A Methodology for Knowledge Representation in Defeasible Logic Programming[*]

Alejandro G. Stankevicius
Guillermo R. Simari
Laboratorio de Investigación y Desarrollo en Inteligencia Artificial
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Buenos Aires - ARGENTINA
e-mail: {ags,grs}@cs.uns.edu.ar

Abstract

Humans have always been intrigued by their ability to *reason*. We have constantly attempted to emulate this process, trying almost everything, from physiological explanations, to sociological accounts. The approach with possibly the longest tradition conceives this process as a mere manipulation of *symbols*. Yet, symbolic reasoning cannot be applied directly over the problem at hand: we require that the knowledge about that domain be also described symbolically, where this description is in turn the outcome of the process called *Knowledge Representation*.

Defeasible Logic Programming (DeLP) is a formalism that by combining Logic Programming with Defeasible Argumentation is able to represent incomplete and potentially contradictory information. Its ability to represent this kind of informartion make it suitable for describing many real world situations, where its inference engine can then later be used to solve concrete problems in those scenarios. In this article we propose a formal methodology, striving to standardize the process of knowledge representation in DeLP, that defines a set of guidelines to be used during this key task.

Keywords: nonmonotonic reasoning, defeasible reasoning, argumentation theories, knowledge representation

1 INTRODUCTION

Humans have always been intrigued by their ability to *reason*, even before the invention of computers. For decades, we have been attempting to emulate the process of reasoning, but it proved to be quite hard to capture in a practical sense. Scientist have tried almost everything, from physiological explanations (involving even quantum physics [12]), to sociological accounts (for instance, problem solving using the ant-colony metaphor [1]). Out of those approaches, the one with possibly the longest tradition, going back at least to the golden age in classical Greece, conceives this process as a mere manipulation of *symbols* [10]. This great insight produced nowadays the most usefull and versatile creations at our disposal (computers included). However, symbolic reasoning cannot be applied directly over the problem at hand: we require that the knowledge about that particular domain be also

[*]Partially supported by Agencia Nacional de Promoción Científica y Tecnológica (PAV 2003 Nro. 076, PICT 2002 Nro. 13096, PICT 2003 Nro. 15043), CONICET (*Consejo Nacional de Investigaciones Científicas y Técnicas de la República Argentina*), and CIC (*Comisión de Investigaciones Científicas de la Provincia de Buenos Aires*).

described symbolically, where this description is in turn the outcome of the process called *Knowledge Representation*.

From a knowledge engineer standpoint, knowledge representation is the process upon which they classifies all the information present in a given domain into two categories: the core characteristics of the scenario under study, and the rest of the knowledge, that is, those other aspects deemed not relevant considering the problem at hand. Although this may look simple or straight, this is by no means a trivial task. Sometimes, it is more an art than a science, provided that what can be derived applying this kind of reasoning is usually intrinsically intertwined with the way knowledge is being represented. When dealing with this, what we actually require is a set of guidelines assuring us that no core characteristics may mistakenly be ignored. In other words, we are looking for a *methodology*.

Defeasible Logic Programming (DeLP) is a formalism that by combining Logic Programming with Defeasible Argumentation is able to represent incomplete and potentially contradictory information. Ideas borrowed from defeasible argumentation such as representing defeasible reasons as arguments or performing a full dialectical analysis before answering queries are carefully added to a knowledge representation language featuring PROLOG-like rules. Its ability to represent incomplete information makes it suitable for describing many real world situations, where its inference engine can then later be used to solve concrete problems in those scenarios. However, often times the results one may obtain by adopting DeLP are directly related to the decisions (or lack thereof) made during this initial phase of knowledge representation. As a consequence, in this article we propose a formal methodology, striving to standardize the process of knowledge representation in DeLP, that defines a set of guidelines to be used during this key task.

In what follows, Sect. 2 briefly introduces the inner working of DeLP. Then, Sect. 3 outlines the proposed methodology for knowledge representation, illustrating its use with some examples from the literature. Finally, Sect. 4 presents the conclusions obtained.

2 DEFEASIBLE LOGIC PROGRAMMING

This section briefly introduces DeLP's essentials following its most recent formulation [6] (we refer the reader looking for a more comprehensive presentation to its original formulation [5]).

