An Alternative Foundation for DeLP: Defeating Relations and Truth Values

Ignacio D. Viglizzo[1,2,5], Fernando A. Tohmé[1,3,5], and Guillermo R. Simari[1,4]

[1] Artificial Intelligence Research and Development Laboratory
[2] Department of Mathematics
viglizzo@criba.edu.ar
[3] Department of Economics
ftohme@criba.edu.ar
[4] Department of Computer Science and Engineering
Universidad Nacional del Sur – Av. Alem 1253 - B8000CPB Bahía Blanca - Argentina
grs@cs.uns.edu.ar
[5] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

Abstract. In this paper we recast the formalism of argumentation formalism known as DeLP (Defeasible Logic Programming) in game-theoretic terms. By considering a game between a *Proponent* and an *Opponent*, in which they present arguments for and against each literal we obtain a bigger gamut of truth values for those literals and their negations as they are defended and attacked. An important role in the determination of warranted literals is assigned to a *defeating* relation among arguments. We consider first an unrestricted version in which these games may be infinite and then we analyze the underlying assumptions commonly used to make them finite. Under these restrictions the games are always *determined* -one of the players has a winning strategy. We show how varying the defeating relation may alter the set of truth values reachable under this formalism. We also show how alternative characterizations of the defeating relation may lead to different assignations of truth values to the literals in a DeLP program.

1 Introduction and Motivation

The development of *defeasible reasoning* in the last decades [Pol87, SL92, Nut94, Pol95, CML00, PV02], provided the foundations of an alternative form of declarative programming, Defeasible Logic Programming (DeLP) [GS04]. This formalism blends Logic Programming with Defeasible Argumentation, allowing the representation of tentative knowledge and leaving for the inference mechanism the task of finding the conclusions that the knowledge base warrants [CDSS03].

DeLP inherits from Logic Programming (LP) the formal characterization of *programs* as sets of rules. The difference is that in DeLP two kinds of rules are considered. On one hand, *strict rules*, which are assumed to represent sound knowledge and are handled as the rules in LP. On the other hand, *defeasible rules* represent tentative knowledge that may be defeated by other information.

S. Hartmann and G. Kern-Isberner (Eds.): FoIKS 2008, LNCS 4932, pp. 42–57, 2008.

Again as in LP, DeLP operates by resolving the status of pieces of knowledge that we may deem as queries. A query l succeeds if there exists a warranted argument for l. Arguments are constructed using both types of rules and facts (which can be seen as special cases of strict rules).

The core of DeLP resides in the characterization of the *warrant procedure*. Defeasible Argumentation has provided a solid foundation over which the standard formalization of this procedure has been constructed. A key element in the warrant procedure is the set of criteria according to which two contradicting arguments are compared and eventually one of them deemed as *defeating* the other. While pure syntactic criteria like *specificity* constitute the main choice in the design of the warrant procedure [Poo85, SL92, SGCS03], the notion of warrant can be abstracted away from the details of the order relation among arguments.

The inference mechanism generates all the arguments that either support or contradict l. Then, a *warrant procedure* is applied determining which arguments end up undefeated. If there exists at least one argument warranting l, it yields a positive answer.

Various papers have been published by the LIDIA group in Argentina, developing the theory, providing prototype implementations and exploring a variety of applications. Some other groups have also been publishing papers that extend the theory and use it for applications. But while this framework has well-established and interesting features, we find that many aspects of the inference mechanism of DeLP as well as its corresponding semantics, have a strong game-theoretic flavor. This is inherited from the dialogical schema on which DeLP bases the warrant of queries [GS04].

As it is well known, one of the salient ways in which rational agents try to establish the appropriateness of claims is by means of discussions. The idea is that more appropriate contentions should be supported by the more cogent *arguments* in a given discussion. When discussions are seen as foundations for logic and reasoning systems, the goal is not to determine *who* wins the discussion, but instead which conclusions are supported by the better arguments. Furthermore, for the purpose of determining the relevant conclusions, it does not matter either whether a discussion has a non-cooperative or cooperative nature. The former case is known as a *debate*, in which some agents win and others lose. In a cooperative discussion, instead, agents collaborate in order to find the best answer to some question. But even if the goals of the discussants are non-conflicting, arguments for and against potential answers have to be weighted up to find out which one is more appropriate. It seems natural then, to resort to the idea of a game in which two fictitious players make their moves, that is, they select arguments to be uttered in a discussion. The winning strategies end up leading to a win by either one of the two players.

Although game-theoretic approaches to logic systems are well-known [Hin73, LL78, HS97, vB02], our goals here are somewhat different. We intend to provide an alternative foundation for DeLP closer to the developments in dialogue systems ([Ham70], [Ham71]), particularly in AI ([Bre01], [Lou98]) with

applications in law ([LAP00], [MP02], [PWA03]). Even closer to our goals are developments like the explicit or implicit use of game-theoretic notions for *argumentation* [BH01, AC02, Dun95, Pra05].

The whole idea in that literature is that there exists a relation of defeat among arguments (which are basically derivations of claims). This relation allows to determine the arguments that are either undefeated or with defeated defeaters. The claims supported by these arguments are said warranted. The shape of the class of either these claims or of their supporting arguments has been assessed in many settings. But it has to be said that the use of the results of game-theoretic results (instead of just its terminology) in those analysis has been rather scarce.

What we intend to do here is to introduce an alternative, game-theoretic, presentation of DeLP. In particular, we use the notion of *winning strategy* as a central component of the inference mechanism, that yields answers to queries, and consequently for the determination of their truth values. We find that this alternative does preserve DeLP central features, in particular the class of warranted claims, and this is why we claim that the formalism presented here is an alternative foundation, instead of an entirely new formalism. On the other hand, our use of game-theoretic notions can be easily extended to the whole gamut of argumentation formalisms, yielding an alternative semantics to which strategic concepts can be applied.

