Expert Systems with Applications 40 (2013) 1858–1870

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

ONTOarg: A decision support framework for ontology integration based on argumentation

Sergio Alejandro Gómez *, Carlos Iván Chesñevar, Guillermo Ricardo Simari

Department of Computer Science and Engineering, Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca, Argentina
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina

ARTICLE INFO

Keywords:
Ontology integration
Description logics
Inconsistency handling
Defeasible argumentation
Defeasible logic programming

ABSTRACT

Description Logic Programming (DLP) is a well-known approach to reason with Description Logic (DL) ontologies, translating them into the language of logic programming (LP). Even though DLP offers several advantages in terms of efficiency and reuse of existing logic programming tools (such as Prolog environments), a major hindrance of this approach is its limitation for reasoning in the presence of inconsistent ontologies. Recent research has led to the use of defeasible argumentation to model different DL reasoning capabilities when handling inconsistent ontologies, resulting in so-called δ-ontologies. In this article we present ONTOarg, a decision support framework for performing local-as-view integration of possibly inconsistent and incomplete ontologies in terms of Defeasible Logic Programming (DeLP). We show how to reason on Description Logics (DL) ontologies in an ontology integration system by performing a dialectical analysis in order to determine the membership of individuals to concepts. We present formal definitions of a framework for ontology integration of DL ontologies based on DeLP along with a case study and review some of the properties of the approach.

1. Introduction

The *Semantic Web* (Berners-Lee, Hendler, & Lassila, 2001) (SW) is a vision of the Web where agents can reason about resources whose meaning is assigned in terms of ontologies (Gruber, 1993). Within the Semantic Web, the OWL language has become the current standard for defining ontologies and its underlying semantics is based on *Description Logics* (DL) (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003), for which specialized reasoners exist (such as Racer (Haarslev & Möller, 2001) and Pellet (Parsia & Sirin, 2004)). In this context, *Description Logic Programming* (DLP) provides a practical approach to reason with DL ontologies, translating them into the language of logic programming (LP) (Grosof, Horrocks, Volz, & Decker, 2003). Although DLP offers several advantages in terms of efficiency and reuse of existing LP tools (such as Prolog environments), DLP is incapable of reasoning in the presence of *inconsistent ontologies*. Dealing with inconsistencies and potentially contradictory information is a common issue in semantic web reasoning. The semantic integration of potentially inconsistent information sources in the SW is complicated, as the knowledge engineer usually has no authority to correct foreign ontologies. Inconsistencies can also arise whenever the domains modeled are inherently inconsistent. Additionally, in many situations resources and their data are modeled in terms of ontologies whose terms can differ, so that the ontologies must be aligned to put their terms into mutual agreement (Klein, 2001).

Argumentation provides a sophisticated mechanism for the formalization of commonsense reasoning, which has found application and proven its importance in different areas of Artificial Intelligence (AI) such as legal systems, multi-agent systems, and decision support systems among others (see e.g. Bench-Capon & Dunne, 2007; Janjua & Hussain, 2012; Modgil et al., 2013; Rahwan & Simari, 2009). Intuitively, an argument can be thought of as a coherent set of statements that supports a claim. The ultimate acceptance of an argument will depend on a dialectical analysis of arguments in favor and against the claim (Rahwan & Simari, 2009). In the last decade, several frameworks for formalizing argumentative reasoning have been developed (also called generically *Argumentation Systems*), providing different knowledge representation and inference capabilities.

Recent research has led to the use of defeasible argumentation to model different DL reasoning capabilities when handling inconsistent ontologies, resulting in so-called *δ-ontologies* (Gómez, Chesñevar, & Simari, 2010). These special ontologies are based on *Defeasible Logic Programming* (DeLP) (García & Simari, 2004), a defeasible argumentation framework based on logic programming. In contrast with other approaches, δ-ontologies have the flexibility of allowing to assess defeasibly the membership of an individual to a concept description in the presence of a potential inconsistent

* Corresponding author at: Department of Computer Science and Engineering, Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca, Argentina.
E-mail addresses: sag@cs.uns.edu.ar (S.A. Gómez), cic@cs.uns.edu.ar (C.I. Chesñevar), grs@cs.uns.edu.ar (G.R. Simari).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.10.025

ontology. However, this approach was intended for a single ontology, and fell short when dealing with different ontologies which have to be integrated in a single one. There are two kinds of ontology integration approaches, viz. global-as-view (GAV) and local-as-view (LAV) (Calvanese, Giacomo, & Lenzerini, 2001). When performing LAV integration, concepts of the local ontologies are mapped to queries over a global ontology.

In this article, we present ONTOarg, a decision support framework for modeling LAV ontology integration when the involved ontologies can be potentially inconsistent. The ontologies are expressed in the language of DL but their semantics is expressed in terms of DeLP. The alignments between the local and global ontologies are expressed as DL inclusion axioms that are also interpreted as DeLP sentences. As the ontologies are potentially inconsistent, a dialectical analysis is performed on the interpretation of both the ontologies and the mappings from the local to the global ontology. ONTOarg can be seen as a front-end expert system for the ontology engineers, which solves automatically the problem of alignments using an argumentative proof procedure.

The rest of this paper is structured as follows. In Section 2 we present the fundamentals of Description Logics and Defeasible Logic Programming. Section 3 recalls the δ-ontologies framework for reasoning with possibly inconsistent ontologies. In Section 4, we extend δ-ontologies for performing local-as-view integration of possibly inconsistent ontologies. Section 5 discusses some properties of the proposed approach. In Section 6 we compare our approach with related work. Finally Section 7 concludes the paper.

2. Knowledge representation and reasoning in the semantic web

2.1. Description logics

Description Logics (DL) (Baader et al., 2003) 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 $\langle a,b\rangle{:}R$.

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. An interpretation $I = \langle \Delta^I, \cdot^I \rangle$ consists of a non-empty set Δ^I (the domain of I) and a function $\cdot^I$ (the interpretation function of I) which maps every concept name A to a subset A^I of Δ^I, and every role name R to a subset R^I of $\Delta^I \times \Delta^I$. The interpretation function is extended to arbitrary concept descriptions as follows: $(\neg C)^I = \Delta^I \setminus C^I$; $(C \sqcup D)^I = C^I \cup D^I$; $(C \sqcap D)^I = C^I \cap D^I$; $(\exists R.C)^I = \{x | \exists y \text{ s.t. } (x,y) \in R^I \text{ and } y \in C^I\}$, and $(\forall R.C)^I = \{x | \forall y, (x,y) \in R^I \text{ implies } y \in C^I\}$.

The semantics of Tbox statements is as follows. An interpretation I satisfies $C \sqsubseteq D$ iff $C^I \subseteq D^I$, I satisfies $C \equiv D$ iff $C^I = D^I$. An interpretation I *satisfies* the assertion $a{:}C$ iff $a^I \in C^I$, and it *satisfies* $\langle a,b\rangle{:}R$ iff $(a^I,b^I) \in R^I$. An interpretation I is a *model* of a DL (Tbox or Abox) statement ϕ iff it satisfies the statement, and is a model of a DL ontology Σ iff it satisfies every statement in Σ. A DL ontology Σ *entails* a DL statement ϕ, written as $\Sigma \models \phi$, iff every model of Σ is a model of ϕ.

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* (Baader et al., 2003). 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.

Two anomalies in ontologies are incoherence and inconsistency (Baader et al., 2003). A concept is *unsatisfiable* in a Tbox if its interpretation is empty in any interpretation of the Tbox. A Tbox is *incoherent* if there exists an unsatisfiable concept. An ontology is *incoherent* if its Tbox is incoherent. An ontology is *inconsistent* if there exists no models. In the following example we present an inconsistent ontology.

Example 1. Let $\Sigma_1 = (T,A)$ be an ontology, where:

$$T = \left\{ \begin{array}{l} \mathsf{Penguin} \sqsubseteq \mathsf{Bird} \\ \mathsf{Bird} \sqsubseteq \mathsf{Fly} \\ \mathsf{Bird} \sqcap \mathsf{Broken\_Wing} \sqsubseteq \neg \mathsf{Fly} \\ \mathsf{Super\_Penguin} \sqsubseteq \mathsf{Penguin} \sqcap \mathsf{Fly} \end{array} \right\}, \quad \text{and}$$

$$A = \left\{ \begin{array}{l} \mathsf{OPUS} : \mathsf{Super\_Penguin} \\ \mathsf{OPUS} : \mathsf{Broken\_Wing} \end{array} \right\}.$$

The Tbox T says that penguins are birds; birds can fly unless they have a broken wing, and superpenguins are penguins capable of flying. The Abox A asserts that it is known that Opus is a superpenguin having a broken wing.

We will show intuitively why Σ_1 is inconsistent. As Opus is a super-penguin with a broken wing, and super-penguins are penguins, then he cannot fly (*i.e.*, OPUS is an instance of $\neg$Fly). At the same time, as Opus is a super-penguin, so he is also a penguin, and indirectly a bird, he is able to fly (*i.e.*, OPUS is an instance of Fly). Therefore, $\mathsf{Super\_Penguin} \sqsubseteq \mathsf{Fly} \sqcap \neg \mathsf{Fly} = \bot$, so Super_Penguin should be an empty concept but is not.

In the next section, we will introduce Defeasible Logic Programming, a formalism for knowledge representation and non-monotonic reasoning that can handle situations like the one presented above in a natural way.