The DeLP language is defined in terms of three disjoint sets: *facts*, *strict rules*, and *defeasible rules*. Literals can be ground atoms ($e.g.$, A), or their strong negation ($e.g.$, $\sim A$). Facts are simply literals. Strict rules are ordered pairs $L_0 \leftarrow L_1, \ldots, L_n$ whose first component, L_0, is a literal, and whose second component, $L_1, \ldots, L_n$, is a finite non-empty set of literals. In a like manner, a defeasible rule is an ordered pair $L_0 \prec L_1, \ldots, L_n$ whose first component, L_0, is a literal, and whose second component, $L_1, \ldots, L_n$, is a finite non-empty set of literals. Syntactically, the symbol '$\prec$' is all that distinguishes a defeasible rule from a strict one. Pragmatically, defeasible rules are used to represent defeasible knowledge ($i.e.$, tentative information that can be used as long as nothing is posed against it), whereas strict rules are used to represent non-defeasible knowledge ($i.e.$, incontrovertible information). Observe that both strict and defeasible rules induce a meta-relation between set of literals, so the symbols '$\leftarrow$' and '$\prec$' have no interaction with the literals.

In this formalism, the state of the world is modelled as a *Defeasible Logic Program* ($de.l.p$), essentially a possibly infinite set of facts, strict rules and defeasible rules. In a given $de.l.p$ $\mathcal{P}$, the subset of facts and strict rules is referred to as Π, and the subset of defeasible rules as Δ. When required, the $de.l.p$ $\mathcal{P}$ can also be noted as (Π, Δ). Since the set Π represent non-defeasible information, it is assumed that it is non-contradictory, that is, no pair of complementary literals can be derived from it at the same time. As usual, the literals that may be derived are obtained chaining as many rules and facts as required.

Definition 1. (strict and defeasible derivations) [6]
Let $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$, and L a ground literal. A *defeasible derivation* of L from $\mathcal{P}$, noted $\mathcal{P} \mathrel{|\!\!\sim} L$, is a finite sequence $L_1, L_2, \dots, L_n = L$ of ground literals, such that for every L_i, $1 \leq i \leq n$:

- $L_i \in \Pi$ (L_i is one of the facts in Π), or
- there exists a strict or defeasible rule in $\mathcal{P}$, with head L_i and body $B_1, B_2, \dots, B_m$, such that for every B_j, $1 \leq j \leq m$, there exists a k, $1 \leq k < i$, such that $B_j = L_k$.

Those defeasible derivations that only use facts and strict rules are also called *strict derivations*. ■

Since modelling incomplete infomation usually gives rise to conflicting conclusions, we should not accept at the same time all the literals that may be defeasible derived from a given program. Only those derivations that constitute *logical arguments* are to be considered. In this formalism, an argument is a tentative piece of reasoning supporting a given conclusion, that satisfies the following restrictions:

Definition 2. (argument structure) [6]
Let h be a literal, and $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$. We say that $\langle \mathcal{A}, h \rangle$ is an *argument structure* for h, if, and only if, $\mathcal{A}$ is a set of defeasible rules from $\mathcal{P}$ ($i.e.$, $\mathcal{A} \subseteq \Delta$), such that:

- there exists a defeasible derivation for h from $\Pi \cup \mathcal{A}$,
- the set $\Pi \cup \mathcal{A}$ is non-contradictory, and
- $\mathcal{A}$ is minimal with respect to set inclusion ($i.e.$, no $\mathcal{A}' \subset \mathcal{A}$ satisfies the previous conditions).

■

Argument structures only contain the defeasible rules used in order to defeasible derive a given conclusions. Even though one may be tempted to add the facts and strict rules used too, recall that only defeasible rules may later be challenged (given that facts and strict rules, as such, cannot be questioned). Also, argument structures are required to be minimal with respect to set inclusion. This restriction, often forgotten in other theories of argumentation, is paramount in importance, since we do not want to allow as sensible arguments those that include more “defeasible information” than what is strictly required. Should we fail to do so, valid arguments might end up rejected on the basis of a certain superfluous assumption that happens to be too weak. In a sense, this restriction is embodying the sound principle that the strength of an argument must be related to the amount of defeasible information upon which it depends.

Example 1. Let us consider the $de.l.p$ $\mathcal{P}_1 = (\Pi_1, \Delta_1)$, where:

Π_1	Δ_1
`bird(X) <- penguin(X)`	`flies(X) -< bird(X)`
`~penguin(X) <- ~bird(X)`	
`~flies(X) <- penguin(X)`	
`penguin(pengo)`	
`bird(tweety)`	

According to $\mathcal{P}_1$, $\langle \mathcal{A}, flies(tweety) \rangle$ constitutes an argument structure for $flies(tweety)$, where $\mathcal{A} = \{flies(tweety) \prec bird(tweety)\}$, but, at the same time, it is impossible to formulate an argument structure for $flies(pengo)$, provided that the set of defeasible rules $\{flies(pengo) \prec bird(pengo)\}$ allowing the derivation of $flies(pengo)$ is in conflict with Π_1, since $\Pi_1 \mathrel{|\!\!\sim} {\sim}flies(pengo)$). Therefore, $\langle \emptyset, {\sim}flies(pengo) \rangle$ constitutes a valid argument structure for ${\sim}flies(pengo)$.