As said, this paper takes some basic ideas from game theory and uses them for an alternative definition for DeLP. The approach is more abstract. It is interesting because it clarifies some of the notions in DeLP, and it suggests some alternative definitions for aspects of DeLP.

Perhaps more importantly the paper suggests a more general framework for viewing a variety of other argumentation systems in a common game-theoretic way. If so, this paper could be the first step of a very important line of research for understanding the foundations of argumentation. The goal of this paper is to provide an alternative specification of DeLP. It allows to express the main ideas in the formalism in terms of the sets of literals that are warranted. It allows a game-theoretic representation of the warrant procedure and yields a graded truth valuation of literals. We will show how this depends on the properties of the *defeating* relation among arguments.

2 The Basics of DeLP

In order to discuss the set-theoretical properties of DeLP, we have to present the basics of this formalism.[1]

Each *Defeasible Logic Program* $\mathbb{P}$ is a finite set of facts, strict rules, and defeasible rules $\mathbb{P} = \langle \Pi, \Delta \rangle$, where Π denotes the set of facts and strict rules, while Δ denotes the set of defeasible rules. The set Π is the disjoint union of the sets Π_F of facts and Π_R of strict rules.

Facts and rules are defined in terms of atoms. More precisely, let At be the set of atoms that occur in a given program $\mathbb{P}$. Given a set $X \subseteq \mathsf{At}$ of atoms, $\sim X$

[1] We follow very closely the presentation in [GS04].

is the set $\{\sim x : x \in X\}$. Then, the set Lit is the set of all literals in $\mathsf{At} \cup \sim \mathsf{At}$. The complement $\bar{l}$ of a literal $l \in$ Lit is $\sim x$ if l is an atom x and x if l is a negated atom $\sim x$. This indicates the strong syntactic bent of DeLP and makes the formalism purely propositional.

Then, the main components of a program $\mathbb{P}$ are:

Π_F: *Facts*, which are ground literals in Lit.
Π_R: *Strict Rules* of the form $l_0 \leftarrow l_1, \ldots, l_n$, where l_0 is the *head* and $\{l_i\}_{i>0}$ is the *body*. Each l_i in the body or the head is in Lit.
Δ: *Defeasible Rules* of the form $l_0 \prec\!\!- l_1, \ldots, l_n$, where l_0 is the *head* and $\{l_i\}_{i>0}$ is the *body*. Again, each l_i in the body or the head is a literal.

Example 1. Let's consider a simple program $\mathbb{P}_1$ with $\Pi_F = \{b, c\}$; $\Pi_R = \{d \leftarrow a\}$ and $\Delta = \{a \prec\!\!- b, \sim a \prec\!\!- c\}$.

Rules, both strict and defeasible act on facts, allowing to derive literals. More precisely, a *defeasible derivation* of l up from $X \subseteq$ Lit, $\mathcal{R} \subseteq \Pi_R$ and $\mathcal{A} \subseteq \Delta$ is a finite sequence $l_1, \ldots, l_n = l$ of literals in Lit such that each l_i is either in X or there exists a rule in $\mathcal{R}$ with l_i as its head, and every literal b_j in its body is such that $b_j \in \{l_k\}_{k<i}$.

Then:

Definition 1. *Given sets $X \subseteq$ Lit, $\mathcal{R} \subseteq \Pi_R$ and $\mathcal{A} \subseteq \Delta$, $C(X, \mathcal{R}, \mathcal{A})$ is the set of all literals defeasibly derivable from $X \cup \mathcal{R} \cup \mathcal{A}$. The set of* strict consequences *of $X \subseteq$ Lit is $C_s(X) = C(X, \Pi_R, \emptyset)$. Finally, we are going to use often $C(\mathcal{A}) = C(\Pi_F, \Pi_R, \mathcal{A})$ for sets $\mathcal{A} \subseteq \Delta$.*

Definition 2. *Given a set $X \subseteq$ Lit, let $X^+ = X \cap \mathsf{At}$ and $X^- = \{a \in \mathsf{At} : \sim a \in X\}$. X is said to be* contradictory *if $X^+ \cap X^- \neq \emptyset$. A set of defeasible rules $\mathcal{A}$ is* contradictory *if $C(\mathcal{A})$ is contradictory.*

Example 2. For $\mathbb{P}_1$ of example 1, we have $C(\{b\}, \emptyset, \{a \prec\!\!- b\}) = \{a, b\}$; $C_s(\{a\}) = \{a, d\}$; $C_s(\Pi_F) = \Pi_F$ and $C(\Delta) = \{a, b, c, d, \sim a\}$, a contradictory set.

We assume that for all programs, the set Π of facts and strict rules is not contradictory. A fundamental relation in DeLP is that of *disagreement* between literals:

Definition 3. *Two literals $h, q \in$ Lit are said to* disagree *(for a given program $\mathbb{P}$) if $C_s(\Pi_F \cup \{h, q\}) = C(\Pi_F \cup \{h, q\}, \Pi_R, \emptyset)$ is contradictory. We define a binary relation $D \subseteq$ Lit $\times$ Lit to record which pairs of literals disagree. This relation is clearly symmetric.*

This relation matters for the comparison of arguments. In order to get to that, let us define the fundamental concept of argument. We will use $\mathcal{P}$ to denote the powerset construction.

