l_1 \rangle$ is strictly preferred over $\langle \mathcal{A}, l \rangle$ according to the preference criterion "$\preceq$" (then $\langle \mathcal{A}_1, l_1 \rangle$ is a *proper defeater* of $\langle \mathcal{A}_2, l_2 \rangle$), or

2. $\langle \mathcal{A}_1, l_1 \rangle$ is unrelated to $\langle \mathcal{A}, l \rangle$ by "$\preceq$" (then $\langle \mathcal{A}_1, l_1 \rangle$ is a *blocking defeater* of $\langle \mathcal{A}_2, l_2 \rangle$).

■

Defeaters are arguments and may in turn be defeated. Thus, a complete dialectical analysis is required to determine which arguments are ultimately accepted. Such analysis results in a tree structure called *dialectical tree*, in which arguments are nodes labeled as undefeated (`U-nodes`) or defeated (`D-nodes`) according to a marking procedure. Formally:

Definition 6. ([Dialectical Tree]) The *dialectical tree* for an argument $\langle \mathcal{A}, l \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}, l \rangle}$, is recursively defined as follows:

1. A single node labeled with an argument $\langle \mathcal{A}, l \rangle$ with no defeaters (proper or blocking) is by itself the dialectical tree for $\langle \mathcal{A}, l \rangle$.

2. Let $\langle \mathcal{A}_1, l_1 \rangle, \langle \mathcal{A}_2, l_2 \rangle, \ldots, \langle \mathcal{A}_n, l_n \rangle$ be all the defeaters (proper or blocking) for $\langle \mathcal{A}, l \rangle$. The dialectical tree for $\langle \mathcal{A}, l \rangle$, $\mathcal{T}_{\langle \mathcal{A}, l \rangle}$, is obtained by labeling the root node with $\langle \mathcal{A}, l \rangle$, and making this node the parent of the root nodes for the dialectical trees of $\langle \mathcal{A}_1, l_1 \rangle, \langle \mathcal{A}_2, l_2 \rangle, \ldots, \langle \mathcal{A}_n, l_n \rangle$

■

Definition 7. ([Marking of the Dialectical Tree])Let $\langle \mathcal{A}_1, l_1 \rangle$ be an argument and $\mathcal{T}_{\langle \mathcal{A}_1, l_1 \rangle}$ its dialectical tree, then:

1. All the leaves in $\mathcal{T}_{\langle \mathcal{A}_1, l_1 \rangle}$ are marked as a `U-node`.

2. Let $\langle \mathcal{A}_2, l_2 \rangle$ be an inner node of $\mathcal{T}_{\langle \mathcal{A}_1, l_1 \rangle}$. Then $\langle \mathcal{A}_2, l_2 \rangle$ is marked as `U-node` iff every child of $\langle \mathcal{A}_2, l_2 \rangle$ is marked as a `D-node`. The node $\langle \mathcal{A}_2, l_2 \rangle$ is marked as a `D-node` if and only if it has at least a child marked as `U-node`.

■

Dialectical analysis may in some situations give rise to *fallacious argumentation* [16]. In ODeLP dialectical trees are ensured to be free of fallacies [2] by applying additional constraints when building *argumentation lines* (the different possible paths in a dialectical tree). A detailed analysis of these issues is outside the scope of this paper.

Given a query Q and an ODeLP program $\mathcal{P}$, we will say that Q is *warranted* wrt $\mathcal{P}$ iff there exists an argument $\mathcal{T}_{\langle \mathcal{A}, Q \rangle}$ such that the root of its associated dialectical tree $\mathcal{T}_{\langle \mathcal{A}, Q \rangle}$ is marked as a U-node.