Having an argument structure for a certain conclusion is not enough to warrant it, since there may also exist another argument structure for a conflicting conclusion.

Definition 3. (disagreement) [6]
Let $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$ and let L_1 and L_2 be two literals. We say that L_1 *disagrees* with L_2, if, and only if, the set $\Pi \cup \{L_1, L_2\}$ is contradictory. ■

This relation between literals is generalized to argument structures using the auxiliary notion of *sub-argument structure*: we say that $\langle \mathcal{A}', h' \rangle$ is a sub-argument structure of $\langle \mathcal{A}, h \rangle$ if, and only if, $\mathcal{A}' \subseteq \mathcal{A}$.

Definition 4. (counter-argumentation) [6]
Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures. We say that $\langle \mathcal{A}_1, h_1 \rangle$ *counter-argues* (also *rebuts* or *attacks*) $\langle \mathcal{A}_2, h_2 \rangle$ at h if, and only if, there exists a sub-argument structure $\langle A, h \rangle$ from $\langle \mathcal{A}_2, h_2 \rangle$ such that h and h_1 disagrees. ■

Considering that $\langle \mathcal{A}, h \rangle$ is one of its own sub-argument structures, this relation may become symmetric, and as such, might not be able to tell appart successful attacks from those that are not. Thus, it must be further refined with the aid of an argument comparison criterion '$\succ$'. By doing so, two types of attacks can be distinguished:

Definition 5. (proper defeater) [6]
Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures. We say that $\langle \mathcal{A}_1, h_1 \rangle$ is a *proper defeater* for $\langle \mathcal{A}_2, h_2 \rangle$ at the literal h if, and only if, there exists a sub-argument structure $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$, such that $\langle \mathcal{A}_1, h_1 \rangle$ counter-argues $\langle \mathcal{A}_2, h_2 \rangle$ at h, and also $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}, h \rangle$ ($i.e.$, $\langle \mathcal{A}, h \rangle$ is preferred over $\langle \mathcal{A}_1, h_1 \rangle$). ■

Definition 6. (blocking defeat) [6]
Let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures. We say that $\langle \mathcal{A}_1, h_1 \rangle$ is a *blocking defeater* for $\langle \mathcal{A}_2, h_2 \rangle$ at the literal h if, and only if, there exists a sub-argument structure $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$, such that $\langle \mathcal{A}_1, h_1 \rangle$ counter-argues $\langle \mathcal{A}_2, h_2 \rangle$ at h, and neither $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}, h \rangle$ nor $\langle \mathcal{A}, h \rangle \succ \langle \mathcal{A}_1, h_1 \rangle$ ($i.e.$, $\langle \mathcal{A}, h \rangle$ and $\langle \mathcal{A}_1, h_1 \rangle$ are unrelated according $\succ$). ■

This distintion between defeaters will play a role in the upcoming dialectical analysis. In a sense, a proper defeat denotes a more straightforward form of defeat than a blocking one. When required, we say that an argument structure *defeats* another when the former is either a proper or blocking defeater of the latter.

Regarding the argument-comparison criterion, DeLP provides a modular design where the knowledge engineer can also specify which criterion is better suited for the domain at hand. Not whitstanding, many attractive results have been achieved using a particular criterion called *specificity,* initially introduced by Poole [13], later adapted for DeLP use as follows:

Definition 7. (generalized specificity) [6]
Let $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$, and let Π_G be the set of all the strict rules in Π. Also, let $\mathcal{F}$ be the set of all the literals that have a defeasible derivation from $\mathcal{P}$ ($i.e.$, $\mathcal{F}$ is a set of facts). Finally, let $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ be two argument structures obtained from $\mathcal{P}$. We say that $\langle \mathcal{A}_1, h_1 \rangle$ is *strictly more specific* than $\langle \mathcal{A}_2, h_2 \rangle$ if, and only if, the following conditions hold:

1. For all $H \subseteq \mathcal{F}$, whenever $\Pi_G \cup H \cup \mathcal{A}_1 \mathrel{|\!\!\!\sim} h_1$ ($i.e.$, H activates h_1) and $\Pi_G \cup H \not\mathrel{|\!\!\!\sim} h_1$ ($i.e.$, a non trivial activation), then it must be the case that $\Pi_G \cup H \cup \mathcal{A}_2 \mathrel{|\!\!\!\sim} h_2$ ($i.e.$, H also activates h_2).