Definition 4. *An* argument *is a pair* $\langle \mathcal{A}, h\rangle \in \mathcal{P}(\Delta) \times \text{Lit}$ *that satisfies:*

1. $h \in C(\mathcal{A})$
2. $\mathcal{A}$ *is not contradictory.*
3. *If* $h \in C(\mathcal{A}')$*, then* $\mathcal{A}' \not\subset \mathcal{A}$*, that is,* $\mathcal{A}'$ *is not a proper subset of* $\mathcal{A}$*.*

Definition 5. *Let* $\text{Arg}(\mathbb{P})$ *be the set of all arguments of a program* $\mathbb{P}$*. The argument* $\langle \mathcal{A}_1, h\rangle$ *is a subargument of* $\langle \mathcal{A}_2, h'\rangle$ *iff* $\mathcal{A}_1 \subseteq \mathcal{A}_2$*. We denote this with* $\langle \mathcal{A}_1, h\rangle \preceq \langle \mathcal{A}_2, h'\rangle$*.*

Example 3. With the program $\mathbb{P}_1$ of Example 1, we let $\mathcal{A}_1 = \{a \multimap b\}$ and $\mathcal{A}_2 = \{\sim a \multimap c\}$. Then the arguments are:

$$\text{Arg}(\mathbb{P}_1) = \{\langle \emptyset, b\rangle, \langle \emptyset, c\rangle, \langle \mathcal{A}_1, a\rangle, \langle \mathcal{A}_1, d\rangle, \langle \mathcal{A}_2, \sim a\rangle\}.$$

The *disagreeing relation* D is formed by the set $\{(a, \sim a), (d, \sim a)\}$ together with its symmetric pairs.

Note that the subargument relation $\preceq$ is a preorder over $\text{Arg}(\mathbb{P})$. Furthermore, if $\langle \mathcal{A}_1, h\rangle \preceq \langle \mathcal{A}_2, h'\rangle$ and $\langle \mathcal{A}_2, h'\rangle \preceq \langle \mathcal{A}_1, h\rangle$, then $\mathcal{A}_1 = \mathcal{A}_2$. This means that if we identify arguments with the same first component, $\preceq$ is simply a restriction of the inclusion relation over $\mathcal{P}(\Delta)$.

Definition 6. *For each literal* h *we define the binary relation* R_h *on* $\text{Arg}(\mathbb{P})$ *by* $\langle \mathcal{A}_1, h_1\rangle R_h \langle \mathcal{A}_2, h_2\rangle$ *iff there exists* $\langle \mathcal{A}, h\rangle \in \text{Arg}(\mathbb{P})$ *such that* $\mathcal{A} \subseteq \mathcal{A}_1$ *and* $(h, h_2) \in D$*. We say in this case that the argument* $\langle \mathcal{A}_1, h_1\rangle$ *is attacked by* $\langle \mathcal{A}_2, h_2\rangle$ *at* h *or that* $\langle \mathcal{A}_2, h_2\rangle$ attacks *or* rebutts $\langle \mathcal{A}_1, h_1\rangle$ *at* h*. We also say that* $\langle \mathcal{A}_2, h_2\rangle$ *is a* counter-argument *of* $\langle \mathcal{A}_1, h_1\rangle$*.*

Example 4. Following again the analysis of $\mathbb{P}_1$, we have that, for example, $\langle \mathcal{A}_1, a\rangle R_a \langle \mathcal{A}_2, \sim a\rangle$ and also $\langle \mathcal{A}_1, a\rangle R_d \langle \mathcal{A}_2, \sim a\rangle$. On the other hand, we have that $\langle \mathcal{A}_2, \sim a\rangle R_{\sim a} \langle \mathcal{A}_1, a\rangle$ and $\langle \mathcal{A}_2, \sim a\rangle R_{\sim a} \langle \mathcal{A}_1, d\rangle$.

We have the following easy consequences of the definition:

Proposition 7. *1. If* $\langle \mathcal{A}_1, h_1\rangle R_h \langle \mathcal{A}_2, h_2\rangle$*, then for some* $\mathcal{A} \subseteq \mathcal{A}_1$*,*$\langle \mathcal{A}, h\rangle$ *is an argument and* $\langle \mathcal{A}_2, h_2\rangle R_{h_2} \langle \mathcal{A}, h\rangle$*.*
2. If $\langle \mathcal{A}, l\rangle R_l \langle \mathcal{B}, p\rangle$*, then* $\langle \mathcal{B}, p\rangle R_p \langle \mathcal{A}, l\rangle$*.*
3. If $q \in C_s(\Pi_F)$*, then* $\langle \emptyset, q\rangle$ *is an argument and it has no counter-arguments.*
4. Furthermore, an argument $\langle \emptyset, q\rangle$ *cannot be a counterargument of any argument.*

Proof. 1. From the definition of R_h, we know that for some $\langle \mathcal{A}, h\rangle \in \text{Arg}(\mathbb{P})$ we have that $\mathcal{A} \subseteq \mathcal{A}_1$ and $(h, h_2) \in D$. Therefore, there exists $\mathcal{A}_2 \subseteq \mathcal{A}_2$ and $(h, h_2) \in D$ so we can claim that $\langle \mathcal{A}_2, h_2\rangle R_h \langle \mathcal{A}, h\rangle$.

2. Using the definition of R_l, this simply means that l and p disagree, and this is enough to justify that $\langle \mathcal{B}, p\rangle R_p \langle \mathcal{A}, l\rangle$, since the arguments are attacked precisely at the literals they support, so no subarguments need be considered. Recall also the minimality of the sets that form the argument.

3. It is easy to check that since $q \in C(\emptyset)$ while $\emptyset$ is not contradictory and has no proper subset, so $\langle \emptyset, q\rangle$ is an argument. Now assume it has a counterargument $\langle \mathcal{A}_2, h'\rangle$ that attacks $\langle \emptyset, q\rangle$ at h. Then there exists an argument $\langle \mathcal{A}, h\rangle$ with $\mathcal{A} \subseteq \emptyset$, so $h \in C(\emptyset)$, and $(h, h') \in D$. This means that $C_s(\Pi_F \cup \{h, h'\})$ is contradictory, but since $C(\emptyset) \subseteq C(\mathcal{A}_2)$, $\{h, h'\} \subseteq C(\mathcal{A}_2)$, so $\mathcal{A}_2$ is contradictory, and therefore $\langle \mathcal{A}_2, h'\rangle$ cannot be an argument.
4. If $\langle \mathcal{A}, h\rangle$ is an argument and $\langle \mathcal{A}, h\rangle R_p \langle \emptyset, q\rangle$, then there is an argument $\langle \mathcal{A}', p\rangle$ with $(p, q) \in D$ but we have seen in the previous proof that this contradicts the fact that $\mathcal{A}'$ is not contradictory.

