A Preliminary Approach Towards a Logic for Warrant

Sergio Alejandro Gómez and Guillermo Ricardo Simari

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
Email: {sag, grs}@cs.uns.edu.ar

Abstract. We extend Defeasible Logic Programming for it to be able to warrant complex logical formulas. We show how negation, conjunction, disjunction and implication of ground literals can be tested for warrant. We show a running scenario to test the suitability of the approach.

Keywords: Defeasible Logic Programming, Argumentation, Artificial Intelligence.

1 Introduction

Defeasible Logic Programming (DeLP) is an approach to non-monotonic reasoning [1] based on argumentation [2–4] and logic programming [5], that provides a good trade-off between efficiency, expressiveness and implementability. One limitation of DeLP is that the output of the system is restricted to the warrant of single literals (that is, positive or classically negated atoms). There are cases, however, where the need for warranting more complex expressions arises. We are concerned with extending DeLP's reasoning capabilities for warranting complex logical expressions.

We will propose a two-layer architecture (see Figure 1). The bottom layer is made up by the DeLP engine and the top layer is composed of the logic for warrant presented in this work. DeLP will be used for computing the warrant of single literals (either positive or classically-negated ground atoms) with respect to a DeLP program. The warrants obtained will be combined in complex logical expressions to produce a higher-level warrant. Although, a priori, any logic can be considered for the top layer, in this work we will consider a complex expression as a propositional logic formula. So, the main contribution of this paper is extending DeLP for warranting literals combined arbitrarily with negation, conjunction, disjunction, implication and double implication. We will therefore define the semantics of these operations precisely in the context of the answers obtained by a DeLP interpreter.

The rest of this presentation is structured as follows. In Section 2, we present a brief introduction to Defeasible Logic Programming. In Section 3, we present

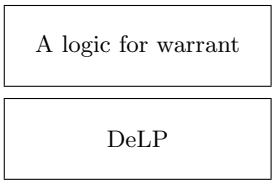

Fig. 1. Layered approach to reasoning

an extension to DeLP to be able to warrant (if one exists) of arbitrarily complex propositional expressions. In Section 4, we discuss related work. Finally, in Section 5, we conclude pointing out lines of future research.

2 Fundamentals of Defeasible Logic Programming

Defeasible Logic Programming (DeLP) [6] provides a language for knowledge representation and reasoning that uses *defeasible argumentation* to decide between contradictory conclusions through a *dialectical analysis*, and providing a good trade-off between expressiveness and implementability for dealing with incomplete and potentially contradictory information. In a DeLP program $\mathcal{P} = (\Pi, \Delta)$, a set Π of strict rules $P \leftarrow Q_1, \ldots, Q_n$ (which encode certain knowledge), and a set Δ of defeasible rules $P \prec Q_1, \ldots, Q_n$ (which encode knowledge with possible exceptions) can be distinguished. An *argument* $\langle \mathcal{A}, H \rangle$ is a minimal non-contradictory set of ground defeasible clauses $\mathcal{A}$ of Δ that allows to derive a ground literal H possibly using ground rules of Π. Since arguments may be in conflict (concept captured in terms of a logical contradiction), an attack relationship between arguments can be defined. To decide between two conflicting arguments, we will use *generalized specificity*—a syntactic criterion that prefers arguments more informed and arguments based on shorter derivations. If the attacking argument is strictly preferred over the attacked one, then it is called a *proper defeater*. If no comparison is possible, or both arguments are equi- preferred, the attacking argument is called a *blocking defeater*. To determine whether a given argument $\mathcal{A}$ is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for $\mathcal{A}$, defeaters for these defeaters, and so on, are taken into account. Given a DeLP program $\mathcal{P}$ and a query H, the final answer to H w.r.t. $\mathcal{P}$ is based on such dialectical analysis. The answer to a query can be: *Yes* (when there exists a warranted argument $\langle \mathcal{A}, H \rangle$), *No* (when there exists a warranted argument $\langle \mathcal{A}, \sim H \rangle$), *Undecided* (when neither $\langle \mathcal{A}, H \rangle$ nor $\langle \mathcal{A}, \sim H \rangle$ are warranted), or *Unknown* (when H does not belong to $\mathcal{P}$).