2.2. Defeasible Logic Programming

When a rule supporting a conclusion may be defeated by new information, it is said that such reasoning is *defeasible* (Pollock, 1974, 1987; Nute, 1988; Pollock, 1995; Simari & Loui, 1992). When defeasible reasons or rules are chained to reach a conclusion, we have *arguments* instead of proofs. Arguments may compete, rebutting each other, so that a *process* of argumentation is a natural result of the search for arguments. Adjudication of competing arguments must be performed, comparing arguments in order to determine what beliefs are ultimately accepted as *warranted* or *justified*. Preference among conflicting arguments is defined in terms of a *preference criterion* which establishes a relation "$\preceq$" among possible arguments; thus, for two arguments

$\mathscr{A}$ and $\mathscr{B}$ in conflict, it may be the case that $\mathscr{A}$ is strictly preferred over $\mathscr{B}$ ($\mathscr{A} \succ \mathscr{B}$), that $\mathscr{A}$ and $\mathscr{B}$ are equally preferable ($\mathscr{A} \succeq \mathscr{B}$ and $\mathscr{A} \preceq \mathscr{B}$) or that $\mathscr{A}$ and $\mathscr{B}$ are not comparable with each other. In the above setting, since we arrive at conclusions by building defeasible arguments, and since *logical argumentation* is usually referred to as *argumentation*, we sometimes call this kind of reasoning *defeasible argumentation*.

The growing success of argumentation-based approaches has caused a rich crossbreeding with other disciplines, providing interesting results in different areas such as group decision making (Zhang, Sun, & Chen, 2005), knowledge engineering (Carbogim, Robertson, & Lee, 2000), legal reasoning (Prakken & Sartor, 2002; Verheij, 2005), and multiagent systems (Parsons, Sierrra, & Jennings, 1998; Sierra & Noriega, 2002; Rahwan et al., 2003), among others. During the last decade several defeasible argumentation frameworks have been developed, most of them on the basis of suitable extensions to logic programming (see Chesñevar, Maguitman, & Loui, 2000; Prakken & Vreeswijk, 2002; Kakas & Toni, 1999). *Defeasible logic programming* (DeLP) (García & Simari, 2004) is one of such formalisms, combining results from defeasible argumentation theory (Simari & Loui, 1992) and logic programming (Lloyd, 1987). DeLP is a suitable framework for building real-world applications which has proven to be particularly attractive in that context, such as clustering (Gómez & Chesñevar, 2004), intelligent web search (Chesñevar & Maguitman, 2004b; Chesñevar, Maguitman, & Simari, in press), knowledge management (Chesñevar, Brena, & Aguirre, 2005a, 2005b), multiagent systems (Brena, Chesñevar, & Aguirre, 2006), and natural language processing (Chesñevar & Maguitman, 2004a), intelligent web forms (Gómez, Chesñevar, & Simari, 2008), among others.

Defeasible Logic Programming (DeLP) (García & Simari, 2004) provides a language for knowledge representation and reasoning that uses *defeasible argumentation* (Chesñevar et al., 2000; Prakken & Vreeswijk, 2002; Simari & Loui, 1992) to decide between contradictory conclusions through a *dialectical analysis*. In a defeasible logic program $\mathscr{P} = (\Pi, \Delta)$, a set Π of strict rules $P \leftarrow Q_1, \ldots, Q_n$, and a set Δ of defeasible rules $P \prec Q_1, \ldots, Q_n$ can be distinguished.

Definition 1. $\mathscr{L}_{DeLP} =_{df} \mathscr{L}_{DeLP_\Pi} \cup \mathscr{L}_{DeLP_\Delta}$ is the language of DeLP programs, where $\mathscr{L}_{DeLP_\Pi}$ is the language of DeLP programs formed by strict rules $B \leftarrow A_1, \ldots, A_n$ with ($n \geqslant 1$) and facts B (*i.e.*, rules where $n = 0$), and $\mathscr{L}_{DeLP_\Delta}$ is the language of DeLP programs formed only by defeasible rules $B \prec A_1, \ldots, A_n$ with ($n \geqslant 1$).

Literals can be positive or negative. The complement of a literal L (noted as $\bar{L}$) is p if $L = \sim p$ and $\sim p$ if $L = p$. Notice that there is an extension to DeLP that allows to define *presumptions* (or defeasible rules without body) that model defeasible facts (see (García & Simari, 2004, Section 6.2)); they are however outside the scope of this work.

Deriving literals in DeLP results in the construction of *arguments*. An argument $\mathscr{A}$ is a (possibly empty) set of ground (*i.e.*, without variables) defeasible rules that together with the set Π provides a logical proof for a given literal Q, satisfying the additional requirements of *non-contradiction* and *minimality*. Formally:

Definition 2. Given a DeLP program $\mathscr{P}$, an *argument* $\mathscr{A}$ for a query Q, denoted $\langle \mathscr{A}, Q \rangle$, is a subset of ground instances of defeasible rules in $\mathscr{P}$, such that: (i) there exists a *defeasible derivation* for Q from $\Pi \cup \mathscr{A}$; (ii) $\Pi \cup \mathscr{A}$ is non-contradictory (*i.e.*, $\Pi \cup \mathscr{A}$ does not entail two complementary literals P and $\sim P$), and, (iii) there is no $\mathscr{A}' \subset \mathscr{A}$ such that there exists a defeasible derivation for Q from $\Pi \cup \mathscr{A}'$. An argument $\langle \mathscr{A}_1, Q_1 \rangle$ is a *sub-argument* of another argument $\langle \mathscr{A}_2, Q_2 \rangle$ if $\mathscr{A}_1 \subseteq \mathscr{A}_2$.

The notion of defeasible derivation corresponds to the usual query-driven SLD derivation used in logic programming, performed by backward chaining on both strict and defeasible rules; in this context a negated literal $\sim P$ is treated just as a new predicate name *no_P*. Minimality imposes a kind of 'Occam's razor principle' on argument construction. The non-contradiction requirement forbids the use of (ground instances of) defeasible rules in an argument $\mathscr{A}$ whenever $\Pi \cup \mathscr{A}$ entails two complementary literals. The notion of contradiction is captured by the notion of counterargument.

Definition 3. An argument $\langle \mathscr{A}_1, Q_1 \rangle$ is a *counterargument* for an argument $\langle \mathscr{A}_2, Q_2 \rangle$ iff there is a subargument $\langle \mathscr{A}, Q \rangle$ of $\langle \mathscr{A}_2, Q_2 \rangle$ such that the set $\Pi \cup \{Q_1, Q\}$ is contradictory. An argument $\langle \mathscr{A}_1, Q_1 \rangle$ is a *defeater* for an argument $\langle \mathscr{A}_2, Q_2 \rangle$ if $\langle \mathscr{A}_1, Q_1 \rangle$ counterargues $\langle \mathscr{A}_2, Q_2 \rangle$, and $\langle \mathscr{A}_1, Q_1 \rangle$ is preferred over $\langle \mathscr{A}_2, Q_2 \rangle$ w.r.t. a *preference criterion* $\preceq$ on conflicting arguments. Such criterion is defined as a partial order $\preceq \subseteq Args(\mathscr{P}) \times Args(\mathscr{P})$. The argument $\langle \mathscr{A}_1, Q_1 \rangle$ will be called a *proper defeater* for $\langle \mathscr{A}_2, Q_2 \rangle$ iff $\langle \mathscr{A}_1, Q_1 \rangle$ is strictly preferred over $\langle \mathscr{A}, Q \rangle$ w.r.t. $\preceq$; if $\langle \mathscr{A}_1, Q_1 \rangle$ and $\langle \mathscr{A}, Q \rangle$ are unrelated to each other will be called a *blocking defeater* for $\langle \mathscr{A}_2, Q_2 \rangle$.

Generalized specificity (Simari & Loui, 1992) is typically used as a syntax-based preference criterion among conflicting arguments, favoring those arguments which are *more informed* or *more direct* (Simari & Loui, 1992; Stolzenburg, García, Chesñevar, & Simari, 2003). For example, let us consider three arguments $\langle \{a \prec b, c\}, a \rangle$, $\langle \{\sim a \prec b\}, \sim a \rangle$ and $\langle \{(a \prec b); (b \prec c)\}, a \rangle$ built on the basis of a program $\mathscr{P} = (\Pi, \Delta)$ where

$$\Pi = \{ b, c \}$$

$$\Delta = \left\{ \begin{array}{l} b \prec c; \\ a \prec b; \\ a \prec b, c; \\ \sim a \prec b \end{array} \right\}.$$

When using generalized specificity as the comparison criterion between arguments, the argument $\langle \{a \prec b, c\}, a \rangle$ would be preferred over the argument $\langle \{\sim a \prec b\}, \sim a \rangle$ as the former is considered *more informed* (*i.e.*, it relies on more premises). However, argument $\langle \{\sim a \prec b\}, \sim a \rangle$ is preferred over $\langle \{(a \prec b); (b \prec c\}), a \rangle$as the former is regarded as *more direct* (*i.e.*, it is obtained from a shorter derivation). However, it must be remarked that besides specificity other alternative preference criteria could also be used; *e.g.*, considering numerical values corresponding to necessity measures attached to argument conclusions (Chesñevar, Simari, Alsinet, & Godo, 2004, 2005) or defining argument comparison using rule priorities. This last approach is used in D-PROLOG (Nute, 1988), Defeasible Logic (Nute, 1992), extensions of Defeasible Logic (Antoniou, Maher, & Billington, 2000, 1998), and logic programming without negation as failure (Kakas, Mancarella, & Dung, 1994; Dimopoulos & Kakas, 1995).

In order to determine whether a given argument $\mathscr{A}$ is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for $\mathscr{A}$, defeaters for these defeaters, and so on, are taken into account. An *argumentation line* starting in an argument $\langle \mathscr{A}_0, Q_0 \rangle$ is a sequence $[\langle \mathscr{A}_0, Q_0 \rangle, \langle \mathscr{A}_1, Q_1 \rangle, \langle \mathscr{A}_2, Q_2 \rangle, \ldots, \langle \mathscr{A}_n, Q_n \rangle \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). Each $\langle \mathscr{A}_i, Q_i \rangle$ is a defeater for the previous argument $\langle \mathscr{A}_{i-1}, Q_{i-1} \rangle$ in the sequence, $i > 0$. In order to avoid *fallacious* reasoning, dialectics imposes additional constraints on such an argument exchange to be considered rationally *acceptable*. Given a DeLP program $\mathscr{P}$ and an initial argument $\langle \mathscr{A}_0, Q_0 \rangle$, the set of all acceptable argumentation lines starting in $\langle \mathscr{A}_0, Q_0 \rangle$ accounts for a whole dialectical analysis for $\langle \mathscr{A}_0, Q_0 \rangle$ (*i.e.*, all possible dialogues about $\langle \mathscr{A}_0, Q_0 \rangle$ between proponent and opponent), formalized as a *dialectical tree*.