Example 5. Consider the program $\mathbb{P}_2$ with $\Pi_F = \{c\}$ and $\Delta = \{a \prec b, b \prec c, \sim b \prec c\}$. Letting $\mathcal{A}_1$ be $\{a \prec b, b \prec c\}$ and $\mathcal{A}_2 = \{\sim b \prec c\}$ we have that $\langle \mathcal{A}_1, a\rangle R_b \langle \mathcal{A}_2, \sim b\rangle$. This follows from the fact that $\langle \mathcal{A}_2, \sim b\rangle$ attacks the subargument $\langle \{b \prec c\}, b\rangle$ of $\langle \mathcal{A}_1, a\rangle$. In symbols, this is $\langle \{b \prec c\}, b\rangle R_b \langle \mathcal{A}_2, \sim b\rangle$. Applying the previous proposition, part 2, we get $\langle \mathcal{A}_2, \sim b\rangle R_{\sim b} \langle \{b \prec c\}, b\rangle$.

Now we have the set $\mathrm{Arg}(\mathbb{P})$ with the relations R_h over it. We want to have a method to decide, given a literal l, whether it's supported by the program $\mathbb{P}$ or not. Clearly we want all literals in $C(\emptyset)$ to be supported or warranted. Which other literals should be supported? If there is a defeasible derivation of a literal l while $\bar{l}$ is not derivable, we want l to be warranted as well.

But what about the cases in which the derivation of l yields a contradictory set of literals, or there are arguments that support the negation of l as well?

We have seen that if the relation R_h holds between two arguments, there is also some attack on the attacking argument (Lemma 7, 1). We need to be able to tell which of these two arguments (if any) 'wins' the discussion.

There are several ways to do this. Our choice is to assume a binary relation $\leq$ contained in $R = \bigcup_{h \in \mathrm{Lit}} R_h$ (it will be of no consequence for us if $\leq$ holds in other cases, so we may as well just concentrate on subsets of R). The notation for this relation is somewhat misleading since we do not assume for the moment that $\leq$ is a partial order or even a preorder. It will just be an arbitrary way of deciding which one of two arguments, if any, is stronger than the other. This information will be used to construct a game as described in the following section.

We call $\leq$ the *defeating* relation. We define proper and blocking defeaters as follows:

Definition 8. $\langle \mathcal{A}_1, h_1\rangle$ *is a* proper defeater *of* $\langle \mathcal{A}_2, h_2\rangle$ *iff* $\langle \mathcal{A}_2, h_2\rangle \leq \langle \mathcal{A}_1, h_1\rangle$ *and it is not the case that* $\langle \mathcal{A}_1, h_1\rangle \leq \langle \mathcal{A}_2, h_2\rangle$.

$\langle \mathcal{A}_1, h_1\rangle$ *is a* blocking defeater *of* $\langle \mathcal{A}_2, h_2\rangle$ *iff* $\langle \mathcal{A}_2, h_2\rangle \leq \langle \mathcal{A}_1, h_1\rangle$ *and* $\langle \mathcal{A}_1, h_1\rangle \leq \langle \mathcal{A}_2, h_2\rangle$. *We denote this by* $\langle \mathcal{A}_1, h_1\rangle \approx \langle \mathcal{A}_2, h_2\rangle$.

Example 6. The defeating relation can be chosen arbitrarily as a subset of R, but this alone provides certain constraints. Continuing our ongoing example of $\mathbb{P}_1$ and its arguments, we observe that no defeating relation can compare the arguments $\langle \mathcal{A}_1, a\rangle$ and $\langle \mathcal{A}_1, d\rangle$. So if we have a defeating relation $\leq_1$ such that $\langle \mathcal{A}_1, d\rangle \leq_1 \langle \mathcal{A}_2, \sim a\rangle \leq_1 \langle \mathcal{A}_1, a\rangle$, we cannot expect it to be transitive. No defeating relation for this program could be a total order, either.

We can also have $\langle \mathcal{A}_1, a\rangle \leq_2 \langle \mathcal{A}_2, \sim a\rangle$, $\langle \mathcal{A}_2, \sim a\rangle \leq_2 \langle \mathcal{A}_1, a\rangle$ and $\langle \mathcal{A}_2, \sim a\rangle \leq_2 \langle \mathcal{A}_1, d\rangle$ so under this relation, $\langle \mathcal{A}_2, \sim a\rangle$ has $\langle \mathcal{A}_1, a\rangle$ as a blocking defeater and $\langle \mathcal{A}_1, d\rangle$ as a proper one.

Note that in [GS04], proper and blocking defeaters are defined in terms of an underlying comparison criterion $\prec$. We can always take as defeating relation the one stemming out of this prior relation, and thus subsume the previous work in our framework. Our definition, however, allows for the case in which two arguments attacking each other are not related under the defeating relation. In [GS04], these two would be blocking defeaters of each other, even if they were unrelated under the comparison criterion.

3 Warrant Games

Given a literal l we want to find out what the program $\mathbb{P}$ has to say about it. What if there are more than one argument supporting l? What if one of them is undefeated and the other is defeated? We will answer these questions through a slightly generalized version of the mechanism of warrant of DeLP [GS04], using the language of game theory, which turns out to be quite natural for framing the dialectical process that leads to the warrant of a literal.

First, we will introduce the definition of extensive games with perfect information. Then we will define *warrant games*, a class of games tailored to our needs, and finally we will show how to use them for answering queries to the program.

