Reasoning with Inconsistent Ontologies in Possibilistic Defeasible Logic Programming

Sergio A. Gómez[†], Carlos I. Chesñevar[†‡], and Guillermo R. 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,cic,grs}@cs.uns.edu.ar
[‡]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina

Abstract. We present a preliminary framework for reasoning with possibly inconsistent Description Logic ontologies in Possibilistic Defeasible Logic Programming. A case study is presented where we show how the proposed approach works. The proposal presented is apt for being used in the context of Semantic Web ontologies as it can be applied to the Web Ontology Language OWL.

1 Introduction

The *Semantic Web* (SW) [1] is a vision of the Web where resources have precise meaning defined in terms of ontologies. The Web Ontology Language (OWL) [2] whose semantics is based on *Description Logics* [3] is the *de facto* standard for the SW. Agents in the SW are supposed to reason over web resources by using standard reasoning systems, thus being able to compute an implicit hierarchy of concepts defined in an ontology and then checking the membership of individuals to those concepts. Over the last few years an alternative approach to reasoning with ontologies called *Description Logic Programming* (DLP) [4] has gained interest. The DLP approach relies on translating DL ontologies into the language of logic programming, so standard Prolog environments can be used to reason on them.

A possible anomaly in ontologies is *inconsistency*. An inconsistency is characterized by a logical contradiction. Inconsistencies in ontologies prevent standard reasoners from obtaining conclusions. Normally, this situation renders an ontology useless unless the knowledge engineer *debugs* it (*i.e.*, repairs the ontology for making it consistent). As the knowledge engineer could not be available, alternative approaches to automatically repairing the ontology consist of using Belief Revision [5] techniques to either extract a maximally consistent subset of the ontology or discard a minimally inconsistent subset of the ontology. Yet another approach consists of using a non-standard reasoning mechanism for accepting inconsistency and just *dealing* with it (for instance using Paraconsistent

Logics [6]). In this line of work, Gómez *et al.* [7] have applied defeasible argumentation (in particular *Defeasible Logic Programming* [8]) to reason with possibly inconsistent ontologies.

Possibilistic Defeasible Logic Programming (P-DeLP) [9] is a logic programming language which combines features from argumentation theory and logic programming, incorporating as well the treatment of possibilistic uncertainty and fuzzy knowledge at object-language level. In this article, we show a preliminary approach to reason with possibly inconsistent DL ontologies in P-DeLP. For this we define the concept of weighted DL ontology which is an ontology whose axioms have given numerical weights indicating their degree of certainty, then the ontology can be interpreted as a P-DeLP program.

Outline: In Section 2, we present the fundamentals of Description Logics. Section 3 reviews the fundamentals of Possibilistic Defeasible Logic Programming. In Section 4, we introduce a framework for reasoning with possibly inconsistent weighted ontologies in P-DeLP. Finally, Section 5 concludes.

2 Fundamentals of Description Logics

Description Logics (DL) [3] are a family of knowledge representation formalisms based on the notions of *concepts* (unary predicates, classes) and *roles* (binary relations) that allow to build complex concepts and roles from atomic ones. Let C, D stand for concepts, R for a role and a, b for individuals. Concept descriptions are built from concept names using the constructors conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), complement ($\neg C$), existential restriction ($\exists R.C$), and value restriction ($\forall R.C$). To define the semantics of concept descriptions, concepts are interpreted as subsets of a domain of interest, and roles as binary relations over this domain. Further extensions are possible including inverse (P^-) and transitive (P^+) roles. A DL ontology consists of two finite and mutually disjoint sets: a *Tbox* which introduces the *terminology* and an *Abox* which contains facts about particular objects in the application domain. Tbox statements have the form $C \sqsubseteq D$ (*inclusions*) and $C \equiv D$ (*equalities*), where C and D are (possibly complex) concept descriptions. Objects in the Abox are referred to by a finite number of *individual names* and these names may be used in two types of assertional statements: *concept assertions* of the type $a : C$ and *role assertions* of the type $\langle a, b\rangle : R$.