Nodes in a dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ 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 $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ 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. An argument $\langle \mathcal{A}_0, Q_0 \rangle$ is ultimately accepted as valid (or *warranted*) w.r.t. a DeLP program $\mathcal{P}$ iff the root of its associated dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ is labeled as *U-node*.

Given a DeLP program $\mathcal{P}$, solving a query Q w.r.t. $\mathcal{P}$ accounts for determining whether Q is supported by (at least) one warranted argument. Different doxastic attitudes can be distinguished as follows: *Yes*, accounts for believing Q iff there is at least one warranted argument supporting Q on the basis of $\mathcal{P}$; *No*, accounts for believing $\sim Q$ iff there is at least one warranted argument supporting $\sim Q$ on the basis of $\mathcal{P}$; *Undecided*, neither Q nor $\sim Q$ are warranted w.r.t. $\mathcal{P}$, and *Unknown*, Q does not belong to the signature of $\mathcal{P}$.

We present a property from García and Simari (2004) that will be useful for proving some results about our proposal (see Section 5).

Proposition 1. *For any program $\mathcal{P}$ in DeLP, it cannot be the case that both Q and $\sim Q$ are warranted.*

In the next sections we will present a method for expressing arbitrary DL ontologies in DeLP. For now on, based on the fact discovered by Grosof et al. (2003) that DL inclusion axioms of the form "$C \sqsubseteq D$" can be equated to PROLOG rules of the form "$D(X) :- C(X)$", we will show how the application problems modeled by the ontologies presented in Section 2.1 can be expressed in DeLP in such a way that it could be possible for an agent to automatically reason in such cases. Notice also that following PROLOG usual conventions, predicates are noted with lowercase letters (*e.g.*, *bird*, *fly*, etc.), variable names with capital letters (*e.g.*, *X*,*Y*,*Z*, etc.), and constant names with lowercase letters (*e.g.*, *opus*, etc.).

Example 2. Here follows the DeLP program $\mathcal{P}_2 = (\Pi, \Delta)$ that models the situation presented in Example 1:

$$\Pi = \left\{ \begin{array}{l} bird(X) \leftarrow penguin(X) \\ penguin(X) \leftarrow super\_penguin(X) \\ super\_penguin(opus) \\ broken\_wing(opus) \end{array} \right\}, \text{ and}$$

$$\Delta = \left\{ \begin{array}{l} \sim fly(X) \prec bird(X), broken\_wing(X) \\ fly(X) \prec bird(X) \\ fly(X) \prec super\_penguin(X) \end{array} \right\}.$$

In DeLP, the answer for *fly*(*opus*) is *Yes* while the answer for $\sim$*fly*(*opus*) is *No* as it is shown next. It is possible to build an argument $\langle \mathcal{A}_1, fly(opus) \rangle$, where:

$$\mathcal{A}_1 = \{ fly(opus) \prec bird(opus) \}.$$

This argument is properly defeated by another argument $\langle \mathcal{A}_2, \sim fly(opus) \rangle$, with:

$$\mathcal{A}_2 = \{ \sim fly(opus) \prec bird(opus), broken\_wing(opus) \}.$$

However, argument $\mathcal{A}_1$ is reinstated by another argument $\langle \mathcal{A}_3, fly(opus) \rangle$ which is a blocking defeater for $\mathcal{A}_2$, where:

$$\mathcal{A}_3 = \{ fly(opus) \prec super\_penguin(opus) \}.$$

In Section 3 we will review under which constraints the translation of arbitrary DL ontologies to DeLP can be performed.

3. Reasoning with inconsistent ontologies: δ-ontologies

In the presence of inconsistent ontologies, traditional DL reasoners (such as Racer (Haarslev & Möller, 2001)) issue an error message and stop further processing. Thus the burden of repairing the ontology (*i.e.*, making it consistent) 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.

In (Gómez et al., 2010), we propose using DeLP to perform such a task by translating DL ontologies into DeLP programs. By doing so we gain the capability of reasoning with inconsistent ontologies. However this approach also loses some expressiveness in the involved ontologies. As we will show in Definition 4, certain restrictions will have to be imposed on DL ontologies in order to be expressed in the DeLP language.

The work (Gómez et al., 2010) is based in part in (Grosof et al., 2003) 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. Given a DL ontology $\Sigma = (T, A)$, we will consider the Tbox T as partitioned into two disjoint sets—a *strict terminology* T_S and a *defeasible terminology* T_D—such that $T = T_S \cup T_D$ and $T_S \cap T_D = \emptyset$. Notice that in this work we suppose that the knowledge engineer decides if an axiom is strict or defeasible (for a discussion on how partitioning automatically DL terminologies, see (Gómez et al., 2010, p. 138)).

We therefore propose translating the DL ontology Σ into a DeLP program $\mathcal{P} = (\Pi, \Delta) = \mathcal{T}(\Sigma)$ by means of a mapping $\mathcal{T}$ from the DL language to the DeLP language. Intuitively the set Π of strict rules in $\mathcal{P}$ will correspond to the Abox A joined with T_S in Σ, and the set Δ of defeasible rules will correspond to T_D in Σ. This translation will be achieved by two specialized functions $\mathcal{T}_\Pi$ and $\mathcal{T}_\Delta$, where $\mathcal{T}_\Pi$ translates from a set of DL sentences into a set of DeLP strict rules and $\mathcal{T}_\Delta$ translates from a set of DL sentences into a set of DeLP defeasible rules, such that $\Pi = \mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A)$ and $\Delta = \mathcal{T}_\Delta(T_D)$. The parallel between a reasoning system based on DeLP and DL can be seen in Fig. 1.

Definition 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*.

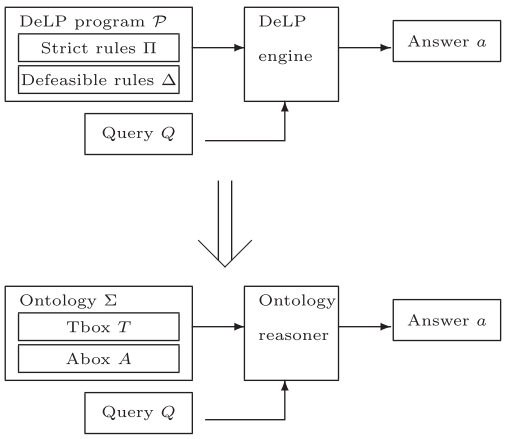

Fig. 1. Parallel between reasoning with DeLP and DL.

Strict terminology T_S^3:

$$T_S^3 = \{\ \mathsf{Checks\_web\_mail} \sqsubseteq \mathsf{Uses\_browser}\ \}$$

Defeasible terminology T_D^3:

$$T_D^3 = \left\{ \begin{array}{l} \mathsf{Studies} \sqsubseteq \mathsf{Pass} \\ \mathsf{Sits\_at\_computer} \sqsubseteq \mathsf{Studies} \\ \mathsf{Sits\_at\_computer} \sqcap \mathsf{Web\_surfing} \sqsubseteq \neg\mathsf{Studies} \\ \neg\mathsf{Studies} \sqsubseteq \neg\mathsf{Pass} \\ \mathsf{Uses\_browser} \sqsubseteq \mathsf{Web\_surfing} \\ \mathsf{Uses\_browser} \sqcap \mathsf{Reads\_javadoc} \sqsubseteq \neg\mathsf{Web\_surfing} \end{array} \right\}$$

Assertional box A^3:

$$A^3 = \left\{ \begin{array}{l} \mathsf{JOHN} : \mathsf{Sits\_at\_computer} \\ \mathsf{PAUL} : \mathsf{Sits\_at\_computer} \\ \mathsf{PAUL} : \mathsf{Uses\_browser} \\ \mathsf{MARY} : \mathsf{Sits\_at\_computer} \\ \mathsf{MARY} : \mathsf{Checks\_web\_mail} \\ \mathsf{MARY} : \mathsf{Reads\_javadoc} \end{array} \right\}$$

Fig. 2. Ontology $\Sigma_3 = \left(T_S^3, T_D^3, A^3\right)$.

Definition 5. Let C be an $\mathscr{L}_b$-class, D an $\mathscr{L}_h$-class, A, B $\mathscr{L}_{hb}$-classes, P,Q properties, a,b individuals. Let T be a set of inclusion and equality sentences in $\mathscr{L}_{DL}$ of the form $C \sqsubseteq D$, $A \equiv B$, $\top \sqsubseteq\ \forall P.D$,$\top \sqsubseteq \forall P^{-}.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^{-}$, or $P^{+} \sqsubseteq P$ such that T can be partitioned into two disjoint sets T_S and T_D. Let A be a set of assertions disjoint with T of the form $a{:}D$ or $\langle a,b\rangle{:}P$. A *δ-ontology* Σ is a tuple (T_S,T_D,A). The set T_S is called the *strict terminology* (or Sbox), T_D the *defeasible terminology* (or Dbox) and A the *assertional box* (or Abox).

Example 3. Consider the δ-ontology $\Sigma_3 = \left(T_S^3, T_D^3, A^3\right)$ presented in Fig. 2. The strict terminology T_S^3 says that somebody who is checking mail uses a web browser. The defeasible terminology T_D^3 expresses that those who study usually pass exams, someone who is sitting at a computer is normally studying unless he is web surfing, those who do not study usually do not pass, if someone is using a web browser then he is presumably web surfing unless he is reading Javadoc documentation. The set A^3 asserts that it is known that John, Paul and Mary are sitting at a computer; Paul is using a browser; finally, Mary is also checking mail and reading Javadoc documentation. Notice that the traditional (in the sense of (Baader et al., 2003)) DL ontology $\left(T_S^3 \cup T_D^3, A^3\right)$ is incoherent since somebody who both sits at a computer and web surfs then belongs both to the "Studies" concept and to its complement, rendering the concept empty.

