- ORIGINAL ARTICLE -

Reasoning with Inconsistent Possibilistic Ontologies by Applying Argument Accrual

Sergio Alejandro Gómez[1,2]

[1]*Laboratorio de Desarrollo e Investigación en Inteligencia Artificial (LIDIA), Instituto de Ciencias e Ingeniería de la Computación (ICIC), Departamento de Ciencias e Ingeniería de la Computación (DCIC), Universidad Nacional del Sur (UNS), San Andrés 800 - Campus de Palihue (8000) Bahía Blanca, Argentina, sag@cs.uns.edu.ar*
[2]*Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC-PBA)*

Abstract

We present an approach for performing instance checking in possibilistic description logic programming ontologies by accruing arguments that support the membership of individuals to concepts. Ontologies are interpreted as possibilistic logic programs where accruals of arguments as regarded as vertexes in an abstract argumentation framework. A suitable attack relation between accruals is defined. We present a reasoning framework with a case study and a Java-based implementation for enacting the proposed approach that is capable of reasoning under Dung's grounded semantics.

Keywords: Argument accrual, ontology reasoning, inconsistency handling, Description Logics.

1 Introduction

Reasoning with inconsistent ontologies plays a fundamental role in Semantic Web applications. An ontology defines axiomatically a set of concepts that, given assertions about the membership of individuals to some concepts, allows to determine the membership of individuals to concepts (a task known as instance checking). Ontologies can suffer from incoherence and inconsistency; either correcting these anomalies or dealing with them with non-monotonic reasoning techniques are the two main accepted solutions [1]. Argumentation [2] is an approach to defeasible reasoning that can be applied to handling inconsistency in ontologies. In argumentation, given an inconsistent knowledge base, arguments compete to decide which are the accepted consequences. Argument accrual formalizes the notion that having more arguments for a certain conclusion makes it more credible [3].

In this paper, that is an extended and revised version of [4], we explore the application of argument accrual to reasoning with inconsistent ontologies. To the best of our knowledge, argument accrual were firstly studied by [5] and its application to the problem of ontology reasoning has only been suggested by Groza [6] in the context of fuzzy description logics. We use possibilistic description logic programming as the language for ontology representation (i.e. an ontology is ultimately interpreted as a possibilistic logic program). Arguments are then computed and accrued to compute the membership of instances to concepts using structured argumentation under Dung's grounded semantics. Our approach is qualitative providing a case study to show how our approach works, and a downloadable Java-based implementation for enacting our results.

As stated above, in this work, for reasoning with inconsistent ontologies instead of relying on the DeLP semantics (as in Gomez et al. did in [7, 8, 9]), we propose reasoning with arguments comparable by their relative weight via a grounded semantics along with argument accrual. See [7, 8, 9] for an account of related work concerning the topic of reasoning with inconsistent ontologies. Several authors approach the implementation of argumentation systems based on Dung's semantics (e.g. [10, 11, 12]; see Section 6 for a review of these works).

The rest of the paper is structured as follows. In Section 2, we recall the fundamentals of possibilistic description logic ontologies. In Section 3, we review how argumentation under Dung's semantics with arguments expressed in possibilistic logic programming is achieved and how possibilistic description logic ontologies can be interpreted as possibilistic logic programs. In Section 4, we introduce how to reason with inconsistent possibilistic ontologies using argument accrual. In Section 5, we discuss implementation details of the enactment of the proposed reasoning framework. In Section 6, we review related work. Finally, we conclude in Section 7.

Citation: S.A. Gómez. *"Reasoning with Inconsistent Possibilistic Ontologies by Applying Argument Accrual"*. Journal of Computer Science & Technology, vol. 17, no. 2, pp. 117–126, 2017.

Received: February 14, 2017. **Revised:** March 09, 2017. **Accepted:** May 17, 2017.

2 Possibilistic description logic ontologies

2.1 Classical description logics

Description Logics (DL) are a well-known family of knowledge representation formalisms [13]. In this work, we will consider a very tight subset $\mathscr{L}_{DL}$ of DLs, to which we will restrict our discussion, based on the notions of *concepts* (unary predicates, classes). Concept descriptions are built from concept names $C, D, \ldots$ using the constructors conjunction $(C \sqcap D)$, disjunction $(C \sqcup D)$ and negation $(\neg C)$. The empty concept is denoted by $\bot$. A DL ontology $\Sigma = (T, A)$ consists of two finite and mutually disjoint sets: a *Tbox* T which introduces the *terminology* and an *Abox* A (assertional box) which contains facts about particular objects in the application domain. The Tbox contains inclusion axioms $C \sqsubseteq D$, where C and D are (possibly complex) concept descriptions, meaning that every individual of C is also a D. Objects in the Abox are referred to by a finite number of *individual names* and these names may be used in assertional statements $a : C$ (meaning the individual a is a member of concept C).

2.2 Description logic programming

In this work we are only interested in the reasoning task known as *instance checking* that refers to determining if an individual a is a member of a concept C. Assigning semantics to a DL ontology can be done based on that DL is isomorphic with first-order logic restricted to two variables. Then, $C \sqsubseteq D$ can be interpreted as the formula $(\forall x)(c(x) \rightarrow d(x))$ and $a : C$ as $c(a)$. Description Logic Programming (DLP) approaches [14] take advantage of this to interpret those expressions as the Prolog rules "$\mathsf{d(X)}$:- $\mathsf{c(X)}$." and "$\mathsf{c(a)}$.", resp. Therefore instance checking of a is a member of C reduces to proving the goal "$\leftarrow \mathsf{c(a)}$".

2.3 Possibilistic description logics

We now recall the fundamentals of possibilistic description logic ontologies [15, 16]. A *possibilistic DL ontology* is a set of possibilistic axioms of the form $(\varphi, W(\varphi))$ where φ is an axiom expressed in $\mathscr{L}_{DL}$ and $W(\varphi) \in [0,1]$ is the degree of certainty (or priority) of φ. Namely, a possibilistic DL ontology Σ is such that $\Sigma = \{(\varphi_i, W(\varphi_i)) : i = 1, \ldots, n\}$. Only somewhat certain information is explicitly represented in a possibilistic ontology. That is, axioms with a null weight ($W(\varphi) = 0$) are not explicitly represented in the knowledge base. The weighted axiom $(\varphi, W(\varphi))$ means that the certainty degree of φ is at least equal to $W(\varphi)$. A possibilistic DL ontology Σ will also be represented by a pair $\Sigma = (T, A)$ where elements in both T and A may be uncertain. Note that if we consider all $W(\varphi_i) = 1$, then we find a classical DL ontology $\Sigma^* = \{\varphi_i : (\varphi_i, W(\varphi_i)) \in \Sigma\}$. We say that Σ is consistent if the classical ontology obtained from Σ by ignoring the weights associated with axioms is consistent, and inconsistent otherwise. Notice that the weights $W(\cdot)$ for axioms must be provided by the knowledge engineer that designs the knowledge base, thus providing the relative importance of axioms/rules and assertions/facts.

