Modelling Argumentation in a Logic-Programming Setting: Formalization and Logical Properties

Guillermo R. Simari – Carlos I. Chesñevar – Alejandro J. García
Laboratory of Research and Development in Artificial Intelligence (LIDIA)
Departament of Computer Science – Universidad Nacional del Sur – Bahía Blanca, ARGENTINA
Email: {grs,cic,aaj}@cs.uns.edu.ar – Tel/Fax: +54-291-455-5135/5136

Keywords: defeasible argumentation, logic programming, labelled deduction

1 Introduction and motivations

Defeasible argumentation [17, 5, 14] has proven to be a successful approach to finding a suitable formalization for reasoning with incomplete and potentially inconsistent information. Recent research (notably [1]) has shown that defeasible argumentation constitutes a confluence point for characterizing traditional approaches to non-monotonic reasoning (such as Gelfond’s extended logic programming and Reiter’s default logic). In this context, logic programming (LP) has provided a useful setting for exploring both theoretical and practical issues in defeasible argumentation. Dung’s work on argumentative semantics for logic programs [6] paved the way for other formalisms, most of them based on different versions of extended logic programming (such as Prakken’s [13], among others).

This paper reports some of the main results from two major research lines that have been explored in our Laboratory since 1994, continuing the research work started in 1987 [16]. First, we will describe the most relevant features of defeasible logic programming (DeLP) [8], a LP-based formalism for defeasible argumentation. Second, we will detail some aspects of LDSar [3], a formalization for defeasible argumentation based on labelled deduction (LD) [7] which allows to explore several logical properties of DeLP, as well as other related formalisms. Finally, we will present the main conclusions that have been obtained, as well as a brief sketch of our currently ongoing research.

2 How we perceive arguments

2.1 Knowledge representation and inference: argument, counterargument, defeat

A knowledge base (DeLP program) can be thought of as a pair \((\Pi, \Delta)\), where \(\Pi\) and \(\Delta\) stand for sets of non-defeasible and defeasible knowledge, respectively. The set \(\Pi\) is assumed to be non-contradictory.¹ This distinction is quite traditional in many existing argumentative frameworks (such as Prakken & Sartor’s [14] and Vreeswijk’s [18]), and it can be traced back to the original Simari-Loui formalism [17] The underlying logical language is that of extended logic programming, enriched with a special symbol “\(<\)” to denote defeasible rules. Both default and classical negation are allowed (denoted not and \(\not\), respectively). Syntactically, the symbol “\(<\)” is all what distinguishes a defeasible rule \(p \leftarrow q_1, \ldots, q_k\) from a strict (non-defeasible) rule \(p \leftarrow q_1, \ldots, q_k\). DeLP rules are thus Horn-like clauses to be thought of as inference rules rather than rules in the object language.

Our notion of argument is a variant of the original definition proposed in [17], properly suited to a logic programming setting. An argument \(A\) is a (possibly empty) set of ground defeasible rules that together with the set \(\Pi\) provide a logical proof for a given literal \(h\), satisfying besides the additional requirements of non-contradiction and minimality. Formally:

**Definition 2.1 (Argument).** Given a DeLP program \(P\), an argument \(A\) for a query \(q\), denoted \((A, q)\), is a subset of ground instances of the defeasible rules of \(P\), such that:

1. there exists a defeasible derivation for \(q\) from \(\Pi \cup A\),
2. \(\Pi \cup A\) is non-contradictory, and
3. \(A\) is minimal with respect to set inclusion.

¹ We use the term non-contradictory instead of inconsistent to emphasise independence of the underlying logical language. Contradiction stands for deriving two complementary literals wrt some kind of negation.
An argument \( \langle A_1, Q_1 \rangle \) is a sub-argument of another argument \( \langle A_2, Q_2 \rangle \) if \( A_1 \subseteq A_2 \).

In the case of DeLP, the notion of logical proof (defeasible derivation) is the usual query-driven derivation used in logic programming, performed by backward chaining on both strict and defeasible rules. Minimality imposes a kind of ‘Occam’s razor principle’ [17] on argument construction, as any superset of an argument \( A \) can be proven to be ‘weaker’ than \( A \) itself as far as possible attacking counterarguments are concerned. The non-contradiction requirement forbids the use of (instances of) defeasible rules in an argument \( A \) whenever \( \Pi \cup A \) entails two complementary literals \( p \) and \( \overline{p} \). It should be noted that non-contradiction captures the two usual approaches to negation in logic programming (viz. default negation and classic negation), both of which are present in DeLP and related to the notion of counterargument, as shown next.