For assigning semantics to a δ-ontology two translation functions $\mathscr{T}_\Delta$ and $\mathscr{T}_\Pi$ from DL to DeLP based on the work of (Grosof et al., 2003) are defined. Firstly we will discuss informally why the interpretation of DL ontologies as DeLP programs is sound. Sboxes are going to be interpreted as strict rules, Aboxes as facts and Dboxes as defeasible rules (thus considering them as *default* inclusions). The basic premise for achieving the translation of DL ontologies into DeLP is based on the observation that a DL inclusion axiom "C⊑D" is regarded as a First-Order Logic statement "$(\forall x)(C(x) \rightarrow D(x))$", which in turn is regarded as a Horn-clause "$d(X) \leftarrow c(X)$". Naturally "C ⊓ D⊑E" is treated as "$e(X) \leftarrow c(X),d(X)$". Lloyd-Topor transformations are used to handle special cases as conjunctions in the head of rules and disjunctions in the body of rules; so "C⊑D ⊓ E" is interpreted as two rules "$d(X) \leftarrow c(X)$" and "$e(X) \leftarrow c(X)$" while "C ⊔ D⊑E" is transformed into "$e(X) \leftarrow c(X)$" and "$e(X) \leftarrow d(X)$". Likewise axioms of the form "∃r.C⊑D" are treated as "$d(X) \leftarrow r(X,Y),c(Y)$". Dbox axioms are treated as *defeasible* and are transformed using the $\mathscr{T}_\Delta$ function (*e.g.*, $\mathscr{T}_\Delta$(C ⊑ D) is interpreted as $d(X) \prec c(X)$); Sbox axioms are considered *strict* and are transformed using $\mathscr{T}_\Pi$ (*e.g.*, $\mathscr{T}_\Pi$(C ⊑ D) is interpreted as $\{(d(X) \leftarrow c(X)), (\sim c(X) \leftarrow \sim d(X))\}$). Abox assertions are always considered strict (*e.g.*, $\mathscr{T}_\Pi$(a : C) is regarded as a fact $c(a)$ and $\mathscr{T}_\Pi(\langle a, b\rangle : r)$ as $r(a,b)$). Notice that there are no technical obstacles to prevent considering Abox assertions as defeasible. In fact, DeLP considers defeasible facts as *presumptions*; however this extension of the δ-ontologies framework is outside the scope of this paper.

Next we recall more formally how DL axioms are interpreted in DeLP. Definition 6 shows how Dbox axioms are translated as DeLP defeasible rules.

Definition 6. Let A, C, D be concepts, X, Y variables, P, Q properties. The $\mathscr{T}_\Delta : 2^{\mathscr{L}_{DL}} \rightarrow 2^{\mathscr{L}_{DeLP_\Delta}}$ mapping is defined in Fig. 3. Besides, intermediate transformations that will end as rules of the form "$(H_1 \wedge H_2) \prec B$" will be rewritten as two rules "$H_1 \prec B$" and "$H_2 \prec B$" (as this is an incorrect DeLP syntax). Similarly transformations of the form "$H_1 \prec H_2 \prec\ B$" will be rewritten as "$H_1 \prec B \wedge H_2$", and transformations of the form "$H \prec\ (B_1 \vee B_2)$" will be rewritten as two rules "$H \prec B_1$" and "$H \prec B_2$".

Definitions 7–9 show how both Sbox axioms and Abox assertions are interpreted as DeLP strict rules.

Definition 7. Let A, C, D be concepts, X, Y variables, P, Q properties. The $\mathscr{T}^*_\Pi : 2^{\mathscr{L}_{DL}} \rightarrow 2^{\mathscr{L}_{DeLP_\Pi}}$ mapping is defined in Fig. 4. 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 "$H \leftarrow B_2$".

As DeLP is based on SLD-derivation of literals, simple translation of DL sentences to DeLP strict rules does not allow to infer negative information by *modus tollens*. For instance, "C⊑D" (all C's are D's) is translated as "$D(X) \leftarrow C(X)$", DeLP is not able to derive "$\sim C(a)$" from "$\sim D(a)$". Thus given "$C_1 \sqcap C_2 \sqcap \ldots \sqcap\ C_{n-1} \sqcap C_n \sqsubseteq D$", instead of only including the strict rule "$D(X) \leftarrow C_1(X),C_2(X),\ldots,C_{n-1}(X),C_n(X)$" in its translation, we propose including all of its *transposes*.

Definition 8. Let $r = H \leftarrow B_1,B_2,B_3,\ldots,B_{n-1},B_n$ be a DeLP strict rule. The *set of transposes of rule r*, noted as "Trans(r)", is defined as:

$$\mathsf{Trans}(r) = \left\{ \begin{array}{l} H \leftarrow B_1, B_2, \ldots, B_{n-1}, B_n \\ \overline{B_1} \leftarrow \overline{H}, B_2, B_3, \ldots, B_{n-1}, B_n \\ \overline{B_2} \leftarrow \overline{H}, B_1, B_3, \ldots, B_{n-1}, B_n \\ \overline{B_3} \leftarrow \overline{H}, B_1, B_2, \ldots, B_{n-1}, B_n \\ \ldots \\ \overline{B_{n-1}} \leftarrow \overline{H}, B_1, B_2, B_3 \ldots, B_n \\ \overline{B_n} \leftarrow \overline{H}, B_1, B_2, \ldots, B_{n-1} \end{array} \right\}$$

Definition 9. We define the mapping from DL ontologies into DeLP strict rules as $\mathscr{T}_\Pi(T) = \mathsf{Trans}\left(\mathscr{T}^*_\Pi(T)\right)$.

Notice that reasoning with transposed strict rules is not more computationally expensive than reasoning without them. Observe that even though as this seems computationally expensive it is

$$
\begin{array}{rcl}
\mathcal{T}_\Delta(\{C \sqsubseteq D\}) & =_{df} & \left\{\ T_h(D,X) \prec T_b(C,X)\ \right\}, \\
 & & \text{if } C \text{ is an } \mathcal{L}_b\text{-class and } D \text{ an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Delta(\{C \equiv D\}) & =_{df} & \mathcal{T}_\Delta(\{C \sqsubseteq D\}) \cup \mathcal{T}_\Delta(\{D \sqsubseteq C\}), \\
 & & \text{if } C \text{ and } D \text{ are } \mathcal{L}_{hb}\text{-classes} \\
\mathcal{T}_\Delta(\{\top \sqsubseteq \forall P.D\}) & =_{df} & \left\{\ T_h(D,Y) \prec P(X,Y)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Delta(\{\top \sqsubseteq \forall P^-.D\}) & =_{df} & \left\{\ T_h(D,X) \prec P(X,Y)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Delta(\{P \sqsubseteq Q\}) & =_{df} & \left\{\ Q(X,Y) \prec P(X,Y)\ \right\} \\
\mathcal{T}_\Delta(\{P \equiv Q\}) & =_{df} & \left\{\begin{array}{l} Q(X,Y) \prec P(X,Y) \\ P(X,Y) \prec Q(X,Y) \end{array}\right\} \\
\mathcal{T}_\Delta(\{P \equiv Q^-\}) & =_{df} & \left\{\begin{array}{l} Q(X,Y) \prec P(Y,X) \\ P(Y,X) \prec Q(X,Y) \end{array}\right\} \\
\mathcal{T}_\Delta(\{P^+ \sqsubseteq P\}) & =_{df} & \left\{\ P(X,Z) \prec P(X,Y) \wedge P(Y,Z)\ \right\} \\
\mathcal{T}_\Delta(\{s_1,\ldots,s_n\}) & =_{df} & \bigcup_{i=1}^{n} \{\mathcal{T}_\Delta(\{s_i\})\}, \text{ if } n > 1 \\
\textbf{where:} & & \\
T_h(A,X) & =_{df} & A(X) \\
T_h((C \sqcap D),X) & =_{df} & T_h(C,X) \wedge T_h(D,X) \\
T_h((\forall R.C),X) & =_{df} & T_h(C,Y) \prec R(X,Y) \\
T_b(A,X) & =_{df} & A(X) \\
T_b((C \sqcap D),X) & =_{df} & T_b(C,X) \wedge T_b(D,X) \\
T_b((C \sqcup D),X) & =_{df} & T_b(C,X) \vee T_b(D,X) \\
T_b((\exists R.C),X) & =_{df} & R(X,Y) \wedge T_b(C,Y)
\end{array}
$$

Fig. 3. Mapping from DL ontologies to DeLP defeasible rules.

$$
\begin{array}{rcl}
\mathcal{T}^*_\Pi(\{C \sqsubseteq D\}) & =_{df} & \left\{\ T_h(D,X) \leftarrow T_b(C,X)\ \right\}, \\
 & & \text{if } C \text{ is an } \mathcal{L}_b\text{-class and } D \text{ an } \mathcal{L}_h\text{-class} \\
\mathcal{T}^*_\Pi(\{C \equiv D\}) & =_{df} & \mathcal{T}^*_\Pi(\{C \sqsubseteq D\}) \cup \mathcal{T}^*_\Pi(\{D \sqsubseteq C\}), \\
 & & \text{if } C \text{ and } D \text{ are } \mathcal{L}_{hb}\text{-classes} \\
\mathcal{T}^*_\Pi(\{\top \sqsubseteq \forall P.D\}) & =_{df} & \left\{\ T_h(D,Y) \leftarrow P(X,Y)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}^*_\Pi(\{\top \sqsubseteq \forall P^-.D\}) & =_{df} & \left\{\ T_h(D,X) \leftarrow P(X,Y)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}^*_\Pi(\{a : D\}) & =_{df} & \left\{\ T_h(D,a)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}^*_\Pi(\{\langle a,b\rangle : P\}) & =_{df} & \left\{\ P(a,b)\ \right\} \\
\mathcal{T}^*_\Pi(\{P \sqsubseteq Q\}) & =_{df} & \left\{\ Q(X,Y) \leftarrow P(X,Y)\ \right\} \\
\mathcal{T}^*_\Pi(\{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}^*_\Pi(\{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}^*_\Pi(\{P^+ \sqsubseteq P\}) & =_{df} & \left\{\ P(X,Z) \leftarrow P(X,Y) \wedge P(Y,Z)\ \right\} \\
\mathcal{T}^*_\Pi(\{s_1,\ldots,s_n\}) & =_{df} & \bigcup_{i=1}^{n} \mathcal{T}^*_\Pi(\{s_i\}), \text{ if } n > 1 \\
\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. 4. Mapping from DL ontologies to DeLP strict rules.

only as expensive as deriving $\sim b$ from $a \leftarrow b$ and $\sim a$. Notice also that, following trends in non-monotonic reasoning, we do not consider transposition of defeasible rules (Brewka, Dix, & Konolige, 1997, p. 45); (Caminada, 2008).