Example 1. *(Originally presented in [17].) Let* $\Sigma_1 = (T, A)$ *be the ontology modeling a variation of the famous Tweety example from the non-monotonic literature. It expresses that a bird usually flies, all penguins are birds, penguins do not usually fly, birds with broken wings normally do not fly either, and pilots can almost always fly. It is known that Tweety is a penguin with almost certainly a broken wing and most likely a pilot. Formally:*

$$T = \left\{ \begin{array}{l} (\mathsf{Bird} \sqsubseteq \mathsf{Flies}, 0.6), \\ (\mathsf{Penguin} \sqsubseteq \mathsf{Bird}, 1.0), \\ (\mathsf{Pilot} \sqsubseteq \mathsf{Flies}, 0.9), \\ (\mathsf{Penguin} \sqsubseteq \neg\mathsf{Flies}, 0.8), \\ (\mathsf{Bird} \sqcap \mathsf{BrokenWing} \sqsubseteq \neg\mathsf{Flies}, 0.7) \end{array} \right\}$$

$$A = \left\{ \begin{array}{l} (\mathsf{TWEETY} : \mathsf{BrokenWing}, 0.8), \\ (\mathsf{TWEETY} : \mathsf{Penguin}, 1.0), \\ (\mathsf{TWEETY} : \mathsf{Pilot}, 0.9) \end{array} \right\}$$

In Σ_1^**, because Tweety a penguin (and therefore a bird), he is both a member of* Flies *and* $\neg\mathsf{Flies}$*, meaning that he is a member of* $\bot$*. Traditional reasoners are not able to infer anything from such an inconsistent ontology, thus invalidating even reasoning with consistent subsets of the offending ontology.*

3 Dung-style ontology reasoning in possibilistic argumentation

We now recall how to reason with possibly inconsistent ontologies by using Dung-style argumentation, an approach we originally explored in [17].

3.1 Notions of possibilistic defeasible logic programming

The P-DeLP [18] language $\mathscr{L}$ is defined from a set of ground fuzzy atoms (fuzzy propositional variables) $p, q, \ldots$ along with the connectives $\sim$ (strong negation), $\wedge$ (written as a comma in Prolog clauses) and $\leftarrow$. A literal $L \in \mathscr{L}$ is a ground (fuzzy) atom $\sim q$, where q is a ground (fuzzy) propositional variable. A *rule* in $\mathscr{L}$ is a formula of the form $Q \leftarrow L_1 \wedge \ldots \wedge L_n$, where $Q, L_1, \ldots, L_n$ are literals in $\mathscr{L}$. When $n = 0$, the formula $Q \leftarrow$ is called a *fact*. The term *goal* will refer to any literal $Q \in \mathscr{L}$. Facts, rules and goals are the well-formed formulas in $\mathscr{L}$. A *certainty-weighted clause*, or simply weighted clause, is a pair (φ, α), where φ is a formula in $\mathscr{L}$ and $\alpha \in [0,1]$ expresses a lower bound for the certainty of φ in terms of a necessity measure.

The proof method for P-DeLP formulas, noted $\vdash$, is based on the generalized modus ponens rule, that from $(P \leftarrow Q_1, \ldots, Q_k, \gamma)$ and $(Q_1, \beta_1), \ldots, (Q_k, \beta_k)$ allows to infer $(P, \min(\gamma, \beta_1, \ldots, \beta_k))$. A clause (φ, α)

is referred to as certain when $\alpha = 1$ and uncertain otherwise. A set of clauses Γ is deemed as *contradictory*, denoted $\Gamma \vdash \bot$, when $\Gamma \vdash (Q, \alpha)$ and $\Gamma \vdash (\sim Q, \beta)$, with $\alpha > 0$ and $\beta > 0$, for some atom Q in $\mathscr{L}$. A program is a set of weighted rules and facts in $\mathscr{L}$ in which certain and uncertain information is distinguished. As an additional requirement, certain knowledge is required to be non-contradictory. A *P-DeLP program* $\mathscr{P}$ (or just *program* $\mathscr{P}$) is a pair (Π, Δ), where Π is a non-contradictory finite set of certain clauses, and Δ is a finite set of uncertain clauses. We build arguments $\mathscr{A}$ for Q with weight γ, noted as $\langle \mathscr{A}, Q, \gamma \rangle$, inductively: If (Q, γ) is a fact, then $\langle \{(Q, \gamma)\}, Q, \gamma \rangle$ is an argument; from $(P \leftarrow Q_1, \ldots, Q_n, \gamma)$ and n arguments $\langle \mathscr{A}_1, Q_1, \beta_1 \rangle$, $\ldots$, $\langle \mathscr{A}_n, Q_n, \beta_n \rangle$, when $Q_1, \ldots, Q_n, P$ is consistent, then we get $\langle \bigcup_{i=1}^{n} \mathscr{A}_i \cup \{(P \leftarrow Q_1, \ldots, Q_n, \gamma)\}, P, \min(\beta_1, \ldots, \beta_n, \gamma) \rangle$.

Conflict among arguments is formalized by the notions of counterargument and defeat. Let $\mathscr{P}$ be a program, and let $\langle \mathscr{A}_1, Q_1, \alpha_1 \rangle$ and $\langle \mathscr{A}_2, Q_2, \alpha_2 \rangle$ be two arguments in $\mathscr{P}$. We say that $\langle \mathscr{A}_1, Q_1, \alpha_1 \rangle$ *counterargues* $\langle \mathscr{A}_2, Q_2, \alpha_2 \rangle$ iff there exists a subargument (called *disagreement subargument*) $\langle \mathscr{S}, Q, \beta \rangle$ of $\langle \mathscr{A}_2, Q_2, \alpha_2 \rangle$ such that $\Pi \cup \{(Q_1, \alpha_1), (Q, \beta)\}$ is contradictory. The literal (Q, β) is called *disagreement literal*. Defeat among arguments involves the consideration of *preference criteria* defined on the set of arguments. The criterion applied here is based on necessity measures associated with arguments. Let $\mathscr{P}$ be a program, and let $\langle \mathscr{A}_1, Q_1, \alpha_1 \rangle$ and $\langle \mathscr{A}_2, Q_2, \alpha_2 \rangle$ be two arguments in $\mathscr{P}$. We will say that $\langle \mathscr{A}_1, Q_1, \alpha_1 \rangle$ is a *defeater* for $\langle \mathscr{A}_2, Q_2, \alpha_2 \rangle$ iff $\langle \mathscr{A}_1, Q_1, \alpha_1 \rangle$ counterargues argument $\langle \mathscr{A}_2, Q_2, \alpha_2 \rangle$ with disagreement subargument $\langle \mathscr{A}, Q, \alpha \rangle$, with $\alpha_1 \geq \alpha$. If $\alpha_1 > \alpha$ then $\langle \mathscr{A}_1, Q_1, \alpha_1 \rangle$ is called a *proper defeater*, otherwise ($\alpha_1 = \alpha$) it is called a *blocking defeater*. Notice that we digress from the original P-DeLP formalism of Chesñevar et al. [18] in that (i) we include facts in the support of arguments and (ii) facts are allowed to have a weight different than one (so allowing them to be considered as presumptions).