**Definition 2.2 (Counterargument. Defeat).**

1. An argument \( \langle A_1, q_1 \rangle \) counterargues an argument \( \langle A_2, q_2 \rangle \) at a literal \( q \) iff there is a subargument \( \langle A, q \rangle \) of \( \langle A_2, q_2 \rangle \) such that the set \( \Pi \cup \{q_1, q\} \) is contradictory.
2. An argument \( \langle A_1, q_1 \rangle \) counterargues an argument \( \langle A_2, q_2 \rangle \) at a literal not \( q_1 \) if a literal not \( q_1 \) is present in the body of some defeasible rule in \( A \).

We will assume a partial order \( \subseteq \text{Args}(P) \times \text{Args}(P) \) on arguments of \( P \). In case 1, \( \langle A_1, q_1 \rangle \) will be called a proper defeater for \( \langle A_2, q_2 \rangle \) iff \( \langle A_1, q_1 \rangle \) is strictly preferred over \( \langle A, q \rangle \). In case \( \langle A_1, q_1 \rangle \) and \( \langle A, q \rangle \) are unrelated to each other, \( \langle A_1, q_1 \rangle \) will be called a blocking defeater for \( \langle A_2, q_2 \rangle \). In case 2, \( \langle A_1, q_1 \rangle \) will be considered a blocking defeater for \( \langle A_2, q_2 \rangle \).

Specificity [17] is typically used as a syntax-based preference criterion among conflicting arguments, although any other partial order would also be valid.

### 2.2 Computing warrant through dialectical analysis

The dialectical analysis is carried out by using a tree-like structure called the dialectical tree (first introduced in [15]) which renders easy the analysis and implementation of defeat and reinstatement among arguments, as well as the definition of “fallacy-checks” (such as detecting circular and/or contradictory argumentation during the dialectical process). A dialectical tree rooted in an argument \( \langle A_0, q_0 \rangle \) is defined by ‘bundling’ together all possible acceptable argumentation lines starting in \( \langle A_0, q_0 \rangle \), i.e., sequences of arguments and defeaters which are fallacy-free. A subsequent marking procedure allows to determine whether the original argument \( \langle A_0, q_0 \rangle \) is ultimately accepted or warranted.

An argumentation line \( \lambda(A_0, q_0) = \langle A_0, Q_0 \rangle, \langle A_1, Q_1 \rangle, \langle A_2, Q_2 \rangle, \ldots, \langle A_n, Q_n \rangle \ldots \rangle \) can be thought of as an exchange of arguments between two parties, a proponent and an opponent [15], such that each \( \langle A_i, Q_i \rangle \) defeats the previous argument \( \langle A_{i-1}, Q_{i-1} \rangle \) in the sequence. Dialectics imposes additional requirements on such an argument exchange to be considered rationanly acceptable. In such a setting, fallacious reasoning (such as circular argumentation and falling into self-contradiction) is to be avoided. This can be done by requiring that all argumentation lines be acceptable [15]. An acceptable argumentation line starting with an argument \( \langle A_0, Q_0 \rangle \) constitutes an exchange of arguments which can be pursued until no more arguments can be introduced because of the dialectical constraints above discussed.

**Definition 2.3 (Acceptable argumentation line).** Let \( P \) be a dlp, and let \( \lambda = \langle A_0, Q_0 \rangle, \langle A_1, Q_1 \rangle, \ldots, \langle A_n, Q_n \rangle, \ldots \rangle \) be an argumentation line in \( P \). Let \( X = \langle [A_0, Q_0], \langle A_1, Q_1 \rangle, \ldots, \langle A_k, Q_k \rangle \rangle \) be an initial segment of \( \lambda \) starting in \( \langle A_0, Q_0 \rangle \). The sequence \( X \) is an acceptable argumentation line in \( P \) iff it is the longest subsequence in \( \lambda \) satisfying the following conditions:

1. The sets \( X_S \) and \( X_I \) are each non-contradictory sets of arguments wrt \( P \).
2. No argument \( \langle A_j, Q_j \rangle \) in \( X \) is a sub-argument of an earlier argument \( \langle A_i, Q_i \rangle \) of \( X \) \((i < j)\).
3. There is no subsequence of arguments \( \langle [A_i-1, Q_{i-1}], \langle A_i, Q_i \rangle, \langle A_{i+1}, Q_{i+1} \rangle \rangle \) in \( X \), such that \( \langle A_i, Q_i \rangle \), is a blocking defeater for \( \langle A_{i-1}, Q_{i-1} \rangle \) and \( \langle A_{i+1}, Q_{i+1} \rangle \) is a blocking defeater for \( \langle A_i, Q_i \rangle \).

