Characterizing Defeat in Observation-based Defeasible Logic Programming

M. Capobianco[1,*] mc@cs.uns.edu.ar
C. I. Chesñevar[1,2] cic@eup.udl.es

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

1 Introduction

In the last thirty years several ways to formalize defeasible reasoning have been studied. A particular approach, *defeasible argumentation* [3, 9], 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 [9].

Specificity is a domain independent principle that has been used in several formalisms. [7, 10, 4] 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 [9]. Nevertheless, specificity is the only known syntactic-based principle that uses domain-specific information to decide among 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) [1]. Efficiency is an important issues in ODeLP, since this framework has been defined for representing the knowledge of intelligent agents in real world applications. Computing specificity using domain knowledge is a demanding operation. Thus, have devised a new version of this criterion, that optimizes the computation of the defeat relation.

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

*Becaria de estudio de la Comisión de Investigaciones Científicas de la provincia de Bs. As.

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

alternative to computing specificity in ODeLP. Finally Section 4 present some conclusions and outlines future work.

2 Argumentation in ODeLP

Defeasible Logic Programming (DeLP) [4] provides a language 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 [5] and intelligent web search [2]), 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 [6].

The ODeLP framework aims at solving this problem by modeling perception and optimizing 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. 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 [10, 8]. These rules have 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, program in ODeLP are defined as follows:

Definition 1 ([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 poor performance. 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 should usually 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* [10, 4]. 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 2 ([Argument – Sub-argument])Given a ODeLP program $\mathcal{P}$, an *argument* $\mathcal{A}$ for a ground literal Q, also denoted $\langle \mathcal{A}, Q \rangle$, is a subset of ground instances of the defeasible rules in $\mathcal{P}$ such that: (1) there exists a defeasible derivation for Q 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 $\langle \mathcal{A}_1, Q_1 \rangle$ and $\langle \mathcal{A}_2, Q_2 \rangle$, we will say that $\langle \mathcal{A}_1, Q_1 \rangle$ is a *sub-argument* of $\langle \mathcal{A}_2, Q_2 \rangle$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$. ■

In this particular framework the notion of *counterargument* characterizes attack among arguments.

Definition 3 ([Counter-argument])An argument $\langle \mathcal{A}_1, Q_1 \rangle$ *counter-argues* an argument $\langle \mathcal{A}_2, Q_2 \rangle$ at a literal Q if and only if there is a sub-argument $\langle \mathcal{A}, Q \rangle$ of $\langle \mathcal{A}_2, Q_2 \rangle$ such that Q_1 and Q are complementary literals. ■

The defeat relation is defined combining the counterargument relationship and a preference criterion "$\preceq$".

Definition 4 ([Defeater])An argument $\langle \mathcal{A}_1, Q_1 \rangle$ *defeats* $\langle \mathcal{A}_2, Q_2 \rangle$ at a literal Q if and only if there exists a sub-argument $\langle \mathcal{A}, Q \rangle$ of $\langle \mathcal{A}_2, Q_2 \rangle$ such that $\langle \mathcal{A}_1, Q_1 \rangle$ counter-argues $\langle \mathcal{A}_2, Q_2 \rangle$ at Q, and either: (1) $\langle \mathcal{A}_1, Q_1 \rangle$ is strictly preferred over $\langle \mathcal{A}, Q \rangle$ according to the preference criterion "$\preceq$" (then $\langle \mathcal{A}_1, Q_1 \rangle$ is a *proper defeater* of $\langle \mathcal{A}_2, Q_2 \rangle$), or
(2) $\langle \mathcal{A}_1, Q_1 \rangle$ is unrelated to $\langle \mathcal{A}, Q \rangle$ by "$\preceq$" (then $\langle \mathcal{A}_1, Q_1 \rangle$ is a *blocking defeater* of $\langle \mathcal{A}_2, Q_2 \rangle$).
■

Leaving the defeat relation parameterized with respect to the preference relation gives more flexibility to the system, allowing the use of different criteria according to the particular domain under consideration.

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. An argument $\mathcal{A}$ for a literal Q is said to be a warrant for iff it is the root of a dialectical tree $\mathcal{T}_{\langle \mathcal{A}, Q \rangle}$ and 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 associated counterarguments $\langle \mathcal{B}, \sim\texttt{suspend(john)} \rangle$ and $\langle \mathcal{C}, \texttt{responsible(john)} \rangle$, where

- $\mathcal{A} = \{$`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)`$\}$.