Example 2. *Consider again the ontology* Σ_1 *presented in Example 1. This ontology is interpreted as the equivalent possibilistic program* $\mathscr{P}_1$ *formed by the rules* $(brokenWing(tweety), 0.8)$, $(penguin(tweety), 1.0)$, $(pilot(tweety), 0.9)$, $(flies(X) \leftarrow bird(X), 0.6)$, $(bird(X) \leftarrow penguin(X), 1.0)$, $(\sim flies(X) \leftarrow penguin(X), 0.8)$, $(\sim flies(X) \leftarrow bird(X), brokenWing(X), 0.7)$ *and* $(flies(X) \leftarrow pilot(X), 0.9)$. *Exactly 8 arguments can be built from this program (notice that they coincide with the ones presented in Example 3):* $\langle \mathscr{A}_1, brokenWing(tweety), 0.8 \rangle$ *where* $\mathscr{A}_1 = \{(brokenWing(tweety), 0.8)\}$; $\langle \mathscr{A}_2, penguin(tweety), 1.0 \rangle$ *where* $\mathscr{A}_2 = \{(penguin(tweety), 1.0)\}$; $\langle \mathscr{A}_3, pilot(tweety), 0.9 \rangle$ *where* $\mathscr{A}_3 = \{(pilot(tweety), 0.9)\}$; $\langle \mathscr{A}_4, bird(tweety), 1.0 \rangle$ *where* $\mathscr{A}_4 = \{(bird(tweety) \leftarrow penguin(tweety), 1.0)\} \cup \mathscr{A}_2$; $\langle \mathscr{A}_5, \sim flies(tweety), 0.8 \rangle$ *where* $\mathscr{A}_5 = \{(\sim flies(tweety) \leftarrow penguin(tweety), 0.8)\} \cup \mathscr{A}_2$; $\langle \mathscr{A}_6, \sim flies(tweety), 0.7 \rangle$ *where* $\mathscr{A}_6 = \{(\sim flies(tweety) \leftarrow bird(tweety), brokenWing(tweety), 0.7)\} \cup \mathscr{A}_1 \cup \mathscr{A}_4$; $\langle \mathscr{A}_7, flies(tweety), 0.9 \rangle$ *where* $\mathscr{A}_7 = \{(flies(tweety) \leftarrow pilot(tweety), 0.9)\} \cup \mathscr{A}_3$, *and* $\langle \mathscr{A}_8, flies(tweety), 0.6 \rangle$ *where* $\mathscr{A}_8 = \{(flies(tweety) \leftarrow bird(tweety), 0.6)\} \cup \mathscr{A}_4$. *The reader should notice that the attacks among these arguments are exactly those presented in Fig. 1 and that these attacks are made into final conclusions (thus they are direct attacks). Nonetheless the reasoning framework presented here and the application we built also allow modeling attacks into premises (i.e. indirect attacks).*

3.2 Dung-style abstract argumentation

Abstract argumentation frameworks [19] do not presuppose any internal structure of arguments, thus considering only the interactions of arguments by means of an attack relation between arguments. An *abstract argumentation framework* $\mathscr{AF}$ is a pair $(Arg, \rightarrow)$ where Arg is a set of arguments and $\rightarrow$ is a relation of Arg into Arg. For two arguments $\mathscr{A}$ and $\mathscr{B}$ in Arg, the relation $\mathscr{A} \rightarrow \mathscr{B}$ means that the argument $\mathscr{A}$ attacks the argument $\mathscr{B}$. Abstract argumentation frameworks can be concisely represented by directed graphs, where arguments are represented as nodes and edges model the attack relation.

Example 3. *Consider the argumentation framework* $\mathscr{AF}_3 = (Arg, \rightarrow)$ *where* $Arg = \{\mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3, \mathscr{A}_4, \mathscr{A}_5, \mathscr{A}_6, \mathscr{A}_7, \mathscr{A}_8\}$ *and* $\rightarrow = \{(\mathscr{A}_5, \mathscr{A}_8), (\mathscr{A}_6, \mathscr{A}_8), (\mathscr{A}_7, \mathscr{A}_5), (\mathscr{A}_7, \mathscr{A}_6)\}$. *The framework is shown graphically in Fig. 1 and, although it is not necessary from a mathematical viewpoint, we can assign meaning to the above arguments to provide some intuition (notice that these arguments coincide with the ones in Example 2):* $\mathscr{A}_1$ *(for "Tweety has a broken wing");* $\mathscr{A}_2$ *(for the conclusion "Tweety is a penguin");* $\mathscr{A}_3$ *(supporting that "Tweety is a pilot");* $\mathscr{A}_4$ *(expressing that "Tweety is a bird");* $\mathscr{A}_5$ *(an argument for "Tweety does not fly because he is a penguin and penguins do not usually fly");* $\mathscr{A}_6$ *(that says that "Tweety does not fly because he has a broken wing");* $\mathscr{A}_7$ *(for "Tweety flies because he is also a pilot"), and finally* $\mathscr{A}_8$ *(for "Tweety flies because he is a bird and birds normally fly").*

Semantics are usually given to abstract argumentation frameworks by means of extensions. An extension E of an argumentation framework $\mathscr{AF} = (Arg, \rightarrow)$ is subset of Arg that gives some coherent view on the argumentation underlying $\mathscr{AF}$. In this work, we will reason under grounded semantics despite that other semantics have been proposed. Let $\mathscr{AF} = (Arg, \rightarrow)$ be an argumentation framework. An extension $E \subseteq Arg$ is *conflict-free* iff there are no $\mathscr{A}, \mathscr{B} \in E$ with $\mathscr{A} \rightarrow \mathscr{B}$.