---

2 The first notion of attack is borrowed from SL framework [17]; the second one is related to Dung’s argumentative approach to logic programming [6] as well as to other formalisations, such as [13, 10].

3 Non-contradiction for a set of arguments is defined as follows: a set \( S = \bigcup_{i=1}^{n} \{\langle A_i, Q_i \rangle\} \) is contradictory wrt a DeLP program \( P \) iff \( \Pi \cup \bigcup_{i=1}^{n} A_i \) is contradictory.
The rationales for the conditions in Definition 2.3 are to be understood in a dialectical setting [15]. Condition 1 dismisses the use of contradictory information on either side (proponent or opponent). Condition 2 eliminates the “circulus in demonstrando” fallacy (circular reasoning). Finally, condition 3 enforces the use of a stronger argument to defeat an argument which acts as a blocking defeater.

Definition 2.4 (Dialectical Tree). Let $\mathcal{P}$ be a dlp, and let $A_0$ be an argument for $Q_0$ in $\mathcal{P}$. A dialectical tree for $(A_0, Q_0)$, denoted $T_{(A_0, Q_0)}$, is a tree structure defined as follows:

1. The root node of $T_{(A_0, Q_0)}$ is $(A_0, Q_0)$.
2. $(B', H')$ is an immediate children of $(B, H)$ iff there exists an acceptable argumentation line $\lambda^{(A_0, Q_0)}_1 = [(A_0, Q_0), (A_1, Q_1), \ldots, (A_n, Q_n)]$ such that there are two elements $(A_i, Q_i) = (B', H')$ and $(A_{i+1}, Q_{i+1}) = (B, H)$, for some $i = 0, \ldots, n - 1$.

Note that a dialectical tree is an and-or tree. Leaves can be marked as undefeated nodes (U-nodes), as they have no defeaters. Every inner node is to be marked as D-node if it has at least one active defeater (U-node) as a child, and as U-node otherwise. An argument $(A_0, Q_0)$ is warranted iff the root of its associated dialectical tree $T_{(A_0, Q_0)}$ is labeled as U-node. Different doxastic attitudes are distinguished when answering a given query $q$, according to the status of warrant.

1. Believe $q$ (resp. $\sim q$) when it is supported by a warranted argument.
2. Believe $q$ is undecided whenever neither $q$ nor $\sim q$ are supported by warranted arguments.
3. Believe $q$ is unknown whenever $q$ does not belong to the signature of the underlying logical language.

3 Formalizing argumentation using labelled deduction

3.1 Motivations and fundamentals

The study of logical properties of defeasible argumentation, particularly those related to the DeLP framework, motivated the development of LDSar [3], an argumentation formalism based on the labelled deduction methodology [7]. In labelled deduction, the usual notion of formula is replaced by the notion of labelled formula, expressed as Label: $f$ where Label represents a label associated with the wff $f$. A labelling language $L_{Label}$ and a knowledge-representation language $L_{kr}$ can be combined to provide a new, labelled language, in which labels convey additional information also encoded at object-language level. Formulas are labelled according to a family of deduction rules, and with agreed ways of propagating labels via the application of these rules.

In LDSar, the language $L_{kr}$ is the one of extended logic programming. Labels extend this language by distinguishing defeasible and non-defeasible information. A consequence relation $\vdash_{arg}$ propagates labels, implementing the SLD resolution procedure along with a consistency check every time new defeasible information is introduced in a proof. This information is collected into a set of support, containing all defeasible information needed to conclude a given formula. Thus, arguments are modelled as labelled formulas $A_{kr}$, where $A$ stands for a set of (ground) clauses, and $h$ for an extended literal.

Given a knowledge base $\Gamma$ the consequence relation $\vdash_{arg}$ allows to infer labelled formulas of the form $arg::literal$. Since arguments may be in conflict, a new, extended consequence relationship $\vdash_{\Gamma}$ will be defined. Those wffs derivable from $\Gamma$ via $\vdash_{\Gamma}$ will correspond to dialectical trees. These new labelled wffs will therefore have the form dialectical tree: conclusion.

3.2 Some logical properties. Equivalence results

LDSar provides a useful formal framework for studying logical properties of defeasible argumentation in general, and of DeLP in particular. Equivalence results with other argumentative frameworks were also studied, particularly those relating DeLP with other LP-based formalisms.