2. There must exist a $H' \subseteq \mathcal{F}$, such that $\Pi_G \cup H' \cup \mathcal{A}_2 \mathrel{|\!\sim} h_2$ ($i.e.$, H' activates h_2) and $\Pi_G \cup H' \not\mathrel{|\!\sim} h_2$ ($i.e.$, a non trivial activation) , where $\Pi_G \cup H' \cup \mathcal{A}_1 \not\mathrel{|\!\sim} h_1$ ($i.e.$, H' does not activate h_1).

■

This criterion essentially reconciles two principles: on the one hand, it favors those argument structures that are more informed, and on the other hand, it also favors those argument structures involving shorter defeasible derivations.

Example 2. Let us consider the $de.l.p$ $\mathcal{P}_2 = (\Pi_2, \Delta_2)$, where:

Π_2	Δ_2
`bird(X) <- penguin(X)`	`flies(X) -< bird(X)`
`~penguin(X) <- ~bird(X)`	`~flies(X) -< weak(X)`
`bird(tweety)`	`~flies(X) -< penguin(X)`
`weak(tweety)`	
`penguin(pengo)`	

In the context of $\mathcal{P}_2$, it is possible to formulate the following argument structures:

$\langle \mathcal{A}_1, flies(pengo) \rangle$, where $\mathcal{A}_1 = \{flies(pengo) \prec bird(pengo)\}$.

$\langle \mathcal{A}_2, {\sim}flies(pengo) \rangle$, where $\mathcal{A}_2 = \{{\sim}flies(pengo) \prec penguin(pengo)\}$.

$\langle \mathcal{A}_3, flies(tweety) \rangle$, where $\mathcal{A}_3 = \{flies(tweety) \prec bird(tweety)\}$.

$\langle \mathcal{A}_4, {\sim}flies(tweety) \rangle$, where $\mathcal{A}_4 = \{{\sim}flies(tweety) \prec weak(tweety)\}$.

Using generalized specificity as the argument-comparison criterion, $\langle \mathcal{A}_2, {\sim}flies(pengo) \rangle$ becomes a proper defeater of $\langle \mathcal{A}_1, flies(pengo) \rangle$, since the former is strictly more specific then the latter. Also, $\langle \mathcal{A}_3, flies(tweety) \rangle$ and $\langle \mathcal{A}_4, {\sim}flies(tweety) \rangle$ block each other, since these argument structures are unrelated under the chosen criterion.

In this formalism, a given literal is deemed warranted if we are able to find an argument structure for it that remains undefeated after considering all its potential defeaters. Now, since defeaters are in turn argument structures as well, there may exists defeaters for these defeaters, and so on. This sequence of argument structures, each one defeating the previous one, is called in this context *argumentation line*, in the sense that this exchange of reasons seems to be exploring a given aspect of the controversy.

Definition 8. (argumentation line) [6]
Let $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$ and let $\langle \mathcal{A}_0, h_0 \rangle$ be an argument structure. We say that the sequence of argument structures $[\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \ldots]$ constitutes an *argumentation line* for $\langle \mathcal{A}_0, h_0 \rangle$, noted $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$, if, and only if, every argument structure $\langle \mathcal{A}_i, h_i \rangle$, $i \geq 1$, in $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$ is such that it defeats its immediate predecessor $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$. ■

According to this definition, argumentation lines may be infinite. This concern is addressed restricting the argumentation lines that may appear during the dialectical analysis of a certain claim, considering that not every exchange of arguments actually constitutes a valid pattern of reasoning. For instance, circular argumentation is a particular case of *fallacious reasoning* which should be avoided at all cost. The occurrence of these undesired situations is prevented imposing a set of conditions over the potential argumentation lines, distinguishing those that do not incur in any sort of fallacious reasoning as being *acceptable*. In order to do so, we must be able to tell appart those argument structures supporting the initial claim from those that instead interfere with it.

Definition 9. (support, interference) [6]
Let $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle} = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \ldots]$ be an argumentation line for $\langle \mathcal{A}_0, h_0 \rangle$. We say that the argument structures occupying the odd positions in $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$ constitute the *set of supporting argument structures* of $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$, noted $\Lambda_S^{\langle \mathcal{A}_0, h_0 \rangle}$, and that the argument structures occupying the even positions in $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$ constitute the *set of interfering argument structures* of $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$, noted $\Lambda_I^{\langle \mathcal{A}_0, h_0 \rangle}$ ■

Another concept required to formally define what constitute an acceptable argumentation line is the notion of *concordance,* a generalization of the notion of being consistent with the strict knowledge.