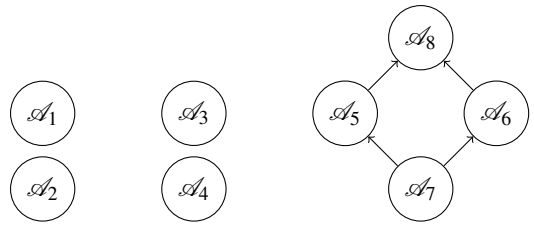

Figure 1: Abstract argumentation framework presented in Example 3

An argument $\mathscr{A} \in Arg$ is *acceptable* with respect to an extension $E \subseteq Arg$ iff for every $\mathscr{B} \in Arg$ with $\mathscr{B} \rightarrow \mathscr{A}$ there is a $\mathscr{A}' \in E$ with $\mathscr{A}' \rightarrow \mathscr{B}$. An extension $E \subseteq Arg$ is *admissible* iff it is conflict free and all $\mathscr{A} \in E$ are acceptable with respect to E. An extension $E \subseteq Arg$ is *complete* iff it is admissible and there is no $\mathscr{A} \in Arg \backslash E$ that is acceptable with respect to E. An extension $E \subseteq Arg$ is *grounded* iff it is complete and E is minimal with respect to set inclusion.

The intuition behind admissibility is that an argument can only be accepted if there are no attackers that are accepted and if an argument is not accepted then there has to be an acceptable argument attacking it. The idea behind the completeness property is that all acceptable arguments should be accepted. The grounded extension is the minimal set of acceptable arguments and uniquely determined. It can be computed as follows: first, all arguments that have no attackers are added to the empty extension E and those arguments and all arguments that are attacked by one of these arguments are removed from the framework; then the process is repeated; if one obtains a framework where there is no unattacked arguments, the remaining arguments are also removed.

Example 4. *Consider again the argumentation framework $\mathscr{AF}_3$ presented in Example 3. The grounded extension E of $\mathscr{AF}_3$ is given by $E = \{\mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3, \mathscr{A}_4, \mathscr{A}_7, \mathscr{A}_8\}$.*

3.3 Expressing possibilistic DL ontologies as possibilistic logic programs

Grosof et al. [14] provide a way of expressing a subset of Description Logic ontologies in logic programming, namely the description logic programming subset of DL that can be expressed as Horn knowledge bases. The idea consist of expressing both DL assertional statements and terminological axioms as equivalent Horn clauses. We will explain only the part of the algorithm relevant to this work.

Given an ontology $\Sigma = (T, A)$, for every terminological axiom or assertional statement $(\varphi, W(\phi))$ we will generate a possibilistic axiom $(\mathscr{T}(\varphi), W(\varphi))$, where $\mathscr{T}(\cdot)$ is the transformation function from the language of description logics to the language of Horn clauses. The specification of the $\mathscr{T}$ is as follows: Assertional statements in A of the form $a : C$ are expressed as facts $c(a)$. We obtain an equivalent ontology composed only of inclusion axioms of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq D$ which are expressed as Horn clauses of the form $d(X) \leftarrow c_1(X), \ldots, c_n(X)$. Given a possibilistic DL ontology Σ, Σ is expressed as an equivalent P-DeLP program $\mathscr{P}$. With this program, a grounded extension E will be computed. If $(c(a), \alpha)$ belongs to E then we will say that the individual A is a member of the concept C with certainty degree α.

Example 5. *Recall from Example 4 that $E = \{\mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3, \mathscr{A}_4, \mathscr{A}_7, \mathscr{A}_8\}$. Therefore we can affirm that, from $\mathscr{A}_1$,* TWEETY *is a member of* BrokenWing *with certainty degree* 0.8*; from $\mathscr{A}_2$,* TWEETY *is a member of* Penguin *with certainty degree* 1.0*; from $\mathscr{A}_3$,* TWEETY *is a member of* Pilot *with certainty degree* 0.9*; from $\mathscr{A}_4$,* TWEETY *is a member of* Bird *with certainty degree* 1.0*; from $\mathscr{A}_7$,* TWEETY *is a member of* Flies *with certainty degree* 0.9*, and from $\mathscr{A}_8$,* TWEETY *is a member of* Flies *with certainty degree* 0.6.

From $\mathscr{A}_7$ and $\mathscr{A}_8$, as Tweety is both a member of Flies with both degrees 0.6 and 0.9. In the next section, we will see how to reconcile these two degrees into only one by using argument accrual.

4 Ontology reasoning with argument accrual

Now we deal with the problem of accruing arguments for justifying the membership of individuals to concepts, thus redefining the task of instance checking by means of argument accrual. Our approach relies in previous work of Gómez Lucero et al. by presenting a variation of their approach that we apply to ontology reasoning. Gómez Lucero et al. [3] presented an approach to model accrual of arguments in a possibilistic setting where, given different arguments supporting the same conclusion, they are able to accumulate their strength in terms of possibilistic values. For this, they define the notion of *accrued structure* whose necessity degree is computed in terms of two mutually recursive functions: $f^{+}_{\Phi}(\cdot)$ (the accruing function) and $f^{MP}_{\Phi}(\cdot)$ (that propagates necessity degrees). The latter is parameterized w.r.t. a user-defined function *ACC* that supports *non-depreciation* (i.e. accruing arguments results in a necessity degree no lower than any single argument involved in the accrual) and *maximality* (i.e. accrual means total certainty only if there is an argument with necessity degree 1). We recall the notion of *argument accrual* as interpreted by [3]:

Definition 1. *Let $\mathscr{P}$ be a P-DeLP program and let Ω be a set of arguments in $\mathscr{P}$ supporting the same conclusion H, i.e. $\Omega = \{\langle \mathscr{A}_1, H, \alpha_1 \rangle, \ldots, \langle \mathscr{A}_n, H, \alpha_n \rangle\}$. The accrued structure $\mathscr{AS}_H$ for H is a 3-uple $[\Phi, H, \alpha]$, where $\Phi = \mathscr{A}_1 \cup \ldots \cup \mathscr{A}_n$ and α is obtained as follows. Let Q be a literal in Φ and let*

$(\varphi_1,\beta_1),\ldots,(\varphi_n,\beta_n)$ *be all the weighted clauses in* Φ *with head* Q*, then*

$$f^{+}_{\Phi}(Q) = ACC(f^{MP}_{\Phi}(\varphi_1),\ldots,f^{MP}_{\Phi}(\varphi_n)).$$

Let (φ,β) *be a weighted clause in* Φ*, whenever* φ *is a fact* Q *then* $f^{MP}_{\Phi}(\varphi)=\beta$ *but if* $\varphi = Q \leftarrow P_1,\ldots,P_n$ *then* $f^{MP}_{\Phi}(\varphi) = \min(f^{+}_{\Phi}(P_1),\ldots,f^{+}_{\Phi}(P_n))$*. And ACC is the one-complement accrual:* $ACC(\alpha_1,\ldots,\alpha_n) = 1-\prod_{i=1}^{n}(1-\alpha_i)$.