A knowledge representation system based on DL is able to perform specific kinds of reasoning, its purpose goes beyond storing concept definitions and assertions. As a DL ontology has a semantics that makes it equivalent to a set of axioms of first-order logic, it contains implicit knowledge that can be made explicit through inferences. Inferences in DL systems are usually divided into Tbox reasoning and Abox reasoning. In this paper we are concerned only with Abox reasoning, more precisely with *instance checking* [3]. Instance checking consists of determining if an assertion is entailed from an Abox. For instance, $T \cup A \models a : C$ indicates that the individual a is a member of the concept C w.r.t. the Abox A and the Tbox T.

3 Possibilistic Defeasible Logic Programming

The P-DeLP [9] language $\mathcal{L}$ is defined from a set of ground fuzzy atoms (fuzzy propositional variables) $\{p, q, \ldots\}$ together with the conectives $\{\sim, \wedge, \leftarrow\}$. The symbol $\sim$ stands for *negation*. A literal $L \in \mathcal{L}$ is a ground (fuzzy) atom $\sim q$, where q is a ground (fuzzy) propositional variable. A *rule* in $\mathcal{L}$ 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 $\mathcal{L}$. When $n = 0$, the formula $Q \leftarrow$ is called a *fact*. The term *goal* will refer to any literal $Q \in \mathcal{L}$. Facts, rules and goals are the well-formed formulas in $\mathcal{L}$.

Definition 1 (Certainty-weighted clause). *A certainty-weighted clause, or simply weighted clause, is a pair (φ, α), where φ is a wff in $\mathcal{L}$ and $\alpha \in [0, 1]$ expresses a lower bound for the certainty of φ in terms of a necessity measure.*

The original P-DeLP language is based on Possibilistic Gödel Logic or PGL, which is able to model both uncertainty and fuzziness and allows for a partial matching mechanism between fuzzy propositional variables. For simplicity, Chesñevar *et al.* [9] restrict themselves to the fragment of P-DeLP built on non-fuzzy propositions, and hence based on the necessity-valued classical propositional Possibilistic logic. As a consequence, possibilistic models are defined by possibility distributions on the set of classical interpretations and the proof method for P-DeLP formulas, written $\vdash$, is defined based on the generalized modus ponens rule:

$$\frac{(L_0 \leftarrow L_1 \wedge \ldots \wedge L_k, \gamma) \quad (L_1, \beta_1), \ldots, (L_k, \beta_k)}{(L_0, \min(\gamma, \beta_1, \ldots, \beta_k))}$$

which is a particular instance of the possibilistic resolution rule, and which provides the *non-fuzzy* fragment of P-DeLP with a complete calculus for determining the maximum degree of possibilistic entailment for weighted literals.

In P-DeLP *certain* and *uncertain* clauses can be distinguished. A clause (φ, α) is referred as certain if $\alpha = 1$ and uncertain otherwise. A set of clauses Γ is deemed as *contradictory*, denoted $\Gamma \vdash \bot$, whenever $\Gamma \vdash (q, \alpha)$ $\Gamma \vdash (\sim q, \beta)$, with $\alpha > 0$ and $\beta > 0$, for some atom in $\mathcal{L}$. A P-DeLP program is a set of weighted rules and facts in $\mathcal{L}$ in which certain and uncertain information is distinguished. As an additional requirement, certain knowledge is required to be non-contradictory. Formally:

Definition 2 (Program). *A P-DeLP program $\mathcal{P}$ (or just program $\mathcal{P}$) 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.*

Definition 3 (Argument. Subargument). *Given a program $\mathcal{P} = (\Pi, \Delta)$, a set $\mathcal{A} \subseteq \Delta$ of uncertain clauses is an argument for a goal Q with necessity degree $\alpha > 0$, denoted $\langle A, Q, \alpha \rangle$, iff: (i) $\Pi \cup \mathcal{A} \vdash (Q, \alpha)$; (ii) $\Pi \cup \mathcal{A}$ is non-contradictory, and (iii) there is no $\mathcal{A}_1 \subset \mathcal{A}$ such that $\Pi \cup \mathcal{A}_1 \vdash (Q, \beta)$, $\beta > 0$. Let $\langle \mathcal{A}, Q, \alpha \rangle$ and $\langle \mathcal{S}, R, \beta \rangle$ be two arguments, $\langle \mathcal{S}, R, \beta \rangle$ is a subargument of $\langle \mathcal{A}, Q, \alpha \rangle$ iff $\mathcal{S} \subseteq \mathcal{A}$.*

Conflict among arguments is formalized by the notions of counterargument and defeat.