Definition 8. ([Warrant]) Let $\mathcal{A}$ be an argument for a literal Q, and let $\mathcal{T}_{\langle \mathcal{A}, Q \rangle}$ be its associated dialectical tree. $\mathcal{A}$ is a *warrant* for Q if and only if the root of $\mathcal{T}_{\langle \mathcal{A}, Q \rangle}$ is marked as a `U-node`. ■

Solving a query Q in ODeLP accounts for trying to find a warrant for Q, as shown in the following example.

Example 2.2. Consider the program shown in Example 2.1, and let `suspend(john)` be a query wrt that program. The search for a warrant for `suspend(john)` will result in an argument $\langle \mathcal{A}, \texttt{suspend(john)} \rangle$ with two defeaters $\langle \mathcal{B}, \sim\texttt{suspend(john)} \rangle$ and $\langle \mathcal{C}, \texttt{responsible(john)} \rangle$, where

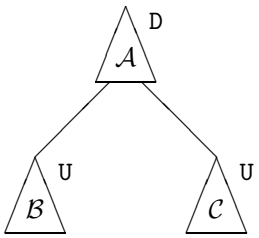

Figure 2: Dialectical tree from Example 2.2

- $\mathcal{A} = \{$`suspend(john)` $\prec$ `~responsible(john)`; `~responsible(john)` $\prec$ `poor_performance(john)`$\}$.
- $\mathcal{B} = \{$`~suspend(john)` $\prec$ `responsible(john)`; `responsible(john)` $\prec$ `poor_performance(john),sick(john)`$\}$.
- $\mathcal{C} = \{$`responsible(john)` $\prec$ `poor_performance(john),sick(john)`$\}$.

Using specificity as the preference criterion, $\langle \mathcal{B}, $`~suspend(john)`$\rangle$ is a blocking defeater for $\langle \mathcal{A}, $`suspend(john)`$\rangle$, and $\langle \mathcal{C}, $`responsible(john)`$\rangle$ is a proper defeater for $\langle \mathcal{A}, $`suspend(john)`$\rangle$. The associated dialectical tree is shown in Fig.2. The marking procedure determines that the root node $\langle \mathcal{A}, $`suspend(john)`$\rangle$ is a `D-node` and hence `suspend(john)` is not warranted. ■

3 Characterizing Defeat in OdeLP

In the past section we defined the defeat relation parameterized wrt to the preference criterion, like in the DeLP system. Nevertheless, we also propose a defeat criterion that can be used when no specific information about the domain is provided by the user. To this end, we have defined a particular version of specificity, adapted for ODeLP. In DeLP, an special version of this criterion was defined [7]. This version could be adapted for ODeLP in a simple manner:

Definition 9. (*ODeLP Specificity (1)*) Let $\mathcal{P}$ be an ODeLP program and $lit(\mathcal{P})$ the set of ground literals that can be derived from $\mathcal{P}$. An argument $\langle \mathcal{A}_1, l_1 \rangle$ is *strictly more specific than* an argument $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) iff: (1) For every $L \subseteq lit(\mathcal{P})$ holds that $L \cup \mathcal{A}_1 \vdash l_1$ and $L \nvdash l_1$ imply that $L \cup \mathcal{A}_2 \vdash l_2$; and (2) There exists $L' \subseteq lit(\mathcal{P})$ such that $L' \cup \mathcal{A}_2 \vdash l_2$, $l_2 \notin L'$ and $L' \cup \mathcal{A}_1 \nvdash l_1$. ■

To understand definition 9, lets analyze the first condition. As a general rule this holds for a non-empty set L, given that arguments do not contain facts, and L is said to *activate* $\mathcal{A}_1$. The restriction $L \nvdash l_1$ avoids trivial cases, forcing the use of L to derive l_1. Thus, definition 9 can be paraphrased as $\langle \mathcal{A}_1, l_1 \rangle$ is more specific than $\langle \mathcal{A}_2, l_2 \rangle$ iff for every set L such that L non-trivially activates $\langle \mathcal{A}_1, l_1 \rangle$ it holds that L non-trivially activates $\langle \mathcal{A}_2, l_2 \rangle$.