An important property in DeLP is that it is impossible to find a warrant both for a literal and its negation simultaneously. Formally:

Property 1. Let $\mathcal{P} = (\Pi, \Delta)$ be a DeLP program and L a literal in $\mathcal{P}$. It cannot be the case that the answer for L and $\sim L$ is both Yes.

For motivating our work, we are going to use a running scenario, but notice that despite this, our approach can be considered general. We will consider an intelligent agent to which we want to endow the capability of reasoning with possibly inconsistent and incomplete knowledge bases. The agent will sense the world, collecting facts F. The agent's model of the world will be composed of two kinds of knowledge: a set R of strict rules (containing information that can be regarded as certain and, thus consistent although perhaps incomplete) and a set D of defeasible rules (containing information that can be viewed as tentative, thus representing situations where both exceptions can exist and inconsistency and incompleteness may coexist). Upon presented to the world, the conclusions of the agent are going to be complex expressions formed by ground literals connected by classical propositional logic operators. For simplicity, we will assume that sensor feed can be incomplete but it is always consistent (if a pair of contradictory literals are received, they will be discarded).

As a possible instance of the agent described above, let us imagine a robot exploring the surface of an extraterrestrial body, it receives sensor feed about its environment and, after analyzing the information provided by those sensors, has to decide which actions it must perform based on its internal model. A possible criteria could be that the robot will move to explore a certain area whenever there is reasonable evidence that the area is both promising and not dangerous. The reasons for determining if a place is promising can be incomplete and maybe contradictory, and the same could happen with the reasons for determining if a place is dangerous.

We will see how DeLP can be used to represent the internal model of the robot as well as sensor feed information and how top level conclusions of the system, such as "we believe that the area to be explored by the robot is both promising and not dangerous" can be modeled by what we call *a logic for warrant*, that is a logic for representing complex logical expressions that can be mounted of top of DeLP for performing complex reasoning beyond classical DeLP mechanisms. The novelty of our approach will be that the answer of the system will be a combination of the warrants of each individual conclusion that will be defined according to a very precise semantics. In short, the agent's model will be modeled as a DeLP program $(S \cup F, R)$. High-level conclusions such as "the place a is both promising and not dangerous" will be represented as $\triangle(promising(a)$ AND NOT$dangerous(a))$, meaning that there are warrants both for a is promising and a is not dangerous. In the next example, we present a DeLP program for modeling this situation and showing how DeLP is used for computing the warrant of simple literals. In Section 3, we will then show how this scenario can be used to show how DeLP can be extended to warrant complex logical expressions.

Example 1. Consider the following DeLP program $\mathcal{P}_1 = (\Pi, \Delta)$ representing sensor feed plus internal model of the interplanetary probe:

$$\Pi = \left\{ \begin{array}{l} area(mare\_crisium). \\ area(mare\_serenitatis). \\ area(mare\_tranquillitatis). \\ area(utopia\_planitia). \\ expert(john). \\ expert(paul). \\ says\_it\_is\_promising(john, mare\_serenitatis). \\ says\_it\_is\_not\_promising(john, mare\_crisium). \\ says\_it\_is\_promising(john, utopia\_planitia). \\ says\_it\_is\_not\_promising(paul, utopia\_planitia). \\ satellite\_view(mare\_tranquillitatis, v1). \\ consistent\_with\_unobtainium\_presence(v1). \\ \sim dangerous(mare\_serenitatis). \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} promising(A) \prec area(A), expert(E), says\_it\_is\_promising(E, A). \\ promising(X) \prec \\ \qquad satellite\_view(X, V), \\ \qquad consistent\_with\_unobtainium\_presence(V). \\ \sim promising(A) \prec area(A), expert(E), says\_it\_is\_not\_promising(E, A). \end{array} \right\}$$