Definition 4 (Counterargument). *Let $\mathcal{P}$ be a program, and let $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ and $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ be two arguments in $\mathcal{P}$. We say that $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ counterargues $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ iff there exists a subargument (called disagreement subargument) $\langle \mathcal{S}, Q, \beta \rangle$ of $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ such that $\Pi \cup \{(Q_1, \alpha_1), (Q, \beta)\}$ is contradictory. The literal (Q, β) is called disagreement literal.*

Defeat among arguments involves a *preference criterion* on conflicting arguments, defined on the basis of necessity measures associated with arguments.

Definition 5 (Defeat). *Let $\mathcal{P}$ be a program, and let $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ and $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ be two arguments in $\mathcal{P}$. We will say that $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ is a* defeater *for $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ iff $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ counterargues argument $\langle \mathcal{A}_2, Q_2, \alpha_2 \rangle$ with disagreement subargument $\langle \mathcal{A}, Q, \alpha \rangle$, with $\alpha_1 \geq \alpha$. If $\alpha_1 > \alpha$ then $\langle \mathcal{A}_1, Q_1, \alpha_1 \rangle$ is called a* proper defeater*, otherwise ($\alpha_1 = \alpha$) it is called a* blocking defeater*.*

Definition 6 (Argumentation line). *An* argumentation line λ *starting in an argument $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ is a finite sequence of arguments $[\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle, \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, \alpha_n \rangle, \ldots]$ such that every $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ defeats $\langle \mathcal{A}_{i-1}, Q_{i-1}, \alpha_{i-1} \rangle$, for $0 < i \leq n$, satisfying certain* dialectical constraints *(see below). Every argument $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ has* level i*. We will distinguish the sets*

$$S^k_\lambda = \bigcup_{i=0,2,\ldots,2\lfloor k/2 \rfloor} \{\langle \mathcal{A}_i, Q_i, \alpha_i \rangle \in \lambda\} \text{ and } I^k_\lambda = \bigcup_{i=1,3,\ldots,2\lfloor k/2 \rfloor + 1)} \{\langle \mathcal{A}_i, Q_i, \alpha_i \rangle \in \lambda\}$$

associated with even-level (resp. odd-level) arguments in λ up to the k-th level ($k \leq n$).

An argumentation line can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). In order to avoid *fallacious* reasoning, argumentation theory imposes additional constraints on such an argument exchange to be considered rationally acceptable w.r.t. a P-DeLP program $\mathcal{P}$, namely:

1. *Non-contradiction:* given an argumentation line λ of length n the set S^n_λ associated with the proponent (resp. I^n_λ for the opponent) should be *non-contradictory* w.r.t. $\mathcal{P}$.
2. *No circular argumentation:* no argument $\langle \mathcal{A}_j, Q_j, \alpha_j \rangle$ in λ is a sub-argument of an argument $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ, $i < j$.
3. *Progressive argumentation:* every blocking defeater $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle$ in λ is defeated by a proper defeater $\langle \mathcal{A}_{i+1}, Q_{i+1}, \alpha_{i+1} \rangle$ in λ.

To determine whether a given argument is ultimately undefeated (or *warranted*) w.r.t. a program $\mathcal{P}$, the P-DeLP framework relies on an exhaustive dialectical analysis. Such analysis is modeled in terms of a *dialectical tree*:

Definition 7 (Dialectical tree). *Let $\mathcal{P}$ be a program, and let $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ be an argument w.r.t. $\mathcal{P}$. A* dialectical tree *for $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$, is a tree structure defined as follows:*

1. *The root node of* $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ *is* $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$.
2. $\langle \mathcal{B}', H', \beta' \rangle$ *is an immediate child of* $\langle \mathcal{B}, H, \beta \rangle$ *iff there exists an argumentation line* $\lambda = [\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle, \langle \mathcal{A}_1, Q_1, \alpha_1 \rangle, \ldots, \langle \mathcal{A}_n, Q_n, \alpha_n \rangle, \ldots]$ *such that there are two elements* $\langle \mathcal{A}_{i+1}, Q_{i+1}, \alpha_{i+1} \rangle = \langle \mathcal{B}', H', \beta' \rangle$ *and* $\langle \mathcal{A}_i, Q_i, \alpha_i \rangle = \langle \mathcal{B}, H, \beta \rangle$, *for some* $i = 0, \ldots, n-1$.