The reader should also notice that the price paid for translating DL ontologies into DeLP programs is the loss of some expressiveness. For instance, as noted by Grosof et al. (2003), DL axioms that generate a rule with a disjunction in its head cannot be represented in logic programming.

Definition 10. Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology. The *interpretation of* Σ is a DeLP program $\mathscr{P} = (\mathscr{T}_\Pi(T_S) \cup \mathscr{T}_\Pi(A), \mathscr{T}_\Delta(T_D))$.

Remark 1. Notice that in order to keep consistency within an argument, some internal coherence between the Abox and the Tbox must be enforced; namely given a δ-ontology $\Sigma = (T_S, T_D, A)$, it must not be possible to derive two complementary literals from $\mathscr{T}_\Pi(T_S) \cup \mathscr{T}_\Pi(A)$.

We recall how the reasoning task of *instance checking* (Baader et al., 2003, p. 19) is interpreted in δ-ontologies:

Definition 11. Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology, C a class name, a an individual, and $\mathscr{P} = (\mathscr{T}_\Pi(T_S) \cup \mathscr{T}_\Pi(A), \mathscr{T}_\Delta(T_D))$.

1. The *individual a potentially belongs to class C* iff there exists an argument $\langle \mathscr{A}, C(a) \rangle$ w.r.t. $\mathscr{P}$;
2. the *individual a justifiedly belongs to class C* iff there exists a warranted argument $\langle \mathscr{A}, C(a) \rangle$ w.r.t. $\mathscr{P}$, and,
3. the *individual a strictly belongs to class C* iff there exists an argument $\langle \emptyset, C(a) \rangle$ w.r.t. $\mathscr{P}$.

Example 4. Consider again the δ-ontology Σ_3 presented in Example 3, which is interpreted as the DeLP program $\mathscr{P}_3$ according to Definition 10 as shown in Fig. 5. From $\mathscr{P}_1$, we can determine that John justifiedly belongs to the concept Pass in Σ_3 as there exists a warranted argument structure $\langle \mathscr{A}_1, pass(john) \rangle$ that says that John will pass the exam as he studies (because he sits at a computer), where

$$\mathscr{A}_1 = \left\{ \begin{array}{l} (pass(john) \prec studies(john)), \\ (studies(john) \prec sits\_at\_computer(john)) \end{array} \right\}.$$

We cannot reach a decision w.r.t. the membership of Paul to the concept "Pass" because there are two arguments attacking each other, so the answer to the query *pass*(*paul*) is *undecided*. Formally, there exist two arguments $\langle \mathscr{B}_1, pass(paul) \rangle$ and $\langle \mathscr{B}_2, \sim pass(paul) \rangle$, where:

$$\mathscr{B}_1 = \left\{ \begin{array}{l} (pass(paul) \prec studies(paul)), \\ (studies(paul) \prec sits\_at\_computer(paul)) \end{array} \right\}, \text{ and}$$

$$\mathscr{B}_2 = \left\{ \begin{array}{l} (\sim pass(paul) \prec \sim studies(paul)), \\ (\sim studies(paul) \prec sits\_at\_computer(paul), web\_surfing(paul)), \\ (web\_surfing(paul) \prec uses\_browser(paul)) \end{array} \right\}.$$

In the case of Mary's membership to Pass, there is an argument $\langle \mathscr{C}_1, pass(mary) \rangle$. that has two defeaters, $\langle \mathscr{C}_2, \sim pass(mary) \rangle$ and $\langle \mathscr{C}_3, \sim studies(mary) \rangle$, which are both defeated by $\langle \mathscr{C}_4, \sim web\_surfing(mary) \rangle$, where:

$$\mathscr{C}_1 = \left\{ \begin{array}{l} (pass(mary) \prec studies(mary)), \\ (studies(mary) \prec sits\_at\_computer(mary)) \end{array} \right\},$$

$$\mathscr{C}_2 = \{ \sim pass(mary) \prec \sim studies(mary) \} \cup \mathscr{C}_3,$$

$$\mathscr{C}_3 = \left\{ \begin{array}{l} (\sim studies(mary) \prec sits\_at\_computer(mary), web\_surfing(mary)), \\ (web\_surfing(mary) \prec uses\_browser(mary)) \end{array} \right\}, \text{ and}$$

$$\mathscr{C}_4 = \left\{ \begin{array}{l} (\sim web\_surfing(mary) \prec uses\_browser(mary), reads\_javadoc(mary)), \\ (uses\_browser(mary) \prec checks\_web\_mail(mary)) \end{array} \right\}.$$

Therefore, Mary belongs justifiedly to the concept "Pass" as the literal *pass*(*mary*) is warranted. The dialectical trees for the three queries are depicted graphically in Fig. 6.

4. The ONTOarg framework: architecture

The notion of δ-ontology is powerful enough to reason defeasibly within a specific ontology with inconsistencies. However, a common problem for a knowledge engineer is associated with dealing with different δ-ontologies, which have to be unified or integrated in order to get a global understanding of the knowledge available in such ontologies, resulting in turn in the characterization of a global ontology. In the literature (Calvanese et al., 2001), the term *ontology integration* is associated with combining ontologies residing in different sources in order to provide the user with a unified view of such ontologies. The problem of designing systems for ontology integration in the Semantic Web is particularly important because ontologies are to be developed independently from each other and, for this reason, they can be mutually inconsistent. One possible architecture for ontology integration systems is based on a global schema and a set of local sources. The local sources contain the actual data while the global schema provides a reconciled, unified view of the underlying sources. A basic service provided by ontology integration systems is that of answering queries posed in terms of the global schema.

There are two main approaches to data integration, namely *global-as-view* (GAV) and *local-as-view* (LAV). In the LAV approach, we assume a global ontology $\mathscr{G}$, a set $\mathscr{S}$ of local/source ontologies, and the mapping between the global and the local ontologies is given by associating each term in the local ontologies with a *view* $V_{\mathscr{G}}$ over the global ontology (Calvanese et al., 2001, Section 4). The intended meaning of associating with a term C in $\mathscr{S}$ a view $V_{\mathscr{G}}$ over $\mathscr{G}$ is that such a view represents the best way to characterize the instances of C using the concepts in $\mathscr{G}$. The correspondence between C and the associated view can be *sound* (all the individuals satisfying C satisfy $V_{\mathscr{G}}$), *complete* (if no individual other than those satisfying C satisfies $V_{\mathscr{G}}$), and/or *exact* (the set of individuals that satisfy C is exactly the set of individuals that satisfy $V_{\mathscr{G}}$).

In the GAV and LAV approaches to data integration, the queries w.r.t. the target ontology are reformulated w.r.t. the sources. Haase

DeLP program $\mathcal{P}_3 = (\Pi_3, \Delta_3)$ obtained from Σ_3:

Facts and strict rules Π_3:

$$\Pi_3 = \left\{ \begin{array}{ll} sits\_at\_computer(john). & sits\_at\_computer(paul). \\ uses\_browser(paul). & sits\_at\_computer(mary). \\ checks\_web\_mail(mary). & reads\_javadoc(mary). \\ uses\_browser(X) \leftarrow checks\_web\_mail(X). & \sim checks\_web\_mail(X) \leftarrow \sim uses\_browser(X). \end{array} \right\}$$

Defeasible rules Δ_3:

$$\Delta_3 = \left\{ \begin{array}{l} pass(X) \prec studies(X). \\ studies(X) \prec sits\_at\_computer(X). \\ \sim studies(X) \prec sits\_at\_computer(X), web\_surfing(X). \\ \sim pass(X) \prec \sim studies(X). \\ web\_surfing(X) \prec uses\_browser(X). \\ \sim web\_surfing(X) \prec uses\_browser(X), reads\_javadoc(X). \end{array} \right\}$$

Fig. 5. DeLP program $\mathscr{P}_3$ interpreting ontology Σ_3.

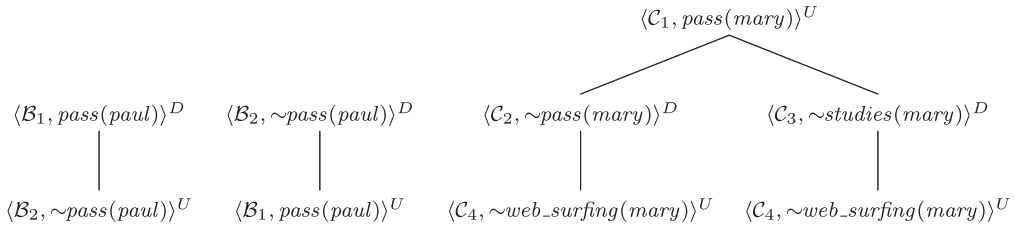

Fig. 6. Dialectical analyses for queries *pass*(*paul*) and *pass*(*mary*).

and Motik (2005) (referring to Lenzerini (2002) explain that in the GAV systems the problem is simply reduced to *unfolding* the views, since the reformulation is explicit in the mappings. In the LAV case, the problem requires more complex reasoning steps as in the case of sound mappings is not clear how to reformulate the concepts of a source ontology in terms of a global ontology. Therefore, in this work, we will restrict the case of LAV integration to complete views.