Definition 9. *(following [OR94]) An* extensive game with perfect information $G = \langle N, H, P, (U_i)_{i\in N}\rangle$ *consists of:*

- *A set N, the set of* players.
- *A set H of sequences (finite or infinite) that satisfies the following three properties:*
  - *The empty sequence $\emptyset$ is in H.*
  - *If $(a_k)_{k=1,\ldots,K} \in H$ (where K may be infinite) and $L < K$ then $(a_k)_{k=1,\ldots,L} \in H$.*
  - *If an infinite sequence $(a_k)_{k=1}^{\infty}$ satisfies $(a_k)_{k=1,\ldots,L} \in H$ for every positive integer L, then $(a_k)_{k=1}^{\infty} \in H$.*

  The members of H are called histories. *Each component a_k of a history is an* action *taken by a player. A history $(a_k)_{k=1,\ldots,K} \in H$ is* terminal *if it is infinite or there is no a_{K+1} such that $(a_k)_{k=1,\ldots,K+1} \in H$. The set of terminal histories is denoted with Z.*
- *A function $P : H \setminus Z \to N$, that indicates for each history in H which one of the players takes an action after the history.*
- *Functions $U_i : Z \to \mathbb{R}$ for $i \in N$ that give for each terminal history and each player, the* payoff *of that player after that history.*

The set H can be seen as a tree with root $\emptyset$, with its nodes labeled by the function P, and the leaves labeled by the functions U_n. We identify the elements a^k with edges of the tree. Therefore, each particular branch from the root is a

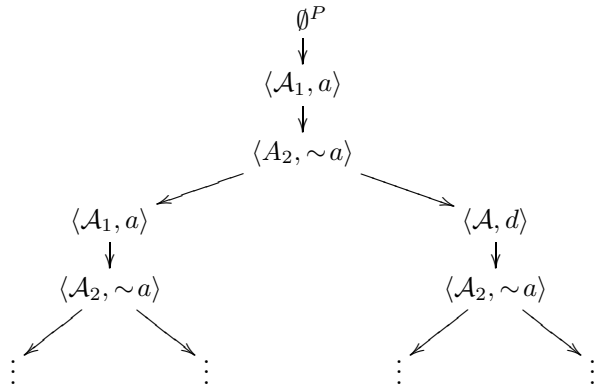
history, in which the edges are the consequent actions chosen by the players. We call the game *finite* if H is finite. After any nonterminal history h player $P(h)$ chooses an action from the set $A(h) = \{a : (h, a) \in H\}$.

A *warrant game* for a literal l is an extensive game with perfect information with two players. We will call these two players *Proponent* and *Opponent*. We define the game as follows:

- $P(\emptyset) = Proponent$.
- The actions that the proponent can take at the root of the tree are all the arguments of the form $\langle \mathcal{A}, l \rangle$.
- The actions after a nonterminal history h are the arguments $\langle \mathcal{A}', q \rangle$ such that $\langle \mathcal{A}, p \rangle \leq \langle \mathcal{A}', q \rangle$, where $\langle \mathcal{A}, p \rangle$ is the last component in h.
- The utility for the proponent assumes the value 1(win) at a history $h \in Z$ if the length of h is odd, and -1 otherwise. The utility for the opponent is -1 times the utility of the proponent.

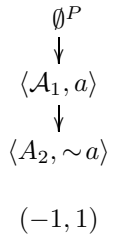
If we have that the relation $\leq$ is simply the relation $R = \bigcup_{h \in \text{Lit}} R_h$, and a literal l has some argument which can be attacked, then we will have an infinite tree.

Example 7. We build the warrant game for the literal a in the program $\mathbb{P}_1$ from example 1, assuming that $\leq = R$.



Notice that in the diagram we indicate the nodes by a single argument. The histories can be reconstructed by tracing the path in the tree up to the root. The superscript P or O on the root indicates the player who moves first.

If we consider instead the relation $\leq$ to favor arguments based on $\mathcal{A}_2$ to those based on $\mathcal{A}_1$, the game gets reduced to



In this tree we have a terminal history and we have written below the payoffs for the players *Proponent* and *Opponent*, respectively.

A plan for a given player, in which she has a response for every possible contingency of the game is called a *strategy*:

Definition 10. *[OR94] A* strategy *for a player $i \in N$ in an extensive game with perfect information $\langle N, H, P, U_n \rangle$ is a function that assigns an action in $A(h)$ to each nonterminal history $h \in H \setminus Z$ for which $P(h) = i$.*

Since players are rational, they will seek to get the highest utility. In order to do that each one will seek the best possible strategy. The joint strategy profiles of all players yield a single history. In the case of warrant games, since each terminal history pays either 1 or -1, we can in principle define a *winning strategy* for a player:

Definition 11. *A* winning strategy *for one of the players in a warrant game is a strategy that yields a terminal history $z \in Z$ such that her utility is* 1*, no matter what the other player's actions are.*

It turns out that in some warrant games (for example those without terminal histories), none of the players has a winning strategy.

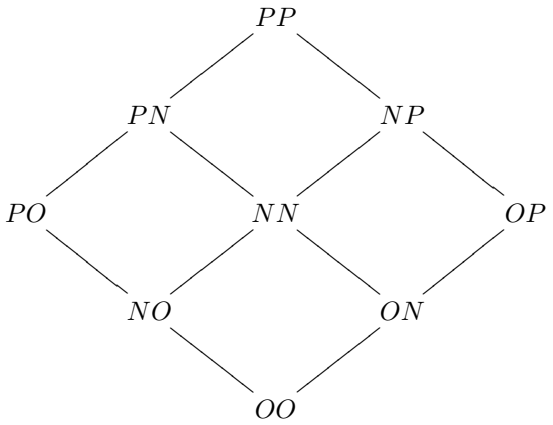
Now we pose a *query* to $\mathbb{P}$. The query is a literal l. Then we analyze two associated games. In the first place, we look at the warrant game for the literal l, and then the warrant game for the complement literal $\bar{l}$ in which the *Proponent* and *Opponent* change their roles. That is, the *Opponent* starts the game by choosing an argument for $\bar{l}$.