Nodes in a dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ can be marked as *undefeated* and *defeated* nodes (U-nodes and D-nodes, resp.). A dialectical tree will be marked as an AND-OR tree: all leaves in $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ will be marked U-nodes (as they have no defeaters), and every inner node is to be marked as *D-node* iff it has at least one U-node as a child, and as *U-node* otherwise.

Definition 8 (Warrant). *An argument* $\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle$ *is ultimately accepted as valid (or* warranted*) with a necessity degree* α_0 *w.r.t. a program* $\mathcal{P}$ *iff the root of the tree* $\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}$ *is marked as* U-node *(*i.e.*,* $Mark(\mathcal{T}_{\langle \mathcal{A}_0, Q_0, \alpha_0 \rangle}) = U$*).*

4 Reasoning with DL Ontologies as P-DeLP Programs

In the presence of inconsistency, traditional DL reasoners issue an error message and stop further processing of ontologies. Thus the burden of repairing the ontology is on the knowledge engineer. However, the knowledge engineer is not always available and in some cases, such as when dealing with imported ontologies, he has neither the authority nor the expertise to correct the source of inconsistency. Therefore, we are interested in coping with inconsistencies such that the task of dealing with them is automatically solved by the reasoning system. We propose performing such a task by translating DL ontologies into P-DeLP programs. By doing so we gain the capability of reasoning with inconsistent ontologies. However we also lose some expressiveness in the involved ontologies. As Def. 11 shows, certain restrictions will have to be imposed on DL ontologies in order to be expressed in the P-DeLP language.

Our proposal is based in part in the work of [4] who show that the processing of ontologies can be improved by the use of techniques from the area of logic programming. In particular they have identified a subset of DL languages that can be effectively mapped into a Horn-clause logics.

Definition 9 (Weighted axiom. Weighted assertion). *Let* C, D *stand for concept names,* P, Q *for role names, and* a, b *for individual names. Let* $\mathcal{A}$ *be an axiom of the form* $C \sqsubseteq D$, $C \equiv D$, $\top \sqsubseteq \forall P.D$, $\top \sqsubseteq \forall P^-.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^-$, *or* $P \sqsubseteq P^+$. *Let* ω *be a real number such that* $0 \leq \omega \leq 1$. *A* weighted axiom *is pair* $(\mathcal{A}, \omega)$. *Let* $\mathcal{B}$ *be an assertion of the form* $a : C$ *or* $\langle a, b \rangle : P$. *A* weighted assertion *is a pair* $(\mathcal{B}, \omega)$.

Definition 10 (Weighted ontology). *Let* T *be a set of weighted axioms and* A *be a set of weighted assertions. A weighted ontology* Σ *is a pair* (T, A). *The set* T *is called* weighted terminology *(or just* weighted Tbox*) and* A *is called* weighted assertional box *(or* weighted Abox *for short).*

Example 1. In Fig. 1, we present a weighted ontology $\Sigma = (T, A)$ based on the fictional universe of the *Highlander* movies. The meaning of weighted terminology T is as follows: Axiom (1) says that a man is apparently a ——mortal; axiom (2) expresses that men from the Highlands that keep their heads on are supposed to be immortals; axiom (3) implies that every beheaded man does not keep his head on, and (4) says that immortals that still are known to be *in the game* have their heads on. The weighted assertional box A expresses that Joe, Duncan and Connor are men; Duncan and Connor are Highlanders, Connor has been beheaded, and it is known that Connor and Duncan have been in the game.