Definition 12. An *ontology integration system* $\mathscr{I}$ is a triple $(\mathscr{G}, \mathscr{S}, \mathscr{M})$ where:

- $\mathscr{G}$ is a *global ontology* expressed as a δ-ontology over an alphabet $\mathscr{A}_{\mathscr{G}}$.
- $\mathscr{S}$ is a *set of n source ontologies* $\mathscr{S}_1, \ldots, \mathscr{S}_n$ expressed as δ-ontologies over alphabets $\mathscr{A}_{\mathscr{S}_1}, \ldots, \mathscr{A}_{\mathscr{S}_n}$, resp. Each alphabet $\mathscr{A}_{\mathscr{S}_i}$ includes a symbol for each concept or role name of the source $\mathscr{S}_i$, $i = 1, \ldots, n$.
- $\mathscr{M}$ is a *set of n mappings* $\mathscr{M}_1, \ldots, \mathscr{M}_n$ between $\mathscr{G}$ and $\mathscr{S}_1, \ldots, \mathscr{S}_n$, resp. Each mapping $\mathscr{M}_i$ is constituted by a set of inclusion axioms of the form $q_{\mathscr{S}_i} \sqsubseteq q_{\mathscr{G}}$, where $q_{\mathscr{G}}$ and $q_{\mathscr{S}_i}$ are concept descriptions defined over the global ontology $\mathscr{G}$ and $\mathscr{S}_i$, $i = 1, \ldots, n$, resp. The concepts $q_{\mathscr{G}}$ are expressed over the alphabet $\mathscr{A}_{\mathscr{G}}$ and $q_{\mathscr{S}_i}$ are expressed over the alphabet $\mathscr{A}_{\mathscr{S}_i}$. The sets $\mathscr{M}_1, \ldots, \mathscr{M}_n$ are called *bridge ontologies*.

An ontology integration system will be interpreted as a DeLP program (See Fig. 7).

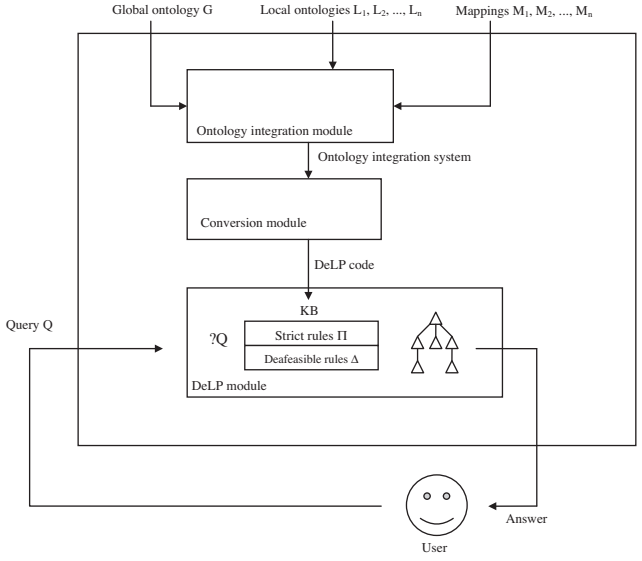

Fig. 7. The ONTOarg system: architecture.

Definition 13. Let $\mathscr{I} = (\mathscr{G}, \mathscr{S}, \mathscr{M})$ be an ontology integration system such that $\mathscr{S} = \{\mathscr{S}_1, \ldots, \mathscr{S}_n\}$ and $\mathscr{M} = \{\mathscr{M}_1, \ldots, \mathscr{M}_n\}$, where $\mathscr{G} = (T_S^{\mathscr{G}}, T_D^{\mathscr{G}}, A^{\mathscr{G}})$; $\mathscr{S}_i = \left(T_S^{\mathscr{S}_i}, T_D^{\mathscr{S}_i}, A_i^{\mathscr{S}_i}\right)$, and, $\mathscr{M}_i = \left(T_S^{\mathscr{M}_i}, T_D^{\mathscr{M}_i}\right)$, with $i = 1, \ldots, n$. The system $\mathscr{I}$ is interpreted as the DeLP program $\mathscr{I}_{DeLP} = (\Pi, \Delta)$, with:

$$\Pi = \left(\mathscr{T}_\Pi\left(T_S^{\mathscr{G}}\right)\right) \cup \left(\mathscr{T}_\Pi\left(A^{\mathscr{G}}\right)\right) \cup \left(\bigcup_{i=1}^{n} \mathscr{T}_\Pi\left(T_S^{\mathscr{S}_i}\right)\right) \cup \left(\bigcup_{i=1}^{n} \mathscr{T}_\Pi\left(T_S^{\mathscr{M}_i}\right)\right), \text{ and}$$

$$\Delta = \left(\mathscr{T}_\Delta\left(T_D^{\mathscr{G}}\right)\right) \cup \left(\bigcup_{i=1}^{n} \mathscr{T}_\Delta\left(T_D^{\mathscr{S}_i}\right)\right) \cup \left(\bigcup_{i=1}^{n} \mathscr{T}_\Delta\left(T_D^{\mathscr{M}_i}\right)\right).$$

Possible inferences in the integrated ontology $\mathscr{I}_{DeLP}$ are modeled by means of a dialectical analysis in the DeLP program that is obtained when each DL sentence of the ontology is mapped into DeLP clauses. Thus conclusions supported by warranted arguments will be the valid consequences that will be obtained from the original ontology, provided the strict information in $\mathscr{I}_{DeLP}$ is consistent. Formally:

Definition 14. Let $\mathscr{I} = (\mathscr{G}, \mathscr{S}, \mathscr{M})$ be an ontology integration system. Let a be an individual name, and C a concept name defined in $\mathscr{G}$.

1. Individual a is a *potential member* of C iff there exists an argument $\mathscr{A}$ for the literal $C(a)$ w.r.t. DeLP program $\mathscr{I}_{DeLP}$.
2. Individual a is a *justified member* of C iff there exists a warranted argument $\mathscr{A}$ for the literal $C(a)$ w.r.t. DeLP program $\mathscr{I}_{DeLP}$.
3. Individual a is an *strict member* of C iff there exists an empty argument for the literal $C(a)$ w.r.t. DeLP program $\mathscr{I}_{DeLP}$.

We will illustrate the above notions with an example. Notice that we label a concept C with the name of the ontology $\mathscr{S}_i$ to which it belongs (as in $\mathscr{S}_i : C$) following the XML name-space convention (this notation follows the one used in (Haase & Motik, 2005)).

Example 5. Let us consider the problem of assigning reviewers for papers. In Fig. 8, we present a global ontology $\mathscr{G}_5$ interpreted as: a professor with a postgraduate degree can be a reviewer; someone should not be a reviewer unless they are either a professor or have a graduate degree; however, a professor, despite not having a postgraduate degree, is accepted as a reviewer if he is an outstanding researcher. We also present local ontologies $\mathscr{L}_1$ and $\mathscr{L}_2$. $\mathscr{L}_1$ expresses that John is a professor who has a PhD, Paul is also a professor but has neither a PhD nor a MSc, Mary just has a MSc, and Steve is not a professor but has a MSc. The mapping $\mathscr{M}_{\mathscr{L}_1,\mathscr{G}}$ expresses that the terms MSc and PhD from local ontology $\mathscr{L}_1$ are contained in the term postgraduated in the global ontology, and that someone have neither a MSc nor a PhD is not a postgraduate. Ontology $\mathscr{L}_2$ expresses that a is an article, b a book,

Global ontology $\mathcal{G}_5 = (\emptyset, T_D^{\mathcal{G}}, \emptyset)$:

$$T_D^{\mathcal{G}} = \left\{ \begin{array}{l} \mathsf{Prof} \sqcap \mathsf{Postgrad} \sqsubseteq \mathsf{Reviewer}; \\ \neg\mathsf{Postgrad} \sqcup \neg\mathsf{Prof} \sqsubseteq \neg\mathsf{Reviewer}; \\ \mathsf{Prof} \sqcap \neg\mathsf{Postgrad} \sqcap \mathsf{Outstanding} \sqsubseteq \mathsf{Reviewer} \end{array} \right\}$$

Local ontology $\mathcal{L}_1 = (\emptyset, \emptyset, A^{\mathcal{L}_1})$:

$$A^{\mathcal{L}_1} = \left\{ \begin{array}{lll} \mathsf{JOHN} : \mathsf{Prof}; & \mathsf{JOHN} : \mathsf{Phd}; & \mathsf{PAUL} : \mathsf{Prof}; \\ \mathsf{PAUL} : \neg\mathsf{Msc}; & \mathsf{PAUL} : \neg\mathsf{Phd}; & \mathsf{MARY} : \mathsf{Msc}; \\ \mathsf{STEVE} : \neg\mathsf{Prof}; & \mathsf{STEVE} : \mathsf{Msc} & \end{array} \right\}$$

Mapping $\mathcal{M}_{\mathcal{L}_1,\mathcal{G}}$ between $\mathcal{L}_1$ and $\mathcal{G}_5$:

$$\mathcal{M}_{\mathcal{L}_1,\mathcal{G}} = \left\{ \begin{array}{l} \mathcal{L}_1 : (\mathsf{Msc} \sqcup \mathsf{Phd}) \sqsubseteq \mathcal{G} : \mathsf{Postgrad}; \\ \mathcal{L}_1 : (\neg\mathsf{Msc} \sqcap \neg\mathsf{Phd}) \sqsubseteq \mathcal{G} : \neg\mathsf{Postgrad} \end{array} \right\}$$

Local ontology $\mathcal{L}_2 = (\emptyset, \emptyset, A^{\mathcal{L}_2})$:

$$A^{\mathcal{L}_2} = \left\{ \begin{array}{ll} a : \mathsf{Article}; & b : \mathsf{Book}; \\ c : \mathsf{Chapter}; & \langle \mathsf{PAUL}, a \rangle : \mathsf{published}; \\ \langle \mathsf{PAUL}, b \rangle : \mathsf{published}; & \langle \mathsf{PAUL}, c \rangle : \mathsf{published} \end{array} \right\}$$

Mapping $\mathcal{M}_{\mathcal{L}_2,\mathcal{G}}$ between $\mathcal{L}_2$ and $\mathcal{G}_5$:

$$\mathcal{M}_{\mathcal{L}_2,\mathcal{G}} = \left\{ \begin{array}{l} \mathcal{L}_2 : (\exists\mathsf{published}.\mathsf{Article} \sqcap \exists\mathsf{published}.\mathsf{Book} \sqcap \\ \qquad \exists\mathsf{published}.\mathsf{Chapter}) \sqsubseteq \mathcal{G} : \mathsf{Outstanding} \end{array} \right\}$$

Fig. 8. LAV ontology integration system.

c a chapter and that Paul has published *a*, *b* and *c*. The mapping $\mathcal{M}_{\mathcal{L}_2,\mathcal{G}}$ expresses that the view corresponding to the individuals who have published an article, a chapter and a book corresponds to the set of outstanding researchers.