This presents a slight departure from the formalism presented in [GS04], where the existence of a winning strategy for the proponent of a literal l is enough to yield the answer "yes" to a query. On the other hand, in that paper, to yield the answer "no" to l, the status of $\bar{l}$ is analyzed, which clearly suggests the road we have taken here.

The following table summarizes the possible outcomes (meaning which of the players has a winning strategy) of both games and how they jointly yield an answer to the query:

Warrant game for l	Warrant game for $\bar{l}$	Answer to the query
Proponent	*Proponent*	YES
Proponent	*Opponent*	Undecided
Proponent	None	yes
Opponent	*Proponent*	Undecided
Opponent	*Opponent*	NO
Opponent	None	no
None	*Proponent*	yes
None	*Opponent*	no
None	None	Undecided

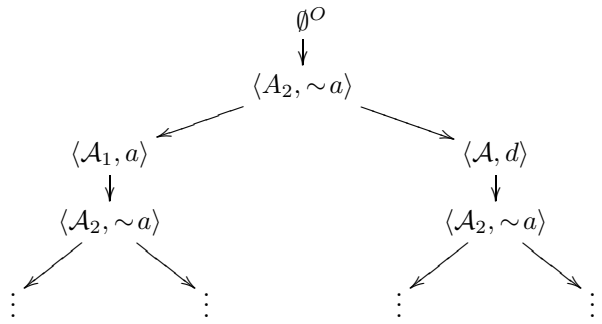
Thus we have a system in which a literal l can take different truth values which can be displayed in a partially ordered set:



where each pair of letters indicates first who has a winning strategy in the game for l and then who has a winning strategy in the game for $\bar{l}$. P corresponds to the *Proponent*, O to the *Opponent* and N, in turn, indicates that none of them has a winning strategy.

We need an interpretation for each of these outcomes. We have marked in the table with YES and NO the cases in which the arguments are clearly settled for or against the literal l. If for l the *Proponent* has a winning strategy and nor she or the *Opponent* have one for the warrant game on $\bar{l}$, we want to give a positive answer for the literal l, but not as strong one as we would for the case in which the *Proponent* has a winning strategy for both games. This presents a slight departure from the formalism presented in [GS04].

Example 8. We look now at the warrant game initiated by the *Opponent* for the literal $\sim a$ in the conditions we established in Example 7. If $\leq = R$, the game is



so the outcome for the query a is NN.

On the other hand, if the relation $\leq$ is empty, we just get

$$\emptyset^O$$
$$\downarrow$$
$$\langle A_2, \sim a \rangle$$

$$(-1, 1)$$

So here we get the result OO, and the answer "NO" for the query a.

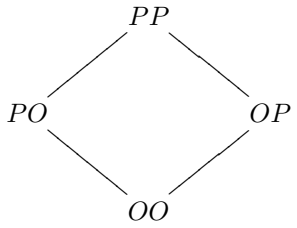
We have the following trivial result:

Proposition 12. *The literals that are facts of a program always get the outcome PP and therefore the answer YES.*

Proof. Since the class of facts is not contradictory, given a fact l, we know by Lemma 7, 1 that $\langle \emptyset, l \rangle$ is an argument and it has no counter-arguments. Therefore, the game for l ends after the *Proponent* chooses $\langle \emptyset, l \rangle$ and since there is no counterargument, she wins. Alternatively, for $\bar{l}$, there can be no arguments so the *Opponent* has no valid moves and the *Proponent* wins again. Therefore, the corresponding element in the lattice is PP, which yields a YES answer.

4 Finite Warrant Games

If we restrict the definition of warrant games in a way that makes them finite, then in each game one of the players has a winning strategy. The reason for this is that finite games are always *determined* [Myc92]. Our set of possible results (or truth-values) for a given query gets reduced to four possibilities: PP, PO, OP and OO.



The top and bottom yield, respectively the YES and NO answers to the query. The middle possibilities yield $UNDECIDED$, but with different meanings. If we obtain PO, both the literal l and its negation are warranted and we are facing a (defeasible) contradiction, while if the answer is OP, neither the literal nor its negation can be convincingly supported.

As we noted before, if we just let the defeating relation be $R = \bigcup_{h \in \text{Lit}} R_h$, then all the branches of a warrant game have length 1 (in the case that a literal has an argument for it that is not attacked) or are infinite.

We claim that the following feature of $\leq$ ensures the finiteness of the warrant games:

Definition 13. *The relation $\leq$ is said to be* s-acyclic *if there is no sequence $\langle \mathcal{A}_0, h_0\rangle$,$\langle \mathcal{A}_1, h_1\rangle$,...,$\langle \mathcal{A}_{k+1}, h_{k+1}\rangle$ in* Arg($\mathbb{P}$)*, such that $\langle \mathcal{A}_j, h_j\rangle \leq \langle \mathcal{A}_{j+1}, h_{j+1}\rangle$, for $j = 0, \ldots, k$, and $\langle \mathcal{A}_{k+1}, h_{k+1}\rangle$ is a subargument of $\langle \mathcal{A}_0, h_0\rangle$.*

Then:

Proposition 14. *If $\leq$ is s-acyclic, the warrant game for any literal l is finite.*

Proof. Each history h in the warrant game is a sequence $\langle \mathcal{A}_0, h_0\rangle$, ..., $\langle \mathcal{A}_K, h_K\rangle$, where $\langle \mathcal{A}_j, h_j\rangle \leq \langle \mathcal{A}_{j+1}, h_{j+1}\rangle$, for $j = 0, \ldots, K-1$. Since the set Arg($\mathbb{P}$) is finite and $\leq$ is s-acyclic, K must be finite as well.