Cumulativity was proven to hold for argumentative formulae. This allows to think of a DeLP program as a knowledge base containing ‘atomic’ arguments (facts and rules), which can be later on extended by incorporating new, more complex arguments. This feature makes easier to formalize dialectical databases, a TMS-based approach to defeasible argumentation which is currently being explored [2]. Cumulativity is proven not to hold for warranted conclusions, following the intuitions suggested by Prakken & Vreeswijk [14].

Superclassicality was shown to hold for both argument construction and warrant wrt SLD resolution. In other words, if $Th_{std}(\Gamma)$ denotes the set of conclusions that can be obtained from $\Gamma$ via SLD, then it holds...
that $C_{arg}(\Gamma) \subseteq Th_{sld}(\Gamma)$ and $C_{war}(\Gamma) \subseteq Th_{sld}(\Gamma)$, where $C_{arg}$ and $C_{war}$ stand for the consequence operator for argument construction and warrant, respectively. This implies, among other things, that the analysis of attack between arguments can be focused on literals in defeasible rules. Analogously, right weakening is proven to hold for both $C_{arg}$ and $C_{war}$. This implies that (warranted) arguments with a conclusion $x$ account also as (warranted) arguments for $y$ whenever $y \leftarrow x$ is present as a strict rule.

Another interesting issue concerns the definition of variants for $LDS_{ar}$. Since $LDS_{ar}$ is a logical framework, its knowledge-encoding capabilities are determined by the underlying logical language, whereas the inference power is characterized by its natural deduction rules. Adopting a different KR language or modifying the existing inference rules will lead to different variants of $LDS_{ar}$. Thus, for instance, adopting a full first-order language will lead to a logical system with a behavior similar to the SL framework [17]. On the other hand, restricting the KR language to Horn clauses will result in a formulation closer to normal logic programming (NLP) under well-founded semantics.

Figure 1 summarizes some of these variants, and shows how they can be related to some existing argumentation frameworks, such as Simari-Loui’s [17], MTDR [15], DeLP [3] and NLP (normal logic programming), conceptualized in an argumentative setting as suggested in [1]. Two distinguished variants of DeLP deserved particular attention, namely $DeLP_{not}$ and $DeLP_{neg}$ ($DeLP$ restricted to default and strict negation, resp.). In [4], the relation between these variants of DeLP and normal logic programming was explored. Different criteria under which both strict and defeasible rules could be rewritten into a simpler but semantically equivalent form were defined.

The notion of dialectical tree and acceptable argumentation lines proved to be very useful for capturing different aspects of the process of argumentation. It should be remarked that similar approaches have been recently tried in other formalisms (see for example [12]). A formal analysis proved that dialectical trees can be pruned (following the procedure introduced in [15]) without affecting the marking procedure. An equivalence theorem between top-down and bottom-up computation of dialectical trees was also established.

4 Conclusions. Ongoing work

As we have shown in this paper, DeLP provides a LP-based setting for defeasible argumentation that combines the well-known advantages of the logic programing paradigm (such as declarativity and implementability) together with the dialectical considerations required to model the process of defeasible argumentation. The DeLP language incorporates the natural benefits of allowing both classical and default negation, distinguishing at the same time between defeasible and non-defeasible information by introducing defeasible and strict rules. Specificity was adopted as the argument-comparison criterion, but it can be replaced by any other partial order among arguments without changing the rest of framework. Implementation issues deserved special consideration. In order to gain efficiency, the language was implemented using an abstract machine defined and implemented as an extension of the Warren Abstract Machine. Recent work showed how to extend DeLP capabilities into a multiagent environment [cite paper Alejandro&Tareau].

During the last decade, a ‘clash of intuitions’ has appeared within the argumentation community [5, 14], where different, alternative approaches have been intended. As we have briefly sketched in the second part of this paper, having a logical formalism such as $LDS_{ar}$ makes it easier to analyze, compare and relate

\[ A \text{ A}\]

\[ A \text{ A}\]

\[ A \text{ A}\]

Fig. 1. A taxonomy relating the expressive power of $LDS_{ar}$ and different argumentation systems

4 A full discussion of different argumentative frameworks encompassed by $LDS_{ar}$ can be found in [3].
different features associated with existing argumentative frameworks, providing at the same time a test-bed for studying other related issues (such as argumentation protocols, resource-bounded reasoning, etc.). These aspects are directly related to formalizing multiagent environments, in which argumentation plays a major role when modelling the communicative and reasoning abilities of the agents involved [11]. Research in this direction is currently being pursued.

References