Example 3.1. The argument $\langle \mathcal{C}, $`responsible(john)`$\rangle$ in example 2.2 is more specific than $\langle \mathcal{D}, $`~responsible(john)`$\rangle$, where $\mathcal{D} = \{$`~responsible(john)` $\prec$ `poor_performance(john)`$\}$. Applying definition 9, every subset of $lit(\mathcal{P})$ that activates $\mathcal{C}$ also activates $\mathcal{D}$, but there is a subset of $lit(\mathcal{P})$ ($\{$`poor_performance(john)`$\}$) that activates $\mathcal{D}$ and does not activate $\mathcal{C}$. ■

Observation 3.1. There is an alternative formulation for this definition that can be stated as follows: Let $\mathcal{P}$ be a ODeLP program and $lit(\mathcal{P})$ the set of basic literals that can be derived from $\mathcal{P}$. An argument $\langle \mathcal{A}_1, l_1 \rangle$ is more especific that an argument $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) if and only if for every $L \subseteq lit(\mathcal{P})$ it holds that $L \cup \mathcal{A}_1 \vdash l_1$ and $L \nvdash l_1$ imply that $L \cup \mathcal{A}_2 \vdash l_2$. $\langle \mathcal{A}_1, l_1 \rangle$ is strictly more especific than $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) if and only if $\langle \mathcal{A}_1, l_1 \rangle \succeq \langle \mathcal{A}_2, l_2 \rangle$ and $\langle \mathcal{A}_2, l_2 \rangle \not\succeq \langle \mathcal{A}_1, l_1 \rangle$. This alternative definition has been used in related works [4, 8].

Definition 9 retains the underlying idea of specificity according to Poole's work [12] and it is technically correct. Nevertheless, it could have practical drawbacks. Computing the subsets of $lit(\mathcal{P})$ is a demanding operation. ODeLP is an argumentative formalism intended for real-world applications in dynamic environments, and therefore efficiency is an important issue. Hence, we have devised a new version of specificity, where a smaller number of activation sets must be considered.

Definition 10. (*ODeLP specificity (2)*) An argument $\langle \mathcal{A}_1, l_1 \rangle$ is strictly more specific than an argument $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) iff: (1) for every $L \subseteq literals(\mathcal{A}_1)$ it holds that $L \cup \mathcal{A}_1 \vdash l_1$ and $L \nvdash l_1$ imply that $L \cup \mathcal{A}_2 \vdash l_2$; and (2) there exists $L' \subseteq literals(\mathcal{A}_1)$ such that $L' \cup \mathcal{A}_2 \vdash l_2$, $l_2 \notin L'$ and $L' \cup \mathcal{A}_1 \nvdash l_1$. ■

In definition 10 only the subsets of the literals in the language generated by $\mathcal{A}_1$ must be taken into account. This optimizes the preference criterion and speeds up its implementation. In example 3.1 we can verify that the subset {`poor_performance(john)`} is also a subset of the literals in $\mathcal{C}$, and thus the example still complains with the new definition.

Observation 3.2. As in definition 9, definition 10 also has the following equivalent formulation: an argument $\langle \mathcal{A}_1, l_1 \rangle$ is more specific than an argument $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) if and only if for each $L \subseteq literals(\mathcal{A}_1)$ it holds that $H \cup \mathcal{A}_1 \vdash l_1$ and $L \nvdash l_1$ implies that $L \cup \mathcal{A}_2 \vdash l_2$. $\langle \mathcal{A}_1, l_1 \rangle$ is strictly more specific than $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) if and only if $\langle \mathcal{A}_1, l_1 \rangle \succeq \langle \mathcal{A}_2, l_2 \rangle$ and $\langle \mathcal{A}_2, l_2 \rangle \not\succeq \langle \mathcal{A}_1, l_1 \rangle$.