Instead of imposing conditions on $\leq$, desirable outcomes can be ensured by means of a *protocol* that restricts the admissible actions that may be taken in a warrant game. The main idea here is that one doesn't want to allow repetition of arguments, or subarguments, and each player should maintain internal consistence in the arguments she supports.

Let $\approx$ be the binary relation between arguments defined by $\langle \mathcal{A}_1, h_1\rangle \approx \langle \mathcal{A}_2, h_2\rangle$ iff $\langle \mathcal{A}_1, h_1\rangle \leq \langle \mathcal{A}_2, h_2\rangle$ and $\langle \mathcal{A}_2, h_2\rangle \leq \langle \mathcal{A}_1, h_1\rangle$. DeLP assumes the following protocol:

Definition 15. *In a warrant game for a literal l, a history is $h = \langle \mathcal{A}_0, h_0\rangle$, ..., $\langle \mathcal{A}_K, h_K\rangle$, where $\langle \mathcal{A}_j, h_j\rangle \leq \langle \mathcal{A}_{j+1}, h_{j+1}\rangle$, for $j = 0, \ldots, K-1$. We say that h is* DeLP-admissible *if and only if:*

- *Given $\langle \mathcal{A}_j, h_j\rangle$, $\langle \mathcal{A}_{j+1}, h_{j+1}\rangle$ and $\langle \mathcal{A}_{j+2}, h_{j+2}\rangle$ in h, if $\langle \mathcal{A}_j, h_j\rangle \approx \langle \mathcal{A}_{j+1}, h_{j+1}\rangle$, then it is not the case that $\langle \mathcal{A}_{j+2}, h_{j+2}\rangle \leq \langle \mathcal{A}_{j+1}, h_{j+1}\rangle$. In other words, a blocking defeater can only be followed by a proper defeater.*
- *The sets $\bigcup_{j\geq 0} \mathcal{A}_{2j}$ and $\bigcup_{j\geq 0} \mathcal{A}_{2j+1}$ are not contradictory (concordance).*
- *Every subsequence of any history of the game $h = \langle \mathcal{A}_0, h_0\rangle, \ldots, \langle \mathcal{A}_K, h_K\rangle$ is an s-acyclic sequence.*

Proposition 16. *A warrant game in which each history is DeLP-admissible is finite.*

Proof. Immediate, since every h is DeLP-admissible, h is a s-acyclic sequence. Given that Arg($\mathbb{P}$) is finite, the length of h is finite.

5 The *Defeating* Relation and Truth Values

Now we turn our attention to two extreme cases for the relation $\leq$ and see how it restricts the DeLP-admissibility of histories and therefore the winning conditions of warrant games. That is, how it partitions Lit in classes corresponding each to the four possibilities in finite games.

Proposition 17. *If $\leq = \emptyset$, then a player has a winning strategy for a warrant game for a given literal l if and only if there exists at least one argument for the literal according to the program.*

Proof. Immediate from the fact that the first move for the *Proponent* (in the game for l) or the *Opponent* (in the game for $\bar{l}$) is to state an argument for either l or $\bar{l}$ and there is no argument the other party can use to counterargue, so the game ends. If one of the players cannot find an argument for her literal the other wins.

This means that for each literal l, if there exist arguments for both the literal and its negation, we get the truth value PO, i.e. a defeasible contradiction. Otherwise, if either l or $\bar{l}$ has no supporting arguments in $\mathbb{P}$ while the other has at least one, the truth value of l will be PP or OO.

Proposition 18. *If* $\leq = R = \bigcup_{h \in \mathrm{Lit}} R_h$ *the DeLP-admissibility implies that every terminal history of the warrant games has length at most two.*

Proof. Since $\leq$ coincides with the attacking relation R, all defeaters are *blocking* defeaters, so no history can have length more than two. There can be terminal histories with length one: those consisting of single arguments that have no attackers, as for instance the facts of the program. Finally, we can have trees with a single node with the empty sequence in case the queried literal has no arguments supporting it.

In this case, if the game for a literal l has a terminal history of length one, the *Proponent* can choose it and win both the games for l and $\bar{l}$, i.e. the truth value of l becomes PP. The same is true if the game for $\bar{l}$ has a terminal history of length one: the truth value of l is OO. In the cases in which the lengths of terminal histories are two, each of the players has a winning strategy which yields a win by blocking the first argument in the history. Therefore, OP arises as the truth value of l.

6 Conclusions and Future Work

We have re-casted the basic definitions of DeLP in a simple mathematical language so that we can analyze its underlying assumptions in a new light. We hope that the proposed formal framework can shed some light on how the warrant mechanism works and in particular, on the role of the *defeating* relation. We have borrowed a page from game theory to present the dialectical process in what we believe is a natural way. By initially dropping all restrictions in the dialectical process we uncovered more possibilities for the outcome of a proposed query. We can see these outcomes as truth values, yielding more information on the nature of what a given DeLP program concludes about each literal.

We checked that the facts of the program get a positive answer before turning to the conditions that make our *warrant games* finite. These conditions can come either from the *defeating* relation itself or from an imposed *protocol* on the way the games are constructed. Once we have a way that the games considered are finite, we have fewer truth values. We analyzed the effect of having the extreme cases of an empty *defeating* relation and also the biggest possible one.

Game semantics for DeLP have already been introduced in in [CS04] and [CFS06] The main differences with our approach are that, on one hand, they do not apply standard notions of game theory, while on the other hand they restrict themselves to a single game, that may yield only three truth values. Their notions of strategy and of when a game is won lead to the collapse of several of our truth values into one of theirs. While this is enough for them, since they seek a game semantics independent of the features of the defeating relation, our approach allows to detect fine-grained details of how DeLP works, in particular how it varies according to that relation.