Given $[\Phi,H,\alpha]$ and $[\Theta,K,\gamma]$, $[\Theta,K,\gamma]$ is an accrued substructure if $\Theta \subseteq \Phi$. Also $[\Theta,K,\gamma]$ is a complete accrued substructure of $[\Phi,H,\alpha]$ iff for any other accrued substructure $[\Theta',K,\gamma']$ of $[\Phi,H,\alpha]$ it holds that $\Theta' \subset \Theta$. We say $[\Psi,K,\beta]$ attacks $[\Phi,H,\alpha]$ at literal H' iff there is a complete accrued substructure $[\Phi',H',\alpha']$ of $[\Phi,H,\alpha]$ such that $K=\overline{H'}$ and $\beta > \alpha'$ ($\bar{\cdot}$ stands for the complement operator where $\overline{P}$ is $\sim P$ and $\overline{\sim P}$ is P).

Definition 2. *Let* $\mathscr{P}$ *be a possibilistic logic program. Let* $Accruals(\mathscr{P})$ *be the set of complete accrued structures of* $\mathscr{P}$*. Let* $\mathscr{ASF}(\mathscr{P}) = (Accruals(\mathscr{P}), attacks)$ *be the argumentation framework induced by the accruals of* $\mathscr{P}$ *where* $attacks \subseteq Accruals(\mathscr{P}) \times Accruals(\mathscr{P})$*. The extension of* $\mathscr{P}$ *is defined as the grounded extension of* $\mathscr{ASF}(\mathscr{P})$ *where attacks stands for the attack relation between complete accrued structures.*

Notice that the notions of both attack and valid conclusions of the system presented here differ from those of [3], thus leading to a different behavior.

Example 6. *Consider again the ontology* Σ_1 *and its interpretation as the possibilistic program* $\mathscr{P}_1$*. The following complete accrued structures can be computed from* $\mathscr{P}_1$*:* $\mathscr{AS}_1 = [\mathscr{A}_2, penguin(tweety), 1.0]$, $\mathscr{AS}_2 = [\mathscr{A}_7 \cup \mathscr{A}_8, flies(tweety), 0.96]$, $\mathscr{AS}_3 = [\mathscr{A}_3, pilot(tweety), 0.9]$, $\mathscr{AS}_4 = [\mathscr{A}_1, brokenWing(tweety), 0.8]$, $\mathscr{AS}_5 = [\mathscr{A}_5 \cup \mathscr{A}_6, \sim flies(tweety), 0.94]$ *and* $\mathscr{AS}_6 = [\mathscr{A}_4, bird(tweety), 1.0]$*. We show* $\mathscr{AS}_2$ *and* $\mathscr{AS}_5$ *in Fig. 2. In this case* $\mathscr{AS}_2$ *attacks* $\mathscr{AS}_5$ *at the literal* $\sim flies(tweety)$ *(see Fig. 3). Therefore the grounded extension of* $\mathscr{ASF}(\mathscr{P}_1)$ *is* $\{\mathscr{AS}_1, \mathscr{AS}_2, \mathscr{AS}_3, \mathscr{AS}_4, \mathscr{AS}_6\}$.

Definition 3. *Given a possibilistic ontology* Σ*, a concept* C *and an individual* a*, we say that* a *is member of* C *with certainty degree* α *iff there is a complete accrued structure for* $[\Phi,c(a),\alpha]$ *in the grounded extension of* $\mathscr{ASF}(\mathscr{T}(\Sigma))$.

Example 7. *Consider again the ontology* Σ_1*,* $\mathscr{T}(\Sigma_1)$ *is the program* $\mathscr{P}_1$*. The complete accrued structures from this program are those presented in Example 6. As* $\mathscr{AS}_2 = [\mathscr{A}_7 \cup \mathscr{A}_8, flies(tweety), 0.96]$ *belongs to the grounded extension of accrued structures of* $\mathscr{P}_1$*, then we conclude* TWEETY *is a member of* Flies *with certainty degree 0.96.*

Property 1. *When each accrued structure is formed by exactly one argument, the grounded extension of the argumentation framework induced by the program coincides with the argumentation framework induced from the accrued program.*

Proof: Let Σ be a possibilistic ontology and $\mathscr{P}$ be $\mathscr{T}(\Sigma)$. Suppose that there is only one argument $\mathscr{A}$ in $\mathscr{P}$ with certainty degree α supporting H. Then the accrued structure $\mathscr{AS}$ for H is formed only by $\mathscr{A}$. Because of how $f^{+}_{\Phi}(H)$ is defined, the certainty degree of $\mathscr{AS}$ is also α. Then the graph of the argumentation framework formed by arguments is identical to the argumentation framework formed by accrued structures. Therefore their grounded extensions coincide.

5 Enactment of the proposed approach

In order to enact the approach proposed in this work, we developed a Java-based implementation that extends the one already presented in [17]. This application can be downloaded from the author's institutional site at `http://cs.uns.edu.ar/~sag/engine-v2/`. The implementation also provides the functionality of loading several examples, editing them, obtaining the equivalent P-DeLP program of a given weighted ontology. In order to give a text representation for possibilistic DL ontologies, in [17] we proposed a RACER-like syntax based on the representation language for ontologies that allows a knowledge engineer to add labels for specifying the weight of each inclusion axiom. The system also allows for computing grounded extensions and also provides a graphical representation of the argumentation framework, individual arguments and accrued structures based on the D3.JS and Dracula Javascript libraries. We show an snapshot of the reasoner's user interface in Fig. 6. With this, we were able to replicate the accrued structures presented by [3]. Based on the experience gained by testing examples, we argue that this approach presents a more complex way to compute the vertexes of the argumentation framework but the resulting graph is smaller, thus leading to a much simpler reasoning framework. We explore the details of our approach in the rest of this section.

5.1 Scripting languages for possibilistic ontologies and programs

In order to give a text representation for possibilistic DL ontologies, we propose a LISP-like syntax based on the representation language for ontologies of the RACER reasoner. Our proposal permits a knowledge engineer to add labels for specifying the weight of each axiom. Notice as the system presented in this work is a running prototype, we have only implemented the constructors for declaring the signature of the ontology (viz. the `(signature :atomic-concepts` lst_1

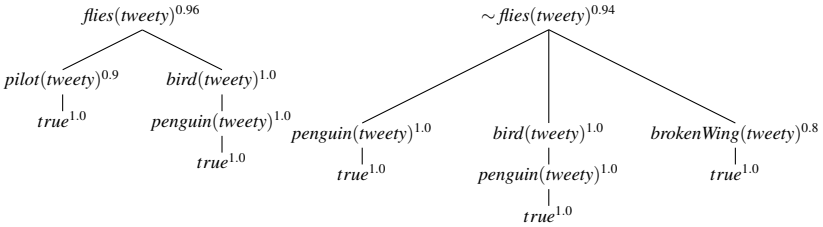

Figure 2: Accrued structures $\mathscr{AS}_2$ and $\mathscr{AS}_5$ from $\mathscr{P}_1$ (Notice that the leftmost branch of $\mathscr{AS}_5$ stands for a sub-structure for $\sim flies(tweety)$ based on the argument $\mathscr{A}_5$ while the two rightmost branches for another based on the argument $\mathscr{A}_6$.)