The interpretation of above ontologies in DeLP yields the code presented in Fig. 9. We show next the dialectical analyses that have to be performed to compute the justified membership of John, Paul, Mary and Steve to the concept "Reviewer" w.r.t. the ontology integration system $(\mathcal{G}, \{\mathcal{L}_1, \mathcal{L}_2\}, \{\mathcal{M}_{\mathcal{L}_1,\mathcal{G}}, \mathcal{M}_{\mathcal{L}_2,\mathcal{G}}\})$.

The individual John is a justified member of the concept "Reviewer" because the argument $\langle \mathcal{A}, reviewer(john) \rangle$ has no defeaters and is thus warranted (see Fig. 10. (a)), with:

$$\mathcal{A} = \left\{ \begin{array}{l} (reviewer(john) \prec postgrad(john), prof(john)), \\ (postgrad(john) \prec phd(john)) \end{array} \right\}.$$

In Paul's case, we conclude that he is a possible reviewer as he is also a justified member of the concept "Reviewer." Notice that Paul

DeLP program $(\emptyset, \Delta_{\mathcal{G}})$ obtained from $\mathcal{G}_5$:

$$\Delta_{\mathcal{G}} = \left\{ \begin{array}{l} reviewer(X) \multimap postgrad(X), prof(X). \\ \sim reviewer(X) \multimap \sim postgrad(X). \\ \sim reviewer(X) \multimap \sim prof(X). \\ reviewer(X) \multimap prof(X), \sim postgrad(X), outstanding(X). \end{array} \right\}$$

DeLP program $\mathcal{P}_1 = (\Pi^{\mathcal{L}_1}, \emptyset)$ obtained from $\mathcal{L}_1$:

$$\Pi^{\mathcal{L}_1} = \left\{ \begin{array}{lll} prof(john). & phd(john). & prof(paul). \\ \sim msc(paul). & \sim phd(paul). & msc(mary). \\ \sim prof(steve). & msc(steve). & \end{array} \right\}$$

Mapping $\mathcal{M}_{\mathcal{L}_1,\mathcal{G}}$ expressed in DeLP:

$$\Delta_{\mathcal{L}_1,\mathcal{G}} = \left\{ \begin{array}{l} \mathcal{G} : postgrad(X) \multimap \mathcal{L}_1 : msc(X). \\ \mathcal{G} : postgrad(X) \multimap \mathcal{L}_1 : phd(X). \\ \mathcal{G} : \sim postgrad(X) \multimap \mathcal{L}_1 : (\sim msc(X), \sim phd(X)). \end{array} \right\}$$

DeLP program $\mathcal{P}_2 = (\Pi^{\mathcal{L}_2}, \emptyset)$ obtained from $\mathcal{L}_2$:

$$\Pi^{\mathcal{L}_2} = \left\{ \begin{array}{ll} article(a). & book(b). \\ chapter(c). & published(paul, a). \\ published(paul, b). & published(paul, c). \end{array} \right\}$$

Mapping $\mathcal{M}_{\mathcal{L}_2,\mathcal{G}}$ expressed in DeLP:

$$\Delta_{\mathcal{L}_2,\mathcal{G}} = \left\{ \begin{array}{l} \mathcal{G} : outstanding(X) \multimap \mathcal{L}_2 : (\\ \quad published(X, Y), article(Y), \\ \quad published(X, Z), book(Z), \\ \quad published(X, W), chapter(W)). \end{array} \right\}$$

Fig. 9. Ontologies $\mathcal{G}_5$, $\mathcal{L}_1$ and $\mathcal{L}_2$ expressed in DeLP.

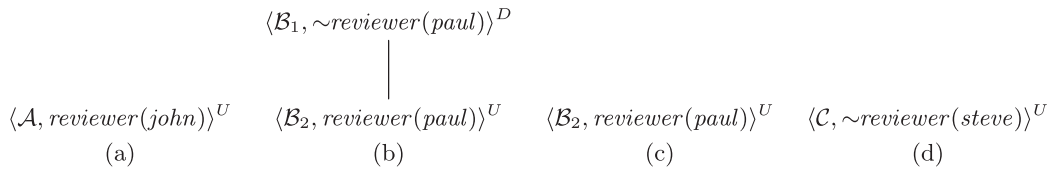

Fig. 10. Dialectical trees for *reviewer*(*john*), *reviewer*(*paul*) and *reviewer*(*steve*).

is a potential member of the concept "¬Reviewer" as there is an argument $\langle \mathscr{B}_1, \sim reviewer(paul)\rangle$, with:

$$\mathscr{B}_1 = \left\{ \begin{array}{l} (\sim reviewer(paul) \prec \sim postgrad(paul)), \\ (\sim postgrad(paul) \prec \sim msc(paul), \sim phd(paul)) \end{array} \right\}.$$

However, we see that there is another argument $\langle \mathscr{B}_2, reviewer(paul)\rangle$ that defeats $\mathscr{B}_1$, where:

$$\mathscr{B}_2 = \left\{ \begin{array}{l} (reviewer(paul) \prec prof(paul), \sim postgrad(paul), \\ outstanding(paul)), \\ (outstanding(paul) \prec \\ published(paul, a), article(a), \\ published(paul, b), book(b), \\ published(paul, c), chapter(c)), \\ (\sim postgrad(paul) \prec \sim msc(paul), \sim phd(paul)) \end{array} \right\}.$$

As $\mathscr{B}_2$ is undefeated, we conclude that the literal *reviewer*(*paul*) is warranted (see Fig. 10(b) and (c)).

Steve is not a reviewer as he is a justified member of the concept "¬Reviewer" (see Fig. 10. (d)). In this case, there exists a unique (undefeated) argument $\langle \mathscr{C}, \sim reviewer(steve)\rangle$. On the other hand, it is not possible to assess Mary's membership to the concept "Reviewer" as no arguments for *reviewer*(*mary*) nor ∼*reviewer*(*mary*) can be built.

5. Some properties of the approach

We now introduce some properties of the proposed approach to ontology integration presented above. Notice that the validity of the properties is strongly related to DeLP reasoning dynamics.

Property 1. *Let $\mathscr{I} = (\mathscr{G}, \mathscr{S}, \mathscr{M})$ be an ontology integration system. It cannot be the case that an individual a is a justified member of concepts C and ¬C simultaneously.*

Proof. Suppose that both *a* is a justified member of both *C* and ¬*C*. Then it must be the case that there exist two warranted arguments $\langle \mathscr{A}, C(a)\rangle$ and $\langle \mathscr{B}, \sim C(a)\rangle$. But this is impossible as DeLP cannot warrant two complementary literals (see (García & Simari, 2004)). □

Property 2. *Let $\mathscr{I} = (\mathscr{G}, \mathscr{S}, \mathscr{M})$ be an ontology integration system. It cannot be the case that an individual a is an strict member of concepts C and ¬C simultaneously.*

Proof. Suppose that $\mathscr{G} = \left(T_S^{\mathscr{G}}, T_D^{\mathscr{G}}, A^{\mathscr{G}}\right)$; $\mathscr{S} = \{\mathscr{S}_1, \ldots, \mathscr{S}_n\}$ with $\mathscr{S}_i = \left(T_i^S, T_i^D, A_i\right)$, and $\mathscr{M}_i = \left(T_{\mathscr{M}_i}^S, T_{\mathscr{M}_i}^D, A_{\mathscr{M}_i}\right)$, with $i = 1, \ldots, n$. Suppose that *a* is an strict member of both *C* and ¬*C*. Then there must exist two arguments $\langle \emptyset, C(a)\rangle$ and $\langle \emptyset, \sim C(a)\rangle$ w.r.t. $\mathscr{I}_{DeLP}$. Therefore

$$\mathscr{T}_\Pi(T_S^{\mathscr{G}}) \cup \bigcup_{i=1}^{n} (\mathscr{T}_\Pi\left(T_i^S\right) \cup \mathscr{T}_\Pi\left(T_{\mathscr{M}_i}^S\right) \cup \mathscr{T}_\Pi(A_i) \cup \mathscr{T}_\Pi(A_{\mathscr{M}_i}))$$

will be inconsistent contradicting the internal coherence assumption (see Remark 1). □

Property 3. *The running time for determining the strict, potential and justified membership of an individual to a class in an ontology integration system is finite.*

Proof sketch: Let $\mathscr{I} = (\mathscr{G}, \mathscr{S}, \mathscr{M})$ be an *ontology integration system*. As the δ-ontologies in $\mathscr{I}$ have a finite number of both named concepts and individual constants, the DeLP program $\mathscr{I}_{DeLP}$ is finite. Cecchi, Fillottrani, & Simari (2006) have shown that determining if there exists an argument for a literal is NP; besides, as the warrant procedure always builds a finite dialectical tree (García & Simari, 2004), then procesess for determining the strict, potential and justified membership always terminate.

6. Related work

Next we will review some recent research in reasoning with inconsistencies in ontologies, reasoning with ontologies in logic programming and recent advances in ontology integration, contrasting existing results with our approach.