The inference procedure associated to finding winning strategies has a natural "semantical" counterpart. That is, the pair of winners, one for each of the two games can be immediately associated to a truth value as described in the table in section 3. In turn, this means that for each defeating relation we have a partition of the class of literals associated to a DeLP program. As it can be seen from Propositions 17 and 18, the partitions may overlap, even for a pair of defeating relations in which one is a subset of the other.

This framework of analysis can be extended to other argumentative system. In a system as DeLP, where arguments support certain logical formulas[2]. Then, any defeating relation among these arguments may be applied to yield games for a formula and its negation. The properties of the defeating relation determine the actual partition of the class of formulas.

In a more general setting, when arguments are abstract entities there is no "negation" involved. But then, we can still partition the class of arguments in terms of a single game for an argument. If there is a winning strategy for it, it is deemed true.

As a next step we want to study some of the proposed intermediate *defeating* relations like *specificity* and look for desirable properties the defeating relations should have.

References

[AC02] Amgoud, L., Cayrol, C.: A reasoning model based on the production of acceptable arguments. Annals of Mathematics and Artificial Intelligence 34, 197–215 (2002)

[BH01] Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artificial Intelligence 128, 203–235 (2001)

[Bre01] Brewka, G.: Dynamic argument systems: a formal model of argumentation processes based on situation calculus. Journal of Logic and Computation 11, 257–282 (2001)

[CDSS03] Chesñevar, C., Dix, J., Stolzenburg, F., Simari, G.: Relating defeasible and normal logic programming through transformation properties. Theor. Comput. Sci. 290(1), 499–529 (2003)

[2] The reason why DeLP is concerned only with literals is the one why LP is: to facilitate the implementation of working interpreters (see [Llo87]).

[CFS06] Cecchi, L., Fillottrani, P., Simari, G.: On the complexity of delp through game semantics. In: Dix, J., Hunter, A. (eds.) NMR 2006. Proc. 11th Intl. Workshop on Nonmonotonic Reasoning. IfI Technical Report Series, pp. 386–394. Clausthal University, Windermere (2006)

[CML00] Chesñevar, C., Maguitman, A., Loui, R.: Logical models of argument. ACM Computing Surveys 32(4), 337–383 (2000)

[CS04] Cecchi, L., Simari, G.: Sobre la relación entre la semántica gs y el razonamiento rebatible. In: X CACiC, pp. 1883–1894. Universidad Nacional de La Matanza (2004)

[Dun95] Dung, P.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming, and n-person games. Artificial Intelligence 77, 321–357 (1995)

[GS04] García, A., Simari, G.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming 4(1), 95–138 (2004)

[Ham70] Hamblin, C.: Fallacies. Methuen, London (1970)

[Ham71] Hamblin, C.: Mathematical models of dialogue. Theoria 37, 130–155 (1971)

[Hin73] Hintikka, J.: Language Games and Information. Clarendon Press, London (1973)

[HS97] Hintikka, J., Sandu, G.: Game-Theoretical Semantics. In: Handbook of Logic and Language, Elsevier, Amsterdam (1997)

[LAP00] Maudet, N., Amgoud, L., Parsons, S.: Modelling dialogues using argumentation. In: Proceedings of the Fourth International Conference on Multi-Agent Systems. Boston, MA, pp. 31–38 (2000)

[LL78] Lorenzen, P., Lorenz, K.: Dialogische Logik. Wissenschaftliche Buchgesellschaft, Darmstadt (1978)

[Llo87] Lloyd, J.: Foundations of Logic Programming, 2nd edn. Springer, Heidelberg (1987)

[Lou98] Loui, R.: Process and policy: Resource-bounded non-demonstrative reasoning. Computational Intelligence 14, 1–38 (1998)

[MP02] McBurney, P., Parsons, S.: Games that agents play: A formal framework for dialogues between autonomous agents. Journal of Logic, Language and Information 13, 315–343 (2002)

[Myc92] Mycielski, J.: Games with perfect information. In: Aumann, R.J., Hart, S. (eds.) Handbook of Game Theory with Economic Applications, ch. 3, vol. 1, pp. 41–70. Elsevier, Amsterdam (1992), `http://ideas.repec.org/h/eee/gamchp/1-03.html`

[Nut94] Nute, D.: Defeasible logic. In: Gabbay, D., Hogger, C.J., Robinson, J.A. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming. Nonmonotonic Reasoning and Uncertain Reasoning, vol. 3, pp. 353–395. Oxford University Press, Oxford (1994)

[OR94] Osborne, M., Rubinstein, A.: A course in game theory. MIT Press, Cambridge (1994)

[Pol87] Pollock, J.: Defeasible reasoning. Cognitive Science 11(4), 481–518 (1987)

[Pol95] Pollock, J.: Cognitive Carpentry: A Blueprint for how to Build a Person. MIT Press, Cambridge (1995)

[Poo85] Poole, D.: On the comparison of theories: Preferring the most specific explanation. In: IJCAI, pp. 144–147 (1985)

[Pra05] Prakken, H.: Coherence and flexibility in dialogue games for argumentation. Journal of Logic and Computation 15, 1009–1040 (2005)

[PV02] Prakken, H., Vreeswijk, G.: Logics for defeasible argumentation. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 4, pp. 219–318. Kluwer Academic, Dordrecht (2002)

[PWA03] Parsons, S., Wooldridge, M., Amgoud, L.: Properties and complexity of some formal interagent dialogues. Journal of Logic and Computation 13, 347–376 (2003)

[SGCS03] Stolzenburg, F., García, A., Chesñevar, C., Simari, G.: Computing generalized specificity. Journal of Applied Non-Classical Logics 13(1), 87 (2003)

[SL92] Simari, G., Loui, R.: A mathematical treatment of defeasible reasoning and its implementation. Artif. Intell. 53(2–3), 125–157 (1992)

[vB02] van Benthem, J.: Extensive games as process models. Journal of Logic, Language and Information 11, 289–313 (2002)