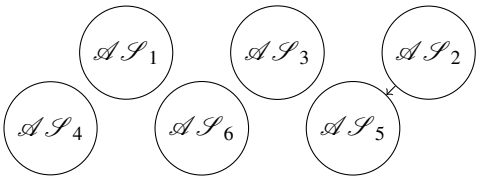

Figure 3: Abstract argumentation framework $\mathscr{ASF}(\mathscr{P}_1)$ of Example 6

```
(signature
:atomic-concepts (bird penguin flies
       broken_wing pilot)
:individuals (tweety)
)

(instance tweety bird 1.0)
(instance tweety broken_wing 0.8)
(instance tweety penguin 1.0)
(instance tweety pilot 0.9)

(implies bird flies 0.6)
(implies penguin bird 1.0)
(implies penguin (not flies) 0.8)
(implies (and bird broken_wing) (not flies) 0.7)
(implies pilot flies 0.9)
```

Figure 4: Racer script for the Tweety ontology

`:individuals` lst_2`)` element expressing the list lst_1 of atomic concepts and the list lst_2 of names of individuals), the assertional class statements (viz. the `(instance` a C α`)` element asserting that an individual a is a member of a class C with a certain degree α), and the inclusion axioms (viz. the `(implies` C D α`)` element expressing that a concept C is a subconcept of another concept D with a certain degree α). Besides, only the operators for the complement of a concept (viz. `not`) and conjunction of concepts (viz. `and`) are supported. In Fig. 4, we present the Racer-like script for the ontology of Example 1.

Our approach to provide a text representation for possibilistic programs is simple and and follows guidelines of DeLP and ASPIC. Facts of the form $(p(a), \alpha)$ are codified as "`p(a) <- true` α" and rules of the form $(p(X) \leftarrow q_1(X), \ldots, q_n(X), \alpha)$ are written as "`p(X) <- q1(X), ..., qn(X)` α". Besides, strong negation of atoms `p(X)` is represented in the usual way as `~p(X)`. As an example, in Fig. 5, we present the PDeLP-like script for the program of Ex. 2.

```
broken_wing(tweety) <- true 0.8
penguin(tweety) <- true 1.0
pilot(tweety) <- true 0.9
flies(X) <- bird(X) 0.6
bird(X) <- penguin(X) 1.0
~flies(X) <- penguin(X) 0.8
~flies(X) <- bird(X), broken_wing(X) 0.7
flies(X) <- pilot(X) 0.9
```

Figure 5: P-DeLP script for the Tweety ontology

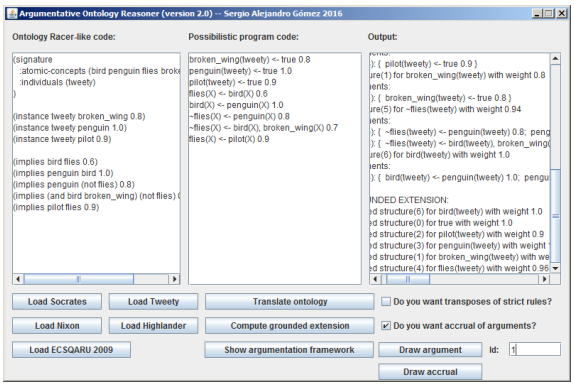

Figure 6: Snapshot of the GUI

5.2 User interface

In order to recreate our experiences, the user can download a zip file containing an executable Java JAR file containing the engine for reasoning with potentially inconsistent possibilistic ontologies. Notice that this is a rudimentary prototype and the parser is unforgiving to errors if it notices them at all. So, the user can edit and modify the preloaded examples but has to be sure that their syntax is correct before executing the reasoning engine. Notice also that Racer-like axioms and possibilistic logic programming rules have to be "one-liners" (i.e. they cannot be split in two or more lines).

The program requires Java running environment version 1.7 or higher installed in the user's computer. The user interface has three text areas: leftmost (labeled as "Ontology Racer-like code"), middle (labeled as "Possibilistic program code") and rightmost (labeled as "Output").

In the leftmost area, the user can preload (or type or paste) a possibilistic ontology. The program offers four preloaded examples, that can be loaded by pressing

```
(signature
       :atomic-concepts (man mortal)
       :individuals (socrates)
)
(instance socrates man 1.0)
(implies man mortal 1.0)
```

Figure 7: Racer script for the Socrates ontology

```
(signature
       :atomic-concepts (pacifist republican quaker)
       :individuals (nixon)
)
(instance nixon republican 1.0)
(instance nixon quaker 1.0)
(implies republican (not pacifist) 0.5)
(implies quaker pacifist 0.5)
```

Figure 8: Racer script for the Nixon's Diamond ontology

one the four buttons "Load Socrates", "Load Nixon", "Load Highlander", "Load Tweety" and "Load ECSQARU 2009" whose meaning is explained below. In every case, the user would be able to automatically translate the ontologies into PDeLP, compute the grounded extensions (with or without accrual of arguments), and see graphically and textually the composition of arguments, accrued structures and argumentation graphs.

The "Load Socrates" button corresponds to the classical Prolog language example: *Socrates is a man, and all men are mortal.* (See Fig. 7.)

The "Load Nixon" button corresponds to the classical Nixon Diamond example: *Nixon is both a Quaker and a Republican. Republicans are not pacifists. And Quakers are pacifists.* (See Fig. 8.)

The "Load Highlander" button corresponds to a modelization of the Highlander the Series universe where mortal and immortal men coexist: *Joe is a man. Duncan is both a man and a Highlander. Men are mortal. Highlanders are immortals.* (See Fig. 9.)

The "Load Tweety" button corresponds to Σ_1 in Fig. 4. Finally, the "Load ECSQARU 2009" button loads an ontology that codifies the running example of Gmez Lucero et al. [3] (see Fig. 10); our prototype is able to find the same accrued structures that they report.

In the middle area, by pressing the "Translate ontology" button, the user can see the ontology interpreted as a possibilistic logic program (in the middle text area). In the rightmost area, the user will see the arguments belonging to the grounded extension of the possibilistic program (for this you first have to press the "Compute grounded extension" button). If the user wants to consider transposes of strict rules (i.e. rules with weight one), he has be sure to check "Want transposes of strict rules". The most important information regarding how the program computes its answer will be shown in the console nonetheless. See the part of the output for the Tweety example in Fig. 11.

The contents of the leftmost and middle text areas can be edited but being sure that the syntax is correct before executing the engine. If the user wants to reason by using accrual of arguments, he must check the checkbox: "Do you want accrual?". In this version, the user can graphically see the graph of the argumentation system (either with accrual or without it). To see a particular argument, he has to specify the argument ID and press the button labeled "Draw Argument". We show how the application displays both the tree for the argument $\mathscr{A}_6$ from Example 2 in Fig. 12 and the accrued structure $\mathscr{AS}_4$ from Example 6 in Fig. 13.