Tbox T**:**
(1) (Man $\sqsubseteq$ Mortal, 0.6)
(2) (Man $\sqcap$ Highlander $\sqcap$ Keeps_head $\sqsubseteq \neg$Mortal, 0.8)
(3) (Beheaded $\sqsubseteq \neg$Keeps_head, 1)
(4) (In_The_Game $\sqsubseteq$ Keeps_head, 0.9)

Abox A**:**
(5) (JOE : Man, 1)
(6) (DUNCAN : Man, 1)
(7) (CONNOR : Man, 1)
(8) (DUNCAN : Highlander, 1)
(9) (CONNOR : Highlander, 1)
(10) (CONNOR : Beheaded, 1)
(11) (DUNCAN : In_The_Game, 1)
(12) (CONNOR : In_The_Game, 1)

Fig. 1. A weighted ontology $\Sigma = (T, A)$

As noted by Grosof *et al.* [4], for DL sentences to be mapped into Horn-logic rules, they must satisfy certain constraints. Conjunction and universal restrictions appearing in the right-hand side of inclusion axioms can be mapped to heads of rules (called $\mathcal{L}_h$-classes). In contrast, conjunction, disjunction and existential restriction can be mapped to rule bodies whenever they occur in the left-hand side of inclusion axioms (called $\mathcal{L}_b$-classes). As equality axioms "$C \equiv D$" are interpreted as two inclusion axioms "$C \sqsubseteq D$" and "$D \sqsubseteq C$", they must belong to the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$ (called $\mathcal{L}_{hb}$ classes).

Definition 11 ($\mathcal{L}_h$, $\mathcal{L}_b$ and $\mathcal{L}_{hb}$ classes (adapted from [4])). *Let A be an atomic class name, C and D class expressions, and R a property. In the $\mathcal{L}_h$ language, $C \sqcap D$ is a class, and $\forall R.C$ is also a class. Class expressions in $\mathcal{L}_h$ are called* $\mathcal{L}_h$-classes*. In the $\mathcal{L}_b$ language, $C \sqcup D$ is a class, and $\exists R.C$ is a class too. Class expressions in $\mathcal{L}_b$ are called* $\mathcal{L}_b$-classes*. The $\mathcal{L}_{hb}$ language is defined as the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$. Class expressions in $\mathcal{L}_{hb}$ are called* $\mathcal{L}_{hb}$-classes*.*

Definition 12. ($\mathcal{T}$ mapping from DL sentences to logic programming rules (adapted from [4])) *Let A, C, D be concepts, X, Y variables, P, Q properties. The $\mathcal{T}$ mapping from the language of DL to the language of P-DeLP is defined in Fig. 2. Besides, intermediate transformations of the form "$(H_1 \wedge H_2) \leftarrow B$" will be rewritten as two rules "$H_1 \leftarrow B$" and "$H_2 \leftarrow B$". Similarly transformations of the form "$H_1 \leftarrow H_2 \leftarrow B$" will be rewritten as "$H_1 \leftarrow B \wedge H_2$", and rules of the form "$H \leftarrow (B_1 \vee B_2)$" will be rewritten as two rules "$H \leftarrow B_1$" and "$H \leftarrow B_2$".*

$$
\begin{array}{rl}
\mathcal{T}(C \sqsubseteq D) =_{df} & T_h(D,x) \leftarrow T_b(C,x), \text{if } C \text{ is an } \mathcal{L}_b\text{-class and } D \text{ an } \mathcal{L}_h\text{-class} \\
\mathcal{T}(C \equiv D) =_{df} & \left\{ \begin{array}{l} \mathcal{T}(C \sqsubseteq D) \\ \mathcal{T}(D \sqsubseteq C) \end{array} \right. , \text{if } C \text{ and } D \text{ are } \mathcal{L}_{hb}\text{-classes} \\
\mathcal{T}(\top \sqsubseteq \forall P.D) =_{df} & T_h(D,y) \leftarrow P(x,y), \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}(\top \sqsubseteq \forall P^{-}.D) =_{df} & T_h(D,x) \leftarrow P(x,y), \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}(a : D) =_{df} & T_h(D,a), \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}(\langle a,b \rangle : P) =_{df} & P(a,b) \\
\mathcal{T}(P \sqsubseteq Q) =_{df} & Q(x,y) \leftarrow P(x,y) \\
\mathcal{T}(P \equiv Q) =_{df} & \left\{ \begin{array}{l} Q(x,y) \leftarrow P(x,y) \\ P(x,y) \leftarrow Q(x,y) \end{array} \right. \\
\mathcal{T}(P \equiv Q^{-}) =_{df} & \left\{ \begin{array}{l} Q(x,y) \leftarrow P(y,x) \\ P(y,x) \leftarrow Q(x,y) \end{array} \right. \\
\mathcal{T}(P^{+} \sqsubseteq P) =_{df} & P(x,z) \leftarrow P(x,y) \wedge P(y,z) \\
\textbf{where:} & \\
T_h(A,x) =_{df} & A(x) \\
T_h((C \sqcap D),x) =_{df} & T_h(C,x) \wedge T_h(D,x) \\
T_h((\forall R.C),x) =_{df} & T_h(C,y) \leftarrow R(x,y) \\
T_b(A,x) =_{df} & A(x) \\
T_b((C \sqcap D),x) =_{df} & T_b(C,x) \wedge T_b(D,x) \\
T_b((C \sqcup D),x) =_{df} & T_b(C,x) \vee T_b(D,x) \\
T_b((\exists R.C),x) =_{df} & R(x,y) \wedge T_b(C,y)
\end{array}
$$