Definition 10 is computationally more efficient than definition 9. Still, before advising its use, we must be prove that both criteria are equivalent, that is to say, that definition 10 can be applied with no changes in the semantic associated to the ODeLP system.

4 Equivalence results

In this section we present a formal proof to the claim that both definitions of specificity, optimized and traditional, are equivalent. Note that in the proof of this claim we will use the alternative formulations of definitions 9 and 10, introduced respectively in observations 3.1 and 3.2, given that it simplifies the demonstrations.

First, we present two auxiliar results that will be used in the main proof. Proposition 4.1 shows that every activation set L of an argument $\mathcal{A}$ has a non empty intersection with $literals(A)$ (that is to say, both sets must have literals in common).

Proposition 4.1. For every non-trivial activation set L of an argument $\mathcal{A}$ for l, such that $L \cup \mathcal{A} \vdash l$ and $l \notin L$ it holds that $L \cap literals(\mathcal{A}) \neq \emptyset$ ■

Proof 1. Suppose that L is a non trivial activation set for an argument $\langle \mathcal{A}, l \rangle$, such that $L \cap literals(\mathcal{A}) = \emptyset$. Since L is a non-trivial activation set of $\mathcal{A}$, there must exist a defeasible derivation for l from $\mathcal{A} \cup L$. Nevertheless, it is not possible to build a derivation from $\mathcal{A} \cup L$ for a literal l_i such that $l_i \in literals(\mathcal{A})$. Then, $L \cup \mathcal{A} \nvdash l_i$ and in particular, $L \cup \mathcal{A} \nvdash l$. This contradiction arises from the initial supposition that $L \cap literals(\mathcal{A}) = \emptyset$. □

Given an argument $\mathcal{B}$ and a set of literals L, we only have to consider the literals in L that belongs to the rules in $\mathcal{B}$ to decide if L activates this argument. According to this, the following lemma shows that meaningful elements of an activation set for a given argument $\mathcal{B}$ are the literals that belong to the rules of that argument.

Lemma 4.1. Let $\langle \mathcal{A}, l \rangle$ be an argument built from a program $\mathcal{P}$, and let L_1 and L_2 be two set of literals that are included in the signature of $\mathcal{P}$ such that $L_1 \cap literals(\mathcal{A}) = L_2 \cap literals(\mathcal{A})$. Then L_1 activates $\mathcal{A}$ if and only if L_2 activates $\mathcal{A}$.

Proof 2. (1) ⇒ (2): Suppose that L_1 activates $\mathcal{A}$ and L_2 does not. Then there exists a derivation δ for l that can be built using the literals in L_1 and the rules in $\mathcal{A}$. Since $L_1 \cap literals(\mathcal{A}) = L_2 \cap literals(\mathcal{A})$ this same derivation can be used to obtain l from $L_2 \cup \mathcal{A}$, since every rule in $\mathcal{A}$ that can be applied using the literals in L_1 can also be applied using the literals in L_2. Therefore L_2 activates $\mathcal{A}$. This contradiction arose from the initial suposition that that L_1 activates $\mathcal{A}$ and L_2 does not. The proof in the other way ((2) ⇒ (1)) goes in a similar manner. □

Finally, we prove the equivalence among both formulations of specificity, using lemma 4.1 and proposition 4.1.

Theorem 4.1. Definitions 10 and 9 are equivalent. ■

Proof 3. As it was previously remarked in observation 3.1, definition 9 can also be stated as follows: an argument $\langle \mathcal{A}_1, l_1 \rangle$ is more especific that an argument $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) if and only if for every $L \subseteq lit(\mathcal{P})$ it holds that $L \cup \mathcal{A}_1 \sim l_1$ and $L \nvdash l_1$ imply that $L \cup \mathcal{A}_2 \sim l_2$.