Given this DeLP program, if we happen to have exactly one opinion on a given subject, we are going to get either one answer. For example, in this case we can see that Mare Crisium is not promising but Mare Serenitatis is because John, who happen to be a an expert, says so. Formally, we have two undefeated arguments $\langle \mathcal{A}_1, \sim promising(mare\_crisium) \rangle$ and $\langle \mathcal{A}_2, promising(mare\_serenitatis) \rangle$, where:

$$\mathcal{A}_1 = \left\{ \begin{array}{l} \sim promising(mare\_crisium) \prec \\ \qquad area(mare\_crisium), expert(john), \\ \qquad says\_it\_is\_not\_promising(john, mare\_crisium) \end{array} \right\}$$

$$\mathcal{A}_2 = \left\{ \begin{array}{l} promising(mare\_serenitatis) \prec \\ \qquad area(mare\_serenitatis), expert(john), \\ \qquad says\_it\_is\_promising(john, mare\_serenitatis) \end{array} \right\}$$

In this case, the answers to the queries $promising(mare\_crisium)$ and $promising($ $mare\_serenitatis)$ are *No* and *Yes* respectively.

On the other hand, if we have two experts whose opinions about a certain are in clash, as is the case of John and Paul respect to Utopia Planitia, then we will have two equi-preferable arguments and DeLP's answer with respect to $promising(utopia\_planitia)$ is going to be *Undecided*.

If we wanted to implement a voting protocol for making decisions, such as for considering what happens when two experts have concurring opinions about a certain issue in contrast to only one expert affirming the opposite, we could manage to use a rule such as the following:

$$\begin{array}{l} promising(A) \prec \\ \quad area(A), expert(E1), expert(E2), E1 \neq E2, \\ \quad says\_it\_is\_promising(E1, A), says\_it\_is\_promising(E2, A). \end{array}$$

An argument based on this rule would be more informed than the one based on the the rule that uses the opinion of only one expert, thus making it more specific and therefore a defeater.

Finally, if instead of relying on expert opinions, we wanted to rely on sensor information such as satellite information, we could find a compelling argument for supporting the conclusion that the area called Mare Tranquillitatis is promising because the satellite view shows that its composition is compatible with the presence of a strange mineral called unobtainium. Notice also that we will asume we kwow that the area known as Mare Serenitatis a non-dangerous one (in an actual scenario, this could have been modeled by a set of rules); for simplicity we assume just a fact $\sim dangerous(mare\_serenitatis)$, so the argument $\langle \emptyset, \sim dangerous(mare\_serenitatis) \rangle$ is trivially justified and the answer to the query $\sim dangerous(mare\_serenitatis)$ is *Yes*.

3 Warranting Complex Logical Expressions

As shown in the previous section, DeLP is able to find warrants for literals provided that they exist. Nevertheless, there are situations when warrants for more complex conclusions need to be found. For instance, suppose that the probe introduced in Example 1 needs to determine if an area is both promising and not dangerous in order to proceed with its exploration. Certainly, it would be fairly easy to model situations when the composition of negation and conjunction would be needed. For instance, this could be achieved by adding a rule of the form "$proceed(A) \prec promising(A), \sim dangerous(A)$". However, it would be impossible to add other constructs due to the intrinsic limitations imposed by the logic programming approach to knowledge representation underlying DeLP.

In the light of the above reasons, we propose extending DeLP's reasoning capabilities for warranting complex logical expressions. Then, we first define some notation for reifying the notion of warrant, and define what we consider to be complex expressions. After that, we describe how to compute warrants of complex expressions, presenting some of the properties that we have found.

Definition 1 (Warrant). *Let $\mathcal{P}$ be a DeLP program. Let L be a literal in $\mathcal{P}$. We note that L is warranted with respect to $\mathcal{P}$ as $\triangle_{\mathcal{P}} L$ (if the program $\mathcal{P}$ is clear from context, we just write $\triangle L$).*

The basic DeLP formalism is just able to find a warrant for a literal, that is a positive atom or its negation. So the answer for L is *Yes* if there is a warrant for L, and the answer for L is *No* if there is a warrant for $\sim L$.