Fig. 2. Mapping from DL axioms to logic programming rules

Definition 13 (Interpretation of a weighted ontology). *Let $\Sigma = (T, A)$ be a weighted ontology such that $T = \{(s_1, \alpha_1), \ldots, (s_n, \alpha_n)\}$ and $A = \{(a_1, \beta_1), \ldots, (a_m, \beta_m)\}$, then:*

$$\mathfrak{Trad}(T) = \{(\mathcal{T}(s_1), \alpha_1), \ldots, (\mathcal{T}(s_n), \alpha_n)\}$$
$$\mathfrak{Trad}(A) = \{(\mathcal{T}(a_1), \beta_1), \ldots, (\mathcal{T}(a_m), \beta_m)\}.$$

Besides, if the cardinal k of $\mathcal{T}(s_i) = \{f_1, \ldots, f_k\}$ is greater than 1, then the translation of (s_i, α_i) is $(f_i, \alpha_i), \ldots, (f_k, \alpha_i)$. Let $\mathfrak{Prop}(\cdot)$ be the propositionalization operator for a first-order theory (each propositional term "p(a)" generated from a predicate "p(x)" and a constant "a" will be noted as "p_a"). The interpretation *of Σ, noted as $\mathfrak{Interpretation}(\Sigma)$, is the P-DeLP program $\mathcal{P} = (\mathfrak{Prop}(\mathfrak{Trad}(T)), \mathfrak{Prop}(\mathfrak{Trad}(A))$.*

Example 2. Consider again the weighted ontology $\Sigma = (T, A)$ presented in Ex. 1. In Fig. 3 we present a logical program $\mathfrak{Trad}(T) \cup \mathfrak{Trad}(A)$. And in Fig. 4, we present the P-DeLP program $\mathcal{P} = \mathfrak{Interpretation}(\Sigma)$. Notice that as there are three constants (*viz.*, *joe*, *duncan* and *connor*), and three first-order rules (*viz.*, (1)–(4)), twelve rules are generated in the propositional program, *i.e.* four for every instantiation of each rule with each one of the three constants.

Definition 14 (Instance checking). *Let $\Sigma = (T, A)$ be a weighted ontology. Let C be a concept name, a an individual name. Let ϵ be real number such that $0 \leq \epsilon \leq 1$. The individual a is a member of the concept C with strength ϵ iff there is a warranted argument $\langle \mathcal{A}, C(a), \epsilon \rangle$ w.r.t. $\mathfrak{Interpretation}(\Sigma)$.*

Example 3. Consider the program $\mathcal{P}$ in Ex. 4 that corresponds to the interpretation of the ontology Σ from Ex. 1. We will show how the operation of

instance checking works in P-DeLP for deciding the membership of the individuals Joe, Duncan and Connor to the concept Mortal. First, consider the case of Joe: An argument $\langle \mathcal{A}, mortal\_joe, 0.6 \rangle$ can be obtained, where $\mathcal{A} = \{(mortal\_joe \leftarrow man\_joe, 0.6)\}$. This argument has no defeaters and is thus warranted, therefore we conclude that JOE is a member of the concept Man with strength 0.6 (see Fig. 5.(a)). Second, consider the case of Duncan: As in the case of Joe, there is an argument $\langle \mathcal{B}_1, mortal\_duncan, 0.6 \rangle$, with $\mathcal{B}_1 = \{(mortal\_duncan \leftarrow man\_duncan, 0.6)\}$. But this argument is defeated by $\langle \mathcal{B}_2, \sim mortal\_duncan, 0.8 \rangle$, where

$$\begin{aligned} \mathcal{B}_2 = \{ & (\sim mortal\_duncan \leftarrow man\_duncan \wedge \\ & highlander\_duncan \wedge keeps\_head\_duncan, 0.8), \\ & (keeps\_head\_duncan \leftarrow in\_the\_game\_duncan, 0.9)\}. \end{aligned}$$