Analogously, definition 10 can be stated as: an argument $\langle \mathcal{A}_1, l_1 \rangle$ is more specific than an argument $\langle \mathcal{A}_2, l_2 \rangle$ (noted as $\langle \mathcal{A}_1, l_1 \rangle \succ \langle \mathcal{A}_2, l_2 \rangle$) if and only if for each $L \subseteq literals(\mathcal{A}_1)$ it holds that $H \cup \mathcal{A}_1 \sim l_1$ and $L \nvdash l_1$ implies that $L \cup \mathcal{A}_2 \sim l_2$.

Thus, to show the equivalence between both definitions, we will the equivalence among these two claims, that is to say, we will show that if $\langle \mathcal{A}_1, l_1 \rangle$ and $\langle \mathcal{A}_2, l_2 \rangle$ are two arguments with respect to a program $\mathcal{P}$ and $lit(\mathcal{P})$ is the set of basic literals that can be derivated from $\mathcal{P}$, then the following conditions are equivalents:

1. For every $L \subseteq lit(\mathcal{P})$ it holds that $L \cup \mathcal{A}_1 \sim l_1$ and $L \nvdash l_1$ imply that $L \cup \mathcal{A}_2 \sim l_2$.

2. For every $L \subseteq literals(\mathcal{A}_1)$ it holds that $L \cup \mathcal{A}_1 \sim l_1$ and $L \nvdash l_1$ imply that $L \cup \mathcal{A}_2 \sim l_2$.

(1) ⇒ (2): In this case the proof is trivial, (the second clause is a particular case of the first one).
(2) ⇒ (1): Suppose by contradiction that this implication does not hold. Then, there must exists a set L_1 of basic literals such that $L_1 \not\subseteq literals(\mathcal{A}_1)$, L_1 non trivially activates $\mathcal{A}_1$ and L_1 does not activate $\mathcal{A}_2$. Applying proposition 4.1 we can conclude that $L_1 \cap literals(\mathcal{A}_1) \neq \emptyset$ (given that L_1 activates $\mathcal{A}_1$). Then there exists another activation set L_2 such that $L_2 =$

$L_1 \cap literals(\mathcal{A}_1)$ y $L_2 \subset literals(\mathcal{A}_1)$. Since $L_1 \cap literals(\mathcal{A}_1) = L_2 \cap literals(\mathcal{A}_1)$ we can apply lemma 4.1 and conclude that L_2 activates $\mathcal{A}_1$. Given our initial hypothesis, L_2 also activates $\mathcal{A}_2$ and since $L_2 \subset L_1$, L_1 must activate $\mathcal{A}_2$. This contradicts the initial supposition that L_1 does not activate $\mathcal{A}_2$. This contradiction arises from supposing the existence of the set L_1. Thus, (2) $\Rightarrow$ (1). □

5 Conclusions

Even though domain-dependent criteria can be useful in many situations, they set an extra burden on the user. In many domains, codifying the preference relation is not straightforward. Therefore, we have devised a new version of specificity tailored for the ODeLP system. The proposed definition is computationally attractive and provides a default criterion to be used in case no particular criterion has been defined. Moreover, equivalence results stated by the formal proof in section 4 assure us that definition 10 can be used in place of 9 without changing the semantics of the system.

The defeat relation is an important component of argumentative systems, and its definition directly affects the behavior of the formalism. If user-defined criteria are permitted, a set of standard conditions should be specified over it.

Some authors believe that the preference criterion should induce a partial order on the set of arguments (as [17, 16, 7]), while Vreeswijk [19] devised a different set of conditions. Nevertheless, developing an agreed set of adequate conditions is still an open problem. The partial order approach is reasonable, but could be too restrictive. As future work, it could be interesting to explore (using a particular framework as an study case) how different restrictions in the preference criterion vary the set of properties of the framework.