Definition 10. (concordance) [6]
Let $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$ and $S = \{\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle\}$ be a set of argument structures obtained from $\mathcal{P}$. We say that S is *concordant* if, and only if, the set $(\bigcup_{i=1}^{n} \mathcal{A}_i) \cup \Pi$ is non-contradictory. ■

All these notions are orchestrated together in the formal definition of what constitutes an acceptable argumentation line:

Definition 11. (acceptable argumentation line) [6]
Let $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle} = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \ldots]$ be an argumentation line for $\langle \mathcal{A}_0, h_0 \rangle$. We say that $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$ constitutes an *acceptable argumentation line* if, and only if, the following conditions hold:

1. $\Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$ is finite.
2. The sets $\Lambda_S^{\langle \mathcal{A}_0, h_0 \rangle}$ of supporting argument structures and $\Lambda_I^{\langle \mathcal{A}_0, h_0 \rangle}$ of interference argument structures are concordant.
3. No argument structure $\langle \mathcal{A}_i, h_i \rangle \in \Lambda^{\langle \mathcal{A}_0, h_0 \rangle}$ is a sub-argument structure of a previous argument structure $\langle \mathcal{A}_j, h_j \rangle$, with $j < i$.
4. If $\langle \mathcal{A}_i, h_i \rangle$, $i \geq 2$, is a blocking defeater of $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$, then $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$ must be a proper defeater of $\langle \mathcal{A}_{i-2}, h_{i-2} \rangle$.

■

Each argumentation line only explores a particular aspect of the controversy about the final state of the initial claim. The complete dialectical analysis, which encompasses multiple argumentation lines, is structured as a tree by virtue of the following recursive characterization:

Definition 12. (dialectical tree) [6]
Let $\langle \mathcal{A}_0, h_0 \rangle$ be an argument structure about h_0. We say that the *dialectical tree* for $\langle \mathcal{A}_0, h_0 \rangle$, noted $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, can be obtained as follows:

1. The root node must be labelled $\langle \mathcal{A}_0, h_0 \rangle$.
2. Let $[\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ be the sequence of labels along a certain branch from the root of the tree up to a node labelled $\langle \mathcal{A}_n, h_n \rangle$, an let $\langle \mathcal{B}_1, q_1 \rangle, \langle \mathcal{B}_2, q_2 \rangle, \ldots, \langle \mathcal{B}_k, q_k \rangle$ be all the defeaters of $\langle \mathcal{A}_n, h_n \rangle$. Then, for every defeater $\langle \mathcal{B}_i, q_i \rangle$ of $\langle \mathcal{A}_n, h_n \rangle$, $1 \leq i \leq k$, such that $[\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \langle \mathcal{B}_i, q_i \rangle]$ happens to be an acceptable argumentation line for $\langle \mathcal{A}_0, h_0 \rangle$, a new node labelled $\langle \mathcal{B}_i, q_i \rangle$ must be added as a child of the node labelled $\langle \mathcal{A}_n, h_n \rangle$.

■

Finally, we can resort to the following bottom-up marking to determine the outcome of the dialectical analysis just structured as a tree:

Definition 13. (marking of a dialectical tree) [6]
Let $\langle \mathcal{A}, h \rangle$ be an argument structure and $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ its corresponding dialectical tree. The *marking* of $\mathcal{T}_{\langle \mathcal{A},h \rangle}$, noted $\mathcal{T}^{\star}_{\langle \mathcal{A},h \rangle}$, can be obtained as follows:

- All the leaves of $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ are marked **U** in $\mathcal{T}^{\star}_{\langle \mathcal{A},h \rangle}$.
- Let N be a inner node of $\mathcal{T}_{\langle \mathcal{A},h \rangle}$. This node should be marked **U** if, and only if, all its children nodes are marked **D**. If that is not the case, it should be marked **U**.

■

At last, this marking allows us to characterize the set of literals sanctioned by a given $de.l.p$, which constitutes the semantics of this formalism.

Definition 14. (warrant) [6]
Let $\mathcal{P} = (\Pi, \Delta)$ be a $de.l.p$, and let h be a literal. We say that h is *warranted* if, and only if, there exists an argument structure $\langle \mathcal{A}, h \rangle$ for h, such that the root of its marked dialectical tree $\mathcal{T}^{\star}_{\langle \mathcal{A},h \rangle}$ bears the mark **U**. ■

After this succinct overview of DeLP, the next section develops the proposed methodology for putting this framework to a good use.

3 A METHODOLOGY FOR KNOWLEDGE REPRESENTATION

This section proposes a set of guidelines describing how different situations can be modeled within DeLP. We begin by addressing the representation of monotonic knowledge in Sect. 3.1, to then consider non-monotonic knowledge in Sect. 3.2.

3.1 Representing Monotonic Knowledge

Monotonic reasoning has been under study for hundreds, even thousands of years. Hence, its corresponding knowledge representation is also well understood. In what follows, we consider in detail those aspects of monotonic knowledge representation which can be straightforwardly expressed within DeLP.