As this argument $\mathcal{B}_2$ is undefeated, we reach the conclusion that DUNCAN is a member of the concept $\neg$Mortal with strength 0.8 (see Fig. 5.(b)). Last, consider the case of Connor: Yet again there is an argument expressing that Connor is mortal since he is a man: $\langle \mathcal{C}_1, mortal\_connor, 0.6 \rangle$, with

$$\mathcal{C}_1 = \{(mortal\_connor \leftarrow man\_connor, 0.6)\};$$

as in the case of Duncan, this argument is defeated by another that says that Connor in immortal because he is a Highlander, *i.e.* $\langle \mathcal{C}_2, \sim mortal\_connor, 0.8 \rangle$ where

$$\begin{aligned} \mathcal{C}_2 = \{ & (\sim mortal\_connor \leftarrow man\_connor \wedge \\ & highlander\_connor \wedge keeps\_head\_connor, 0.8), \\ & (keeps\_head\_connor \leftarrow in\_the\_game\_connor, 0.9)\}. \end{aligned}$$

However, in this case there is another (undefeated) argument $\mathcal{C}_3$ that defeats $\mathcal{C}_2$, namely $\langle \mathcal{C}_3, \sim keeps\_head\_connor, 1 \rangle$, where

$$\mathcal{C}_3 = \{(\sim keeps\_head\_connor \leftarrow beheaded\_connor, 1)\}.$$

In this way, the argument $\mathcal{C}_1$ gets undefeated again, and we conclude that CONNOR is a member of the concept Mortal with strength 0.6 (see Fig. 5.(c)).

5 Conclusions and Future Work

We have presented a preliminary framework for reasoning with inconsistent ontologies by using the Possibilistic Defeasible Logic Programming machinery. The proposed approach allows for determining the degree of tentativeness for the membership of an individual to a class in the potential presence of inconsistency w.r.t. a Description Logic ontology. Axioms and assertions in an ontology are qualified with degrees of certainty which are used to determine the degree of

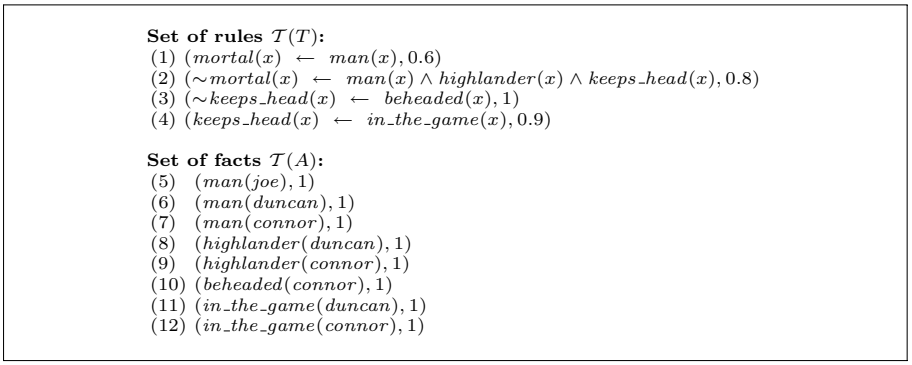

Fig. 3. First-order P-DeLP program that interprets ontology $\Sigma = (T, A)$

membership of individuals to concepts. This approach continues previous work of ours [7] that translates DL ontologies into Defeasible Logic Programming and uses generalized specificity to compare arguments. One advantage of the approach presented in this paper is that it allows to better characterize the preference criterion between axioms and thus arguments that are built when considering them, since the comparison criterion is no longer syntactically determined by the form of the underlying program that represents the ontology. One drawback of our approach is that the propositionalization of a first-order theory produces lots of facts that are irrelevant (notice that given a knowledge base with p k-ary predicates and n constants, there are possible pn^k instantiations, see Ex. 2), impacting the computational efficiency of the method. As part of our current research work, we are interested in applying this proposal to ontology integration and studying what the intrinsic logical properties of the approach are.