The graph corresponding to the argumentation framework can be seen by pressing the button labeled

```
(signature
       :atomic-concepts (highlander man mortal)
       :individuals (duncan joe)
)
(instance joe man 1.0)
(instance duncan man 1.0)
(instance duncan highlander 1.0)
(implies man mortal 0.6)
(implies (and man highlander) (not mortal) 0.8)
```

Figure 9: Racer script for the Highlander The Series ontology

```
(signature
:atomic-concepts (p q s t u v w x y z)
:individuals (a)
)

(instance a p 1.0)
(instance a q 1.0)
(instance a t 1.0)
(instance a u 1.0)
(instance a v 1.0)
(instance a w 1.0)

(implies z x 0.7)
(implies v z 0.5)
(implies u y 0.3)
(implies p s 0.7)
(implies y x 1.0)
(implies w (not z) 0.4)
(implies p (not y) 0.4)
(implies t (not s) 0.9)
(implies t z 0.6)
(implies s (not z) 0.8)
(implies q (not x) 0.45)
```

Figure 10: Racer script for the ontology codifying the example of Gómez Lucero et al.

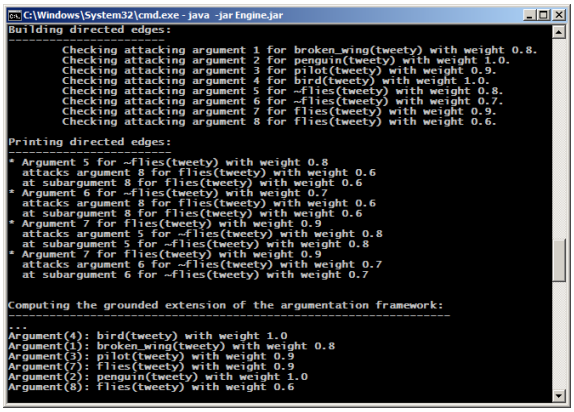

Figure 11: Output of the engine in the console

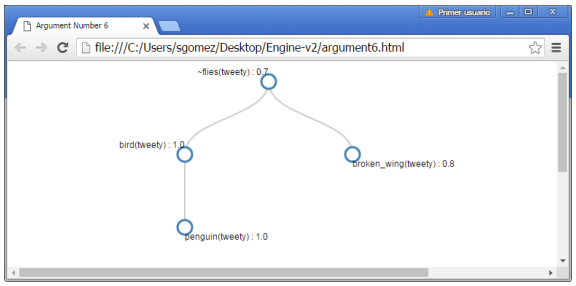

Figure 12: Argument $\mathscr{A}_6$ as displayed by the reasoner

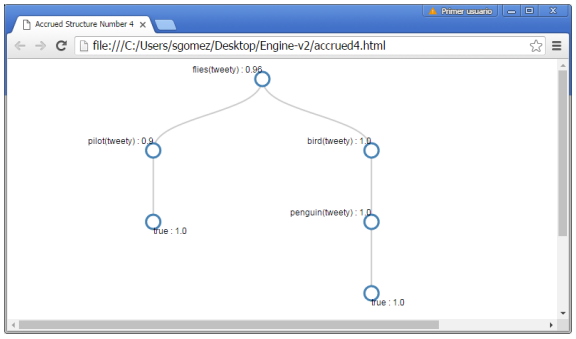

Figure 13: Accrued structure $\mathscr{AS}_4$ as shown by the reasoner

as "Show framework". If the user is not considering accrual of arguments, he will see a graph as shown in Fig. 14. Instead, if the user considers reasoning with accrual of arguments, then the system will output a graph like the one in Fig. 15. Notice that in both cases the initial graph can be customized by dragging its vertexes and edges.

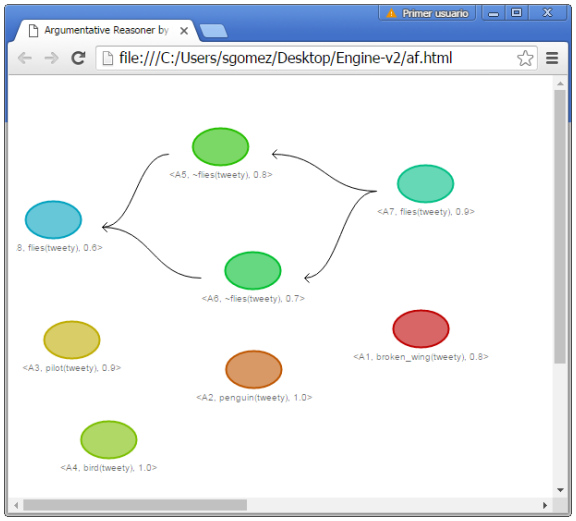

Figure 14: Framework of plain arguments as seen in the reasoner

Figure 15: Framework of accrued structures as seen in the reasoner

6 Related work

In previous works [7, 8, 9], Gómez et al. reviewed related work concerning the topic of reasoning with inconsistent ontologies. In this section, we will focus exclusively on reviewing implementations of argumentation systems based on Dung's semantics. During the last years, some approaches based on interpreting ontologies as logic programs have arisen [14] which are collectively known as Description Logic Programming. Gómez et al. [7, 9] have exploited this to reason on possibly inconsistent ontologies into Defeasible Logic Programming (DeLP). In this work, instead of relying on the DeLP semantics, we propose reasoning with arguments comparable by their relative weight via a grounded semantics along with argument accrual.

Bryant et al. [10] discuss their current work on a prototype light-weight Java-based argumentation engine that can be used to implement a non-monotonic reasoning component in Internet or agent-based applications. Likewise, our system is a Java-based implementation of an argumentation engine based on the representation language of the P-DeLP but interpreting P-DeLP with a Dung-style semantics. However, our engine is not based on the Prolog engine even when the knowledge representation language is somewhat similar and is able to handle argument accrual.

Snaith and Reed [11] present TOAST, a system that implements the ASPIC+ framework and accepts a knowledge base and rule set with associated preference and contrariness information, returning both textual and visual commentaries on the acceptability of arguments in the derived abstract framework. The system can be used as both a web front-end and a web service. Our implementation only provides a graphical user interface that can be run a stand-alone JAR file

requiring only to have installed a suitable Java runtime environment. TOAST can compute grounded, preferred and complete extensions but our system only computes grounded extensions.

Tamani and Croitoru [12] introduce a quantitative preference based argumentation system relying on ASPIC argumentation framework and fuzzy set theory. The knowledge base is fuzzified to allow the experts to express their expertise (premises and rules) attached with grades of importance in the unit interval. Arguments are attached with a score aggregating the importance expressed on their premises and rules. Extensions are then computed and the strength of each of which can also be obtained based on its strong arguments. The strengths are used to rank fuzzy extensions from the strongest to the weakest one, upon which decisions can be made. Likewise, our approach allows to express the importance of rules using numbers in the unit interval following the path marked by P-DeLP.