3.1.1 Relational Databases

To begin with, the most essential form of symbolic knowledge one may want to represent is a set atomic formulas, where in turn each formula states that a certain relation holds over a given tuple of objects. This kind of knowledge can be represented in DeLP as a set of facts, much in the same way this knowledge was represented in PROLOG. Bear in mind that the nature of these relations, or the origin of the objects they relate cannot be inspected: it should be determined by the knowledge engineer for each concrete scenario.

Remark 1. *Let r be a finite relation over a certain domain. This relation can be modeled in $de.l.p$ by adding a new fact $r(\overline{X})$ for every $\overline{X} \in r$.*

The reader might notice only finite relations can be modelled following this approach. Then again, the same holds for relational databases and PROLOG programs.

Example 3. The following $de.l.p$ models the relation 'weekend day' defined over the days of the week:

Π	Δ
`weekend(saturday)`	
`weekend(sunday)`	

3.1.2 Class Inclusion

Another important piece of monotonic knowledge usually worth modeling is the inclusion of concepts or categories. From Aristotle's syllogisms to modern day inheritance hierarchies, being able to represent class inclusion has always been deemed crucial. The classical approach consists in modelling the inclusion of a concept A into the concept B with a material implication $A(\overline{X}) \rightarrow B(\overline{X})$, denoting that all the instances of the former are also instances of the latter. We can follow a simmilar approach in DeLP, as long as we take into consideration that DeLP does not have material implication but inference rules (meta-relations between set of formulas to be precise). That is to say:

Remark 2. *Let A and B be two concepts, already modelled as relations among objects, such that A is included in B. This knowledge can be expressed in a $de.l.p$ by adding the strict rules $b(\overline{X}) \leftarrow a(\overline{X})$ and $\sim a(\overline{X}) \leftarrow \sim b(\overline{X})$.*

Much to our surprise, this result made us aware that all this time we have been modelling class inclusion the wrong way. Considering that this formalism is a refinement of Simari-Loui's system [15], where strict rules were material implications, we incorrectly kept modelling this notion with the first rule alone, when in fact two rules were actually required.

Example 4. The following $de.l.p$ models the fact that humans are mammals.

Π	Δ
`mammal(X) <- human(X)`	
`~human(X) <- ~mammal(X)`	

3.1.3 Logic Programming

The paradigm of logic programming has been throughly explored as a tool for knowledge representation almost since its initial conception. Many attractive expert systems have been developed in PROLOG, the most well-known exponent of this paradigm. In these systems, the knowledge is represented through a standard PROLOG program. These programs, by extending a finite set of facts with general rules, allow the representation of infinite relations, something not possible using standard relational databases. These kind of knowledge can also be expresed in DeLP, since any standard PROLOG program can be reformulated as a $de.l.p$ as follows:

Remark 3. *Let P be a definite* PROLOG *program. The knowledge represented by the program P can be expressed as a $de.l.p$ by virtue of the same set of facts and (strict) rules.*

The previous remark rests upon the following proposition, relating the answers a given PROLOG program returns with DeLP's notion of warranted literals.

Proposition. *Let P be a definite* Prolog *program. Then, q is a ground query entailed from P in* Prolog *if, and only if, q is a warranted literal from (Π, Δ), where $\Pi = P$ and $\Delta = \emptyset$.*

Proof. ($\Rightarrow$)
If q is entailed from P, there must exist a SLD-derivation of q from P. Let $L_1, L_2, \ldots, L_n = q$ be the sequence of literals compossing that SLD-derivation. Observe that this same sequence can also be used to construct a strict derivation of q from P, provided the same restrictions apply to both notions.[1] Therefore, $\langle \emptyset, q \rangle$ is a valid argument structure for q, if we take into account that q can be defeasibly derived from $P \cup \emptyset$, $P \cup \emptyset$ is non-contradictory since Prolog program cannot express negative information, and that there are no subsets of $\emptyset$. This kind of argument structures (those based on strict derivations) are quite particular. For instance, no other argument structure can defeat them, no matter the argument-comparison criterion considered (this follows from Prop. 3.1 in [6]). That is to say, its corresponding dialectical tree $\mathcal{T}_{\langle \emptyset, q \rangle}$ has only one node labelled $\langle \emptyset, q \rangle$ and marked **U**, which in turn means that q is a warranted literal of this $de.l.p$.