Definition 2 (Complex expression). *Complex expressions in DeLP are defined recursively as:*

- *If L is a literal then L is a complex expression;*
- *(Negation) if L is a literal then $\mathsf{NOT} L$ is a complex expression;*
- *(Conjunction) if L_1 and L_2 are literals, then L_1 AND L_2 is a complex expression;*

- *(Disjunction) if L_1 and L_2 are literals, then L_1 OR L_2 is a complex expression;*
- *(Implication) if L_1 and L_2 are literals, then $L_1 \Rightarrow L_2$ is a complex expression;*
- *(Double implication) if L_1 and L_2 are literals, then $L_1 \Longleftrightarrow L_2$ is a complex expression, and,*
- *(Closure) there is no other complex expression.*

We are now going to extend the notion of warrant for complex expressions. In the case of negation, the definition is consistent with the one already existing in DeLP. The definitions for the other operators are the main contribution of this work. We define them formally motivating the form of our definitions and exemplify its behavior to show that they are appropriate.

Definition 3 (Warrant for Negation). *Let $\mathcal{P}$ be a DeLP program and L a literal. The complex expression* NOTL *is warranted, noted as $\triangle$(NOTL), if and only if $\sim L$ is warranted. The answer for the query* NOTL *is:*

- *Yes: whenever $\triangle(\sim L)$;*
- *No: if $\triangle L$, and,*
- *Undecided: when none of the above holds.*

Example 2. The answer for NOT*promising*(*mare_crisium*) is *Yes* because, from Example 1, the answer for $\sim$*promising*(*mare_crisium*) is *Yes*. Likewise the answer for NOT*promising*(*mare_serenitatis*) is *No*. In the same line of thought, the answer for NOT*dangerous*(*mare_serenitatis*) is *Yes* but the answer for the complex expression NOT*dangerous*(*mare_crisium*) is *Undecided*.

Definition 4 (Warrant for conjunction). *The complex expression P* AND *Q is warranted, that is $\triangle(P$* AND *$Q)$, if and only if both $\triangle P$ and $\triangle Q$. The answer for a query P* AND *Q will be:*

- *Yes: when both $\triangle P$ and $\triangle Q$;*
- *No: whenever $\triangle$(NOTP) or $\triangle$(NOTQ), and*
- *Undecided: when none of the above holds.*

We first present a very simple example to show how the definition works:

Example 3. Consider the program $\mathcal{P} = \{(a \multimap b), b, (c \multimap d), d\}$. In this program there is a warranted argument for the literal a and a waranted argument for the literal c, namely: $\langle\{a \multimap b\}, a\rangle$ and $\langle\{c \multimap d\}, c\rangle$. Therefore the complex formula a AND c is warranted, that is $\triangle(a$ AND $c)$.

Example 4. Recalling Example 1, the answer for the query "*promising*(*mare_serenitatis*)" is *Yes* and the answer for "$\sim$*dangerous*(*mare_serenitatis*)" is also *Yes*. Therefore the answer for

promising(*mare_serenitatis*) AND NOT*dangerous*(*mare_serenitatis*)

is *Yes*. When we consider the status of warrant of

$$promising(mare\_crisium) \text{ AND NOT } dangerous(mare\_crisium)$$

we can determine that it is *No* because $\triangle(\text{NOT } promising(mare\_crisium))$. In the case of

$$promising(utopia\_planitia) \text{ AND NOT } dangerous(utopia\_planitia)$$

the answer is *Undecided*.