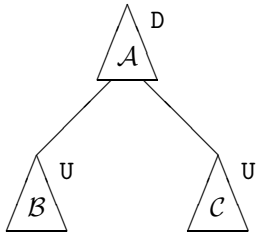

Figure 2: Dialectical tree from Example 2.2

Suppose the preference criterion favors the argument $\langle \mathcal{B}, \sim$`suspend(john)`$\rangle$ and the argument $\langle \mathcal{C},$ `responsible(john)`$\rangle$ over $\langle \mathcal{A},$ `suspend(john)`$\rangle$ Then the associated dialectical tree for `suspend(john)` 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 presented the defeat definition 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. We aim to define a particular version of specificity, adapted for ODeLP. In DeLP, an special version of this criterion was defined [4]. This version could be adapted for ODeLP in a simple manner:

Definition 5 (*ODeLP Specificity (1)*) Let $\mathcal{P}$ be an ODeLP program and *lit*P the set of ground literals that can be derived from $\mathcal{P}$. An argument $\langle \mathcal{A}_1, h_1 \rangle$ is *strictly more specific than* an argument $\langle \mathcal{A}_2, h_2 \rangle$ (noted as $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) iff: (1) For every $H \subseteq litP$ holds that $H \cup \mathcal{A}_1 \vdash h_1$ and $H \not\vdash h_1$ imply that $H \cup \mathcal{A}_2 \vdash h_2$; and (2) There exists $H' \subseteq litP$ such that $H' \cup \mathcal{A}_2 \vdash h_2$, $h_2 \notin H'$ and $H' \cup \mathcal{A}_1 \not\vdash h_1$. ■

To understand definition 5, lets analyze the first condition. As a general rule this holds for a non-empty set H, given that arguments do not contain facts, and H is said to *activate* $\mathcal{A}_1$. The restriction $H \not\vdash h_1$ avoids trivial cases, forcing the use of H to derive h_1. Thus, definition 5 can be paraphrased as $\langle \mathcal{A}_1, h_1 \rangle$ is more specific than $\langle \mathcal{A}_2, h_2 \rangle$ iff for every set H such that H non-trivially activates $\langle \mathcal{A}_1, h_1 \rangle$ it holds that H non-trivially activates $\langle \mathcal{A}_2, h_2 \rangle$.

Example 3.1 The argument $\langle \mathcal{C},$ `responsible(john)`$\rangle$ in example 2.2 is more specific than $\langle \mathcal{D}, \sim$`responsible(john)`$\rangle$, where $\mathcal{D} = \{\sim$`responsible(john)` $\prec$ `poor_performance(john)`$\}$. Applying definition 5, every subset of *lit*P that activates $\mathcal{C}$ also activates $\mathcal{D}$, but there is a subset of *lit*P ($\{$`poor_performance(john)`$\}$) that activates $\mathcal{D}$ and does not activate $\mathcal{C}$. ■

Definition 5 retains the underlying idea of specificity according to Poole's work [7]. Even though it is technically correct, it could have practical drawbacks. Computing the subsets of *lit*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 6 (*ODeLP specificity (2)*) An argument $\langle \mathcal{A}_1, h_1 \rangle$ is strictly more specific than an argument $\langle \mathcal{A}_2, h_2 \rangle$ (noted as $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) iff: (1) for every $H \subseteq literals(\mathcal{A}_1)$ it holds that $H \cup \mathcal{A}_1 \vdash h_1$ and $H \not\vdash h_1$ imply that $H \cup \mathcal{A}_2 \vdash h_2$; and (2) there exists $H' \subseteq literals(\mathcal{A}_1)$ such that $H' \cup \mathcal{A}_2 \vdash h_2$, $h_2 \notin H'$ and $H' \cup \mathcal{A}_1 \not\vdash h_1$. ■

In definition 6 only the subsets of the literals in $\mathcal{A}_1$ 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.

4 Conclusions and future work

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. 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. As future work we will study equivalence results between definition 5 and definition 6 with respect to the ODeLP domain.

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 author believe that the preference criterion should induce a partial order on the set of arguments [10, 4]. Nevertheless, developing an agreed set of adequate conditions is still an open problem.

References

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

[2] 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).

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

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

[5] 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).

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

[7] 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.

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

[9] 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.

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