Acknowledgements

This research was partially supported by Projects TIC2001-1577-C03-01 and TIC2003-00950, by Ramón y Cajal Program (*Ministerio de Ciencia y Tecnología*, Spain), by *Secretaría General de Ciencia y Tecnología de la Universidad Nacional del Sur* (24/N016) and by *Agencia Nacional de Promoción Científica y Tecnológica* (PICT 2002 No. 13096). The first author is also partially supported by a fellowship of the *Comisión de Investigaciones Científicas* (CIC).

References

[1] Amgoud, L., and Cayrol, C. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence 34* (2002), 197–215.

[2] Capobianco, M. *Argumentación rebatible en entornos dinámicos*. PhD thesis, Universidad Nacional del Sur, Bahía Blanca, Argentina, June 2003.

[3] Chesñevar, C., and Maguitman, A. ArgueNet: An Argument-Based Recommender System for Solving Web Search Queries. In *Proc. of Intl. IEEE Conference on Intelligent Systems (IS-2004). Varna, Bulgaria (to appear)* (June 2004).

[4] Chesñevar, C. I., Dix, J., Stolzenburg, F., and Simari, G. Relating defeasible and normal logic programs through transformation properties. In *Proceedings of the 1st*

Workshop en Agentes y Sistemas Inteligentes (WASI), 6th Congreso Argentino de Ciencias de la Computaci n (CACIC) (Ushuaia, Oct. 2000), Universidad Nacional de la Patagonia, pp. 371–382.

[5] Chesñevar, C. I., Maguitman, A., and Loui, R. Logical Models of Argument. *ACM Computing Surveys 32*, 4 (Dec. 2000), 337–383.

[6] Dung, P. M. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming and n-Person Games. *Artificial Intelligence 77*, 2 (1995), 321–357.

[7] García, A., and Simari, G. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming 4*, 1 (2004), 95–138.

[8] García, A. J. *La Programación en Lógica Rebatible: Lenguaje, Semántica Operacional, y Paralelismo*. PhD thesis, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, Dec. 2000.

[9] Gelfond, M., and Lifschitz, V. Classical negation in logic programs and disjunctive databases. *New Generation Computing* (1991), 365–385.

[10] Gomez, S., and Chesñevar, C. A Hybrid Approach to Pattern Classification Using Neural Networks and Defeasible Argumentation. In *Proc. of Intl. FLAIRS Conference. Florida, USA (to appear)* (May 2004).

[11] Pollock, J. L. Taking Perception Seriously. In *Proceedings of the 1st International Conference on Autonomous Agents* (Feb. 1997), pp. 526–527.

[12] Poole, D. L. On the Comparison of Theories: Preferring the Most Specific Explanation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (1985), IJCAI, pp. 144–147.

[13] Prakken, H., and Sartor, G. A System for Defeasible Argumentation with Defeasible Priorities. In *Proceedings of the International Conference on Formal and Applied Practical Reasoning* (1996), Springer Verlag, pp. 510–524.

[14] Prakken, H., and Sartor, G. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics 7* (1997), 25–752.

[15] Prakken, H., and Vreeswijk, G. Logical systems for defeasible argumentation. In *Handbook of Philosophical Logic*, D. Gabbay, Ed., vol. 4. Kluwer Academic Publisher, 2002, pp. 219–318.

[16] Simari, G. R., Chesñevar, C. I., and García, A. J. The Role of Dialectics in Defeasible Argumentation. In *Proceedings of the XIV Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación* (Nov. 1994), pp. 111–121. http://cs.uns.edu.ar/giia.html.

[17] Simari, G. R., and Loui, R. P. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence 53*, 1–2 (1992), 125–157.

[18] Verheij, B. *Rules, Reasons and Arguments: formal studies of argumentation and defeat.* PhD thesis, Maastricht University, Department of Computer Science, Maastricht, Holanda, Dec. 1996.

[19] Vreeswijk, G. Abstract Argumentation Systems. *Artificial Intelligence 90*, 1–2 (1997), 225–279.