Definition 5 (Warrant for disjunction). *The complex expression P OR Q is warranted, that is $\triangle(P \text{ OR } Q)$, if and only if $\triangle P$ or $\triangle Q$. The answer for a query P OR Q will be:*

- *Yes: whenever $\triangle P$ or $\triangle Q$;*
- *No: whenever both $\triangle(\text{NOT}P)$ and $\triangle(\text{NOT}Q)$, and*
- *Undecided: when none of the above holds.*

Definition 6 (Warrant for implication). *The complex expression $P \Rightarrow Q$ is warranted, that is $\triangle(P \Rightarrow Q)$, if and only if the expression $((\text{NOT}P) \text{ OR } Q)$ is warranted.*

Property 2. The answer for a query $P \Rightarrow Q$ will be:

1. Yes: whenever $\triangle(\text{NOT}P)$ or $\triangle Q$;
2. No: whenever both $\triangle P$ and $\triangle(\text{NOT}Q)$, and
3. Undecided: when none of the above holds.

Proof. The proof is based in the truth table for $(\text{NOT}P) \text{ OR } Q$ and is based on cases:

P	Q	NOTP	NOTQ	(NOTP) OR Q
Yes	Yes	No	No	Yes
Yes	No	No	Yes	No
Yes	Undecided	No	Undecided	Undecided
No	Yes	Yes	No	Yes
No	No	Yes	Yes	Yes
No	Undecided	Yes	Undecided	Yes
Undecided	Yes	Undecided	No	Yes
Undecided	No	Undecided	Yes	Undecided
Undecided	Undecided	Undecided	Undecided	Undecided

Notice that the answer is Yes in each row in which the value for NOTP or for Q is Yes; the answer is No in the only row where the value of both P and NOTQ is Yes, and Undecided in the rest of the rows.

Notice also that this truth table coincides with the one in Kleene's three-valued logic.

Definition 7 (Warrant for double implication). *The complex expression* $P \iff Q$ *is warranted, that is* $\triangle(P \iff Q)$*, if and only if it is the case that both* $\triangle((P \Rightarrow Q)$ AND $(Q \Rightarrow P))$*.*

Property 3. The answer for a query $P \iff Q$ is one of:

- Yes: when either $\triangle P$ and $\triangle Q$, or $\triangle(\mathsf{NOT}P)$ and $\triangle(\mathsf{NOT}Q)$;
- No: when either $\triangle P$ and $\triangle(\mathsf{NOT}Q)$, or $\triangle(\mathsf{NOT}P)$ and $\triangle Q$.
- Undecided: when none of the above holds.

Proof. The proof, again, is based in the truth table for $(P \Rightarrow Q)$ AND $(Q \Rightarrow P)$ and is based on cases:

P	Q	$P \Rightarrow Q$	$Q \Rightarrow P$	$(P \Rightarrow Q)$ AND $(Q \Rightarrow P)$
Yes	Yes	Yes	Yes	Yes
Yes	No	No	Yes	No
Yes	Undecided	Undecided	Yes	Undecided
No	Yes	Yes	No	No
No	No	Yes	Yes	Yes
No	Undecided	Yes	Undecided	Undecided
Undecided	Yes	Yes	Undecided	Undecided
Undecided	No	Undecided	Yes	Undecided
Undecided	Undecided	Undecided	Undecided	Undecided

It is easy to check that the value is Yes whenever both P and Q are Yes or No, resp.; the value is No if P is Yes and Q is No and viceversa, and the value is Undecided in the rest of the cases.

Two properties than can also be checked using the truth-table method are the following:

Property 4. If we have $\triangle(P$ OR $Q)$ and $\triangle(\mathsf{NOT}P)$ then we have $\triangle Q$.

Property 5. If we have that $\triangle(P \Rightarrow Q)$ and $\triangle(Q \Rightarrow R)$ then we have that $\triangle(P \Rightarrow R)$.

Here we present a last property:

Property 6. It is never the case that $\triangle(P$ AND $\mathsf{NOT}P)$.

Proof. According to Property 1, in DeLP it cannot be the case that P and $\sim P$ are warranted at the same time. If the answer to P is Yes, then the answer to $\mathsf{NOT}P$ is No; therefore the answer to P AND $\mathsf{NOT}P$ is No. If the answer to P is No, then the answer to $\mathsf{NOT}P$ is Yes; therefore the answer to P AND $\mathsf{NOT}P$ is No. If the answer to P is Undecided, the same happens to the answer to $\mathsf{NOT}P$ and the answer to P AND $\mathsf{NOT}P$ is Undecided.