($\Leftarrow$)
If q is a warranted literal from P, there must exist an argument structure $\langle \mathcal{A}, q \rangle$ for q. According to Def. 2, $\mathcal{A} \subset \Delta$, whereas $\Delta = \emptyset$. Therefore, $\mathcal{A} = \emptyset$, which means that there must exist a strict derivation of q from P. Let $L_1, L_2, \ldots, L_n = q$ be the sequence of literals compossing that strict derivation. Once again, this same sequence can also be used to construct a SLD-derivation of q from P. Finally, since there exists a SLD-derivation of q from P, q must be entailed from P. $\square$

3.1.4 Negative Information

So far, only positive information was considered. Even though positive information is our preferred source of knowledge, sometimes negative information constitutes a valid source of additional knowledge. Both relational databases and Prolog adopt some sort of Closed-World Assumption (CWA) in order to represent this kind of information, but, by doing so, a subtle aspect is lost in the process: CWA does not distinguish between not knowing whether a certain relation holds over some objects, and knowing that this relation in fact does not hold over those objects. This issue has been addressed in Prolog giving birth to *extended logic programming* [7], a formalism that extends Prolog allowing the explicit representation of negative information. Following a simmilar approach, DeLP also allows the representation of this kind of knowledge.

Remark 4. *Let r be a relation and $\overline{X}$ a set of objects such that we have witnessed that $\overline{X} \notin r$. This knowledge can be expressed in a $de.l.p$ by adding a new fact $\sim r(\overline{X})$.*

3.1.5 Strong Conflicts

The conflict arising from complementary literals, called *explicit conflict* (for instance, "being alive" and "not being alive"), is a situation that DeLP's inference engine captures by itself. However, sometimes the conflict existing between a set of situations is more subtle (or less syntactic), and does not involve complementary concepts (for instance, "being alive" and "being dead"). In a sense, this conflict represents a set of situations that cannot happen all at the same time. Let us call this kind of conflict *strong*.

Remark 5. *Let $S_1, S_2, \ldots, S_n$ be all the situations characterizing a strong conflict. This knowledge can be expressed in a $de.l.p$ by adding n new strict rules of the form $\sim S_i \leftarrow S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$, for every S_i, $1 \leq i \leq n$.*

[1]the formal demonstration of this statement, by virtue of structural induction, is trivial.

Example 5. The following $de.l.p$ captures the strong conflict between the notions of being alive and being dead.

Π	Δ
`~dead(X) <- alive(X)`	
`~alive(X) <- dead(X)`	

3.2 Representing Non-Monotonic Knowledge

Nonmonotic reasoning is a recent phenomenon [14, 8, 9], hence the representation of non-monotonic knowledge is a topic still under exploration. Not withstanding, the comunity appear to be reaching an initial agreement on some of its fundamental tenets. In what follows, we consider in detail some classical aspects of the process of representing non-monotonic knowledge in DeLP.

3.2.1 Defaults

When it comes to make the case for non-monotic reasoning, most formal theories resort to argue that there is a feature of common-sense reasoning which cannot be captured using monotonic theories: *defaults*. A default establishes a connection between concepts weaker than class inclusion, yet relevant enough as to require to be made explicit. Put the other way around, defaults can be used to model *relations with exceptions*. The classical example in Artificial Intelligence is “birds usually fly”. Observe that this relation cannot be expressed as class inclusion ($e.g.$, “birds fly”), because it becomes inconsistent with the fact that some birds do not fly ($e.g.$, penguins, ostriches, kiwis, etc.), nor it can be expressed as a rule with explicit exceptions ($e.g.$, “birds that are not penguins, kiwis, $\ldots$ fly”), because all the exception to this relation must be known beforehand (an inconvenience known as the *qualification problem*). It is in this particular regard that DeLP really shines. Its representation of defaults is quite natural, and even more straightforward than the representantion of defaults in Default Logic or other nonmonotonic theories.

Remark 6. *Let A and B be two situations such that when A occurs, B usually also occurs (that is to say, if A, then by default B). This knowledge can be expressed in a $de.l.p$ simply by adding a new defeasible rule $B \prec A$.*

3.2.2 Weak Conflicts

Sometimes, explicit conflicts ($i.e.$, those arising from complementary situations), or strong conflicts ($i.e.$, those involving two or more situations that cannot happen at the same time), are not enough to capture all the conflicts worth modelling in a given scenario. For instance, there is a weaker form of conflict not covered so far, which we will call *weak conflict*. This kind of conflict reflects that two or more situations *usually* cannot occur at the same time, but *exceptionally,* they may. Even though neither explicit conflicts nor strong conflicts can capture this situation, a variation of the previous solution can be used to model weak conflicts.

Remark 7. *Let $S_1, S_2, \ldots, S_n$ be all the situations characterizing a weak conflict. This knowledge can be expressed in a $de.l.p$ by adding n new defeasible rules of the form $\sim S_i \prec S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n$, for every S_i, $1 \leq i \leq n$.*

Note that it is entirely possible to conceive a scenario were all these situations are met at the same time, considering the weak nature of this conflict, whereas strong conflicts make such a scenario outright impossible. It is the existence of this exceptional scenario what distinguishes weak from strong conflits.