7 Conclusions and Future Work

We have presented an approach for performing instance checking in inconsistent possibilistic description logic ontologies based on argument accrual, leading to a more concise reasoning framework when inconsistency is present but several arguments for instance checking may coexist. Our approach involves translating ontologies into the language of possibilistic logic programming and grouping arguments for the same conclusion via argument accrual. We have developed a Java-based implementation that allows the user to input an ontology and select a reasoning mechanism using argument accrual under a grounded semantics based on Dung-style argumentation.

Our current approach has several limitations and addressing them is part of our future research work. For instance, when computing the argument base of the argumentation systems, as arguments are built bottom-up from facts by applying the rules obtained from the Tbox in a forward-chain fashion, the existence of recursion in concept definitions may produce an infinite base of arguments, thus producing an infinite loop in the execution of the reasoning engine. To grasp this, consider, for example, the existence of an Abox assertion $(a : P, 1)$ and a Tbox inclusion axiom of the form $(P \sqsubseteq P, 0.7)$, this situation generates an infinite number of arguments for $p(a)$ with weight 0.7 (this of course could be avoided with a simple check but it is not currently implemented; notice however that systems with an infinite number of arguments are of research interest at least from a theoretical point of view). Besides, in our implementation, we only implemented direct attack between pairs of accrued structures; implementing the full approach of [3] (in which accruals are weakened when attackers are discovered) in the context of Dung-based structured argumentation requires further research. All things considered, our approach leads to a much cleaner reasoning framework when several arguments supporting the membership of a certain individual to a certain class coexist; nonetheless, allowing a progressive weakening of accrued structures would lead to a much more flexible reasoning framework to model the dynamics of reasoning with ontologies. Additionally, our current implementation only deals with grounded semantics, so another improvement of our proposal involves the implementation of other argumentation extension-based semantics for having additional strategies for computing the justification states of arguments and accrued structures (e.g. stable, preferred, stage, semi-stable, ideal, CF2, or prudent semantics).

Acknowledgements

This research is funded by Secretaría General de Ciencia y Técnica, Universidad Nacional del Sur, Argentina and by Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC-PBA), Argentina.

Competing interests

The author has declared that no competing interests exist.

References

[1] X. Zhang, G. Xiao, Z. Lin, and J. V. den Bussche, "Inconsistency-tolerant reasoning with OWL-DL," *International Journal of Approximate Reasoning*, vol. 55, pp. 557–584, 2014.

[2] T. J. M. Bench-Capon and P. E. Dunne, "Argumentation in artificial intelligence," *Artificial Intelligence*, vol. 171, no. 10-15, pp. 619–641, 2007.

[3] M. G. Lucero, C. I. Chesñevar, and G. R. Simari, "Modelling Argument Accrual in Possibilistic Defeasible Logic Programming," in *ECSQARU 2009, LNAI 5590* (C. S. ad G. Chemello, ed.), pp. 131–143, 2009.

[4] S. A. Gómez, "On the Application of Argument Accrual to Reasoning with Inconsistent Possibilistic Ontologies," in *Proc. of the XXII Argentinian Conference of Computer Science (CACIC 2016)*, pp. 14–23, Universidad Nacional de San Luis, oct 2016.

[5] B. Verheij, *Rules, Reasons, Arguments: Formal studies of argumentation and defeat*. PhD thesis, University of Maastricht, 1996.

[6] I. Letia and A. Groza, "Modelling Imprecise Arguments in Description Logics," *Advances in Electrical and Computer Engineering*, vol. 9, no. 3, pp. 94–99, 2009.

[7] S. A. Gómez, C. I. Chesñevar, and G. R. Simari, "Reasoning with Inconsistent Ontologies Through Argumentation," *Applied Artificial Intelligence*, vol. 1, no. 24, pp. 102–148, 2010.

[8] S. A. Gómez and G. R. Simari, "Merging of ontologies using belief revision and defeasible logic programming," *Inteligencia Artificial*, vol. 16, no. 52, pp. 16–28, 2013.

[9] S. A. Gómez, C. I. Chesñevar, and G. R. Simari, "ONTOarg: A Decision Support Framework for Ontology Integration based on Argumentation," *Expert Systems with Applications*, vol. 40, pp. 1858–1870, 2013.

[10] D. Bryant, P. J. Krause, and G. Vreeswijk, "Argue tuProlog: A Lightweight Argumentation Engine for Agent Applications," in *Computational Models of Argument: Proceedings of COMMA 2006*, 2006.

[11] M. Snaith and C. Reed, "TOAST: online ASPIC+ implementation," in *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, IOS Press, 2012.

[12] N. Tamani and M. Croitoru, "Fuzzy argumentation system for decision support," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, vol. 442, pp. 77–86, Communications in Computer and Information Science, 2014.

[13] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds., *The Description Logic Handbook – Theory, Implementation and Applications*. Cambridge University Press, 2003.

[14] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, "Description Logic Programs: Combining Logic Programs with Description Logics," *WWW2003, May 20-24, Budapest, Hungary*, 2003.

[15] S. Benferhat, Z. Bouraoui, S. Lagrue, and J. Rossit, "Merging Inconmensurable Possibilistic DL-Lite Assertional Bases," in *Proceedings of the IJCAI Workshop 13 Ontologies and Logic Programming for Query Answering* (O. Papini, S. Benferhat, L. Garcia, and M.-L. Mugnier, eds.), pp. 90–95, 2015.

[16] S. A. Gómez, C. I. Chesñevar, and G. R. Simari, "Using Possibilistic Defeasible Logic Programming for Reasoning with Inconsistent Ontologies," in *Computer Science & Technology Series. XVII Argentine Congress of Computer Science Selected Papers* (A. D. Giusti and J. Diaz, eds.), pp. 19–29, 2012.

[17] S. A. Gómez, "Towards a practical implementation of a reasoner for inconsistent possibilistic description logic programming ontologies," in *Proc. of the 2nd Argentinian Symposium of Ontologies and their Applications (SAOA 2016)*, pp. 1–14, SADIO–45 JAIIO, sep 2016.

[18] T. Alsinet, C. I. Chesñevar, and L. Godo, "A level-based approach to computing warranted arguments in possibilistic defeasible logic programming," in *COMMA* (P. Besnard, S. Doutre, and A. Hunter, eds.), vol. 172 of *Frontiers in Artificial Intelligence and Applications*, pp. 1–12, IOS Press, 2008.

[19] P. M. Dung, "On the aceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming," in *Proceedings of the 13th International Joint Conference in Artificial Intelligence (IJCAI)*, pp. 852–857, 1993.