Acknowledgments: This research is funded by Projects PIP 112-200801-02798 (CONICET, Argentina), PGI 24/ZN10, PGI 24/N006 (SGCyT, UNS, Argentina) and Universidad Nacional del Sur.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
2. McGuiness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview (2004) http://www.w3.org/TR/owl-features/.
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook – Theory, Implementation and Applications. Cambridge University Press (2003)
4. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logics. WWW2003, May 20-24, Budapest, Hungary (2003)

Set of propositional rules that interprets the Tbox T:
(1.j) $(mortal\_joe) \leftarrow man\_joe, 0.6)$
(2.j) $(\sim mortal\_joe \leftarrow man\_joe \wedge highlander\_joe \wedge keeps\_head\_joe, 0.8)$
(3.j) $(\sim keeps\_head\_joe \leftarrow beheaded\_joe, 1)$
(4.j) $(keeps\_head\_joe \leftarrow in\_the\_game\_joe, 0.9)$
(1.d) $(mortal\_duncan \leftarrow man\_duncan, 0.6)$
(2.d) $(\sim mortal\_duncan \leftarrow man\_duncan \wedge highlander\_duncan \wedge keeps\_head\_duncan, 0.8)$
(3.d) $(\sim keeps\_head\_duncan \leftarrow beheaded\_duncan, 1)$
(4.d) $(keeps\_head\_duncan \leftarrow in\_the\_game\_duncan, 0.9)$
(1.c) $(mortal\_connor \leftarrow man\_connor, 0.6)$
(2.c) $(\sim mortal\_connor \leftarrow man\_connor \wedge highlander\_connor \wedge keeps\_head\_connor, 0.8)$
(3.c) $(\sim keeps\_head\_connor \leftarrow beheaded\_connor, 1)$
(4.c) $(keeps\_head\_connor \leftarrow in\_the\_game\_connor, 0.9)$

Set of propositions that interprets the Abox A:
(5) $(man\_joe, 1)$
(6) $(man\_duncan, 1)$
(7) $(man\_connor, 1)$
(8) $(highlander\_duncan, 1)$
(9) $(highlander\_connor, 1)$
(10) $(beheaded\_connor, 1)$
(11) $(in\_the\_game\_duncan, 1)$
(12) $(in\_the\_game\_connor, 1)$

Fig. 4. Propositionalization of the program presented in Fig. 3

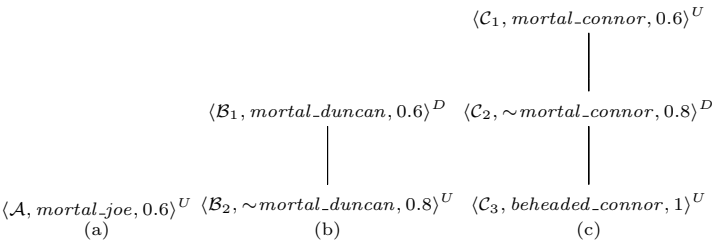

Fig. 5. Dialectical trees for *mortal_joe*, *mortal_duncan* and *mortal_connor*

5. Ribeiro, M.M., Wassermann, R.: Base Revision for Ontology Debugging. J. Log. Comput. **19**(5) (2009) 721–743
6. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with Inconsistent Ontologies. In Kaelbling, L.P., Saffiotti, A., eds.: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), Edinburgh, Scotland (August 2005) 454–459
7. Gómez, S.A., Chesñevar, C.I., Simari, G.R.: Reasoning with Inconsistent Ontologies Through Argumentation. Applied Artificial Intelligence **24**(1) (2010) 102–148
8. García, A., Simari, G.: Defeasible Logic Programming an Argumentative Approach. Theory and Practice of Logic Programmming **4**(1) (2004) 95–138
9. Alsinet, T., Chesñevar, C.I., Godo, L.: A level-based approach to computing warranted arguments in possibilistic defeasible logic programming. In Besnard, P., Doutre, S., Hunter, A., eds.: COMMA. Volume 172 of Frontiers in Artificial Intelligence and Applications., IOS Press (2008) 1–12