Example 6. The following $de.l.p$ captures the weak conflict between a sunny day and the fact that it is raining.

Π	Δ
	`~sunny -< rainy`
	`~rainy -< sunny`

3.2.3 Assumptions

Some theories incorporate nonmonotonic features by allowing the reasoner to make tentative assumptions. This possibility alone gives rise to nonmonotonic reasoning, since an assumption compatible under certain state of the world may become unfeasible with the addition of new information. For example, Bondarenko $et\ al.$ [2] developed a formalisms built around abductive reasoning following this approach. Although the nonmonotic features in DeLP are the consequence of the dialectical analysis performed to determine if a conclusion is warranted, there is a particular kind of assumption, called *presumption*, which can also be modelled in this formalism. Bluntly put, a presumption is a *default fact*. This notion, first mentioned by Nute in his Defeasible Logic [11], allow the knowledge engineer to define a set of assumptions or suppositions which may be freely used as long as no other reasons are raised against them.

Remark 8. *Let A be an assumption worth being modeled. This knowledge can be captured in a $de.l.p$ by adding a new fact F, for a literal F not appearing in that $de.l.p$, and a new defeasible rule of the form $A \prec F$.*

Example 7. Consider the following $de.l.p$:

Π	Δ
`bird(X) <- chicken(X)`	`flies(X) -< bird(X)`
`~chicken(X) <- ~bird(X)`	`~flies(X) -< chicken(X)`
`chicken(coco)`	`flies(X) -< chicken(X), scared(X)`
`assumption`	`scared(coco) -< assumption`

According to this $de.l.p$, $flies(coco)$ is warranted (a conclusion reached under the assumption that $coco$ was scared). Observe how the addition of the fact $assumption$ and the corresponding defeasible rule captured the essence of the assumption that $coco$ was scared.

4 CONCLUSIONS

Defeasible Logic Programming constitutes an attractive framework for representing incomplete and potentially contradictory information, a cornerstone of many real-world problems. As such, the task of representing the knowledge required to reason upon these problems is not trivial. To that end, in this article we have developed a methodology for representing knowledge in DeLP, essentially a set of guidelines covering the major challenges knowledge engineers usually encounter during this phase. This methodology encompasses two distinctive forms of knowledge representation: first monotonic knowledge, and then non-monotonic knowledge. For each of them, we have shown how to represented some prototypical situations using DeLP's language.

As a future work, we plan to extend this methodology to cover other forms of monotonic and non-monotonic knowledge representation. For instance, we would like to further investigate the different ways of modelling priorities among defaults, and also take into account other variants of DeLP such as ODeLP [3] or DeLP$^{\emptyset}$ [4].

REFERENCES

[1] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[2] Andrei Bondarenko, Phan M. Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1–2):63–101, 1997.

[3] Marcela Capobianco, Carlos I. Chesñevar, and Guillermo R. Simari. Argumentation and the Dynamics of Warranted Beliefs in Changing Environments. *Journal of Autonomous Agents and Multiagent Systems*, 11:127–151, 2005.

[4] Telma Delladio, Nicolás D. Rotstein, and Guillermo R. Simari. A Comparison between Non-Monotonic Formalisms. In *Proceedings del 8vo Workshop de Investigadores en Ciencias de la Computación (WICC)*, pages 143–148, 2006.

[5] Alejandro J. García. *Programación en Lógica Rebatible: Lenguaje, Semántica Operacional, y Paralelismo.* PhD thesis, Departamento de Ciencias de la Computacin, Universidad Nacional del Sur, Bahía Blanca, Argentina, December 2000.

[6] Alejandro J. García and Guillermo R. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[7] Michael Gelfond and Vladimir Lifschitz. Logic Programs with Classical Negation. In David H. D. Warren and Perter Szeredi, editors, *Proceedings of the 7th International Conference on Logic Programming*, pages 579–597, 1990.

[8] John L. McCarthy. Circumscription—A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980.

[9] Drew V. McDermott and Jon Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13(1–2):41–72, 1980.

[10] Allen Newell and Herbert Simon. Computer Science as Empirical Inquiry. In *ACM Turing Award Lectures*, pages 287–387. Addison-Wesley, 1987.

[11] Donald Nute. Defeasible Reasoning. In *Proceedings of the XX Annual Hawaii International Conference on System Sciences*, pages 470–477, 1987.

[12] Roger Penrose. *The Emperor's New Mind.* Oxford University Press, 1989.

[13] David L. Poole. On the Comparison of Theories: Preferring the Most Specific Explanation. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 144–147, 1985.

[14] Raymond Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1–2):81–132, 1980.

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