4 Related Work

The operators presented here are related to three-valued approaches to logic such as Kleene's and Lukasiewicz's, that uses the truth values false, unknown and true, and extend conventional Boolean connectives to a trivalent context. In those logics our value Yes is modeled as True, our No as False, and our Undecided as Unknown. Our definition of material implication coincides with the definition given by Kleene. Lukasiewicz's definition of material implication differs with the one given by Kleene, its truth-table is:

P	Q	$P \Rightarrow Q$
Yes	Yes	Yes
Yes	No	No
Yes	Undecided	Undecided
No	Yes	Yes
No	No	Yes
No	Undecided	Yes
Undecided	Yes	Yes
Undecided	No	Undecided
Undecided	Undecided	Yes

On the side of extensions to DeLP, we can trace several over the years: M. Gomez Lucero *et al.* developed accrual of arguments [7]; M.V. Martinez *et al.* extended the notion of reasoning in DeLP with presumptions (that is fact-like information that is not known with certainty to be true) [8]; Budan *et al.*, an approach for temporal argumentation in DeLP [9] (an approach that considers at which time arguments are active in DeLP considering attacks and defenses). In contrast to our approach, these works redefine fundamental notions of DeLP (such as disagreement, attack and defeat); in our case, we just use DeLP as a black box for obtaining the warrant of single literals which are later combined into higher-level answers.

Besnard and Hunter [10] focus on deductive arguments in the setting of classical logic. Their position is that a deductive argument consists of a claim entailed by a collection of statements such that the claim and the statements are denoted by formulas of classical logic and entailment is deduction in classical logic. In our case, we restrict our formulas to Horn-clauses, the conclusions of the system are positive or negative literals which are then combined through classical logic connectives to form more complex conclusions.

5 Conclusions and Future Work

We have presented an approach for extending Defeasible Logic Programming in order to being capable of warranting complex expressions formed by ground literals connected by propositional operations. We have presented a case scenario where this could be condidered useful. Much remains to be done, such us characterizing its properties, doing a thorough comparison with related work and determining how the warrant of complex formulas can be defined. These issues are currently part of our current research.

Acknowledgments: This research is funded by Secretaría General de Ciencia y Técnica, Universidad Nacional del Sur, Argentina.

References

1. Brewka, G., Dix, J., Konolige, K.: Non monotonic reasoning. An overview. CSLI Publications, Stanford, USA (1997)
2. Chesñevar, C.I., Maguitman, A., Loui, R.: Logical Models of Argument. ACM Computing Surveys **32**(4) (December 2000) 337–383
3. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artif. Intell. **171**(10-15) (2007) 619–641
4. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence. Springer (2009)
5. Lloyd, J.: Foundations of Logic Programming. Springer-Verlag (1987)
6. García, A., Simari, G.: Defeasible Logic Programming an Argumentative Approach. Theory and Practice of Logic Programming **4**(1) (2004) 95–138
7. Lucero, M.J.G., Chesñevar, C.I., Simari, G.R.: On the accrual of arguments in defeasible logic programming. In Boutilier, C., ed.: IJCAI. (2009) 804–809
8. Martinez, M.V., García, A.J., Simari, G.R.: On the use of presumptions in structured defeasible reasoning. In Verheij, B., Szeider, S., Woltran, S., eds.: COMMA. Volume 245 of Frontiers in Artificial Intelligence and Applications., IOS Press (2012) 185–196
9. Budan, M., Lucero, M.G., Simari, G.: An approach for temporal argumentation using labeled defeasible logic programming. Journal of Computer Science and Technology **12**(2) (August 2012) 56–63
10. Besnard, P., Hunter, A.: Argumentation based on classical logic. In Rahwan, I., Simari, G.R., eds.: Argumentation in Artificial Intelligence. Springer (2009) 133–152