6.1. Inconsistency treatment in ontologies

To the best of our knowledge, the treatment of DL ontologies in Prolog is attributed to (Grosof et al., 2003). They show how to interoperate, semantically and inferentially, between the leading Semantic Web approaches to rules (RuleML Logic Programs) and ontologies (OWL DL) by analyzing their expressive intersection. They define a new intermediate knowledge representation called Description Logic Programs (DLP), and the closely related Description Horn Logic (DHL) which is an expressive fragment of FOL. They show how to perform the translation of premises and inferences from the DLP fragment of DL to logic programming. Part of our approach is based on Grosof et al.'s work as the algorithm for translating DL ontologies into DeLP is based on it. However, as Grosof et al. (2003) use standard Prolog rules, they are not able to deal with inconsistent DL knowledge bases as our proposal does.

The treatment of inconsistency in Description Logics ontologies is not new since it has been addressed in related non-monotonic approaches including the addition of a preference order on the axioms (Heymans & Vermeir, 2002), imposing answer set programming rules on top of ontologies (Eiter, Lukasiewicz, Schindlauer, & Tompits, 2004), using Paraconsistent Logics (Huang, van Harmelen, & ten Teije, 2005) or Belief Revision (Wasserman, 1989) to determine a consistent subset of an inconsistent ontology. In Heymans & Vermeir (2002), the authors extend the DL $\mathscr{SHOQ}(D)$ with a preference order on the axioms. With this strict partial order certain axioms can be overruled, if defeated with more preferred ones. They also impose a preferred model semantics, introducing nonmonotonicity into $\mathscr{SHOQ}(D)$. Similarly as in (Heymans & Vermeir, 2002) we allow to perform inferences from inconsistent ontologies by considering subsets (arguments) of the original ontology. (Heymans & Vermeir, 2002) also impose a hard-coded comparison criterion on DL

axioms. In our work, the system, and not the programmer, decides which DL axioms are to be preferred as we use specificity as argument comparison criterion. We think that our approach can be considered more declarative in this respect. In particular the comparison criterion in DeLP is modular, so that rule comparison could also be adopted (García & Simari, 2004).

Eiter et al. (2004) propose a combination of logic programming under the answer set semantics with the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$. This combination allows for building rules on top of ontologies. In contrast to our approach, they keep separated rules and ontologies and handle exceptions by codifying them explicitly in programs under answer set semantics.

Huang et al. (2005) use paraconsistent logics to reason with inconsistent ontologies. They use a selection function to determine which consistent subsets of an inconsistent ontology should be considered in the reasoning process. In our approach given an inconsistent ontology Σ, we consider the set of warranted arguments from $\mathcal{T}(\Sigma)$ as the valid consequences.

Williams & Hunter (2007) use argumentation to reason with possibly inconsistent rules on top of DL ontologies. In contrast, we translate possible inconsistent DL ontologies to DeLP to reason with them within DeLP. Laera, Tamma, Euzenat, Bench-Capon, & Payne (2006) propose an approach for supporting the creation and exchange of different arguments, that support or reject possible correspondences between ontologies in the context of a multi-agent system. In our work we assume correspondences between ontologies as given.

Antoniou & Bikakis (2007) propose a rule-based approach to defeasible reasoning based on a translation to logic programming with declarative semantics that can reason with rules, RDF (S) and parts of OWL ontologies. RDF data of the form rdf (Subject, Predicate, Object) is translated as Prolog facts of the form Predicate (Subject,Object). They also define Prolog rules for processing RDF Schema information (*e.g.*, C (X):- rdf:type (X,C)) for modeling the type construct). All of the rules are created at compile time before actual querying takes place. To the best of our knowledge, in (Antoniou & Bikakis, 2007), the authors' approach distinguishes between strict and defeasible and possess flexibility for defining different priority relations between arguments. As our approach is based on DeLP, it also distinguishes between strict and defeasible rules, and it also allows to replace the comparison criterion between arguments in a modular way. Antoniou and Bikakis translate the semantics of their argumentation system into Prolog, the semantics of OWL sentences into Prolog, the semantics of RuleML sentences into Prolog and then they perform query processing. We translate OWL into DL and the into DeLP, thus keeping separated the knowledge representation (in DeLP) from the query processing (performed into the JAM).

Horridge, Parsia, and Sattler (2008) present a *justification-based* approach to reasoning with ontologies. In that context, a justification for an entailment in an OWL ontology is a minimal subset of the ontology that is sufficient for that entailment to hold. Besides they are able to find *laconic justifications* that provide *minimal* axioms supporting an entailment, that can also be used to pinpoint the cause of inconsistencies. The notion of justification for an entailment in Horridge et al. (2008) is similar to the notion of argument for a literal in our work as an argument is made up of a minimal subset of the defeasible information in a DeLP program along with the strict information that allows to derive a defeasible conclusion (see Definition 2), thus providing a sort of justification for an entailment to hold. However, in the case of inconsistencies DeLP is not only able to consider all the arguments for a literal, but also the defeat relation holding among those arguments in order to determine what the valid consequences of a δ-ontology are (characterized as the warranted arguments).

6.2. Ontology integration with inconsistent ontologies

The approach to ontology integration presented in this work is an extension to Calvanese's framework (Calvanese et al., 2001), which in turn can be considered an instance of the more general framework for data integration presented in Lenzerini (2002). Unlike Calvanese's approach, our proposal is able to deal with inconsistent ontologies.

Imam, MacCaull, and Kennedy (2007) discuss various implementation issues for the development of a prototype merging system which will provide an inconsistency-tolerant reasoning mechanism applicable to the healthcare domain. Their approach, which is based on paraconsistent logic, is similar to ours in the sense that does not lose information but instead is semi-automatic as opposed to ours which is fully automated (except for the calculation of the mappings between ontologies that we assume as given).

Next we show how our approach behaves with one of the case studies presented in Imam et al. (2007). Consider the two ontologies Σ_1 and Σ_2 presented in (Imam et al., 2007, Section 2), where:

$$\Sigma_1 = \left\{ \begin{array}{l} \mathsf{Brain} \sqsubseteq \mathsf{CentralNervousSystem} \\ \mathsf{Brain} \sqsubseteq \mathsf{BodyPart} \\ \mathsf{CentralNervousSystem} \sqsubseteq \mathsf{NervousSystem} \end{array} \right\}, \quad \text{and}$$

$$\Sigma_2 = \left\{ \begin{array}{l} \mathsf{Brain} \sqsubseteq \mathsf{CentralNervousSystem} \\ \mathsf{CentralNervousSystem} \sqsubseteq \mathsf{NervousSystem} \\ \mathsf{BodyPart} \sqsubseteq \neg\mathsf{NervousSystem} \end{array} \right\}.$$

Intuitively ontologies Σ_1 and Σ_2 express that the brain is part of the central nervous system, the brain is a body part, and the central nervous system is part of the nervous system, but a body part cannot be included in the nervous system. According to (Imam et al., 2007), ontologies Σ_1 and Σ_2 are merged as the single ontology Σ, where:

$$\Sigma = \left\{ \begin{array}{l} \mathsf{Brain} \sqsubseteq \mathsf{CentralNervousSystem} \\ \mathsf{Brain} \sqsubseteq \mathsf{BodyPart} \\ \mathsf{CentralNervousSystem} \sqsubseteq \mathsf{NervousSystem} \\ \mathsf{BodyPart} \sqsubseteq \neg\mathsf{BodyPart} \end{array} \right\}.$$

Merging the two ontologies above would imply the contradiction that Brain is a body part and not a body part. This kind of inconsistency is known as ontological inconsistency, where a concept is asserted as subclass of multiple disjoint concepts. In our work, according to the translation approach from DL to DeLP, this ontology Σ along with an assertion "b:Brain" is interpreted as the DeLP program $\mathcal{P}$, where:

$$\mathcal{P} = \left\{ \begin{array}{l} centralNervousSystem(X) \prec brain(X). \\ bodyPart(X) \prec brain(X). \\ nervousSystem(X) \prec centralNervousSystem(X). \\ \sim nervousSystem(X) \prec bodyPart(X). \\ brain(b). \end{array} \right\}.$$

In this case two arguments $\langle \mathcal{A}, nervousSystem(b) \rangle$ and $\langle \mathcal{B}, \sim nervousSystem(b) \rangle$ can be built, where:

$$\mathcal{A} = \left\{ \begin{array}{l} (nervousSystem(b) \prec centralNervousSystem(b)), \\ (centralNervousSystem(b) \prec brain(b)) \end{array} \right\}, \text{ and}$$

$$\mathcal{B} = \left\{ \begin{array}{l} (\sim nervousSystem(b) \prec bodyPart(b)), \\ (bodyPart(b) \prec brain(b)) \end{array} \right\}.$$

As these two arguments are equi-preferred in DeLP, the answer to the query $nervousSystem(b)$ is *Undecided*. Therefore our system cannot determine the membership of the individual b to the concept NervousSystem.

7. Conclusions

We have presented an approach for performing local-as-view integration of Description Logic ontologies when these ontologies can be potentially inconsistent. We have adapted the notion of ontology integration system of Calvanese et al. (2001) for making it suitable for the δ-ontology framework, presenting both formal definitions and a case study.

For reasoning with an inconsistent ontology, it can be repaired manually by the knowledge engineer or automatically (*e.g.*, using Belief Revision (Ribeiro & Wassermann, 2009)). Other approaches involve using some non-monotonic inference procedure to obtain meaningful answers (*e.g.*, paraconsistent logics (Huang et al., 2005) which consider reasoning with a consistent subset of the inconsistent ontology). Our approach directly copes with inconsistency by reasoning within the framework of DeLP, thus relying in defeasible argumentation for determining the membership status of individuals to concepts. It must be noted that the proposed approach is only useful in the case of complete mappings, and therefore the case for local-as-view integration with sound and exact mappings remains as an open problem and is part of our current research efforts.

Acknowledgements

The research was funded by LACCIR Project 1211LAC004, by PIP CONICET Projects 112-200801-02798 and 112-200901-00863, and by Sec. Gral. de Ciencia y Tecnología, Universidad Nacional del Sur.

References

Antoniou, G., & Bikakis, A. (2007). DR-Prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering, 19*(2), 233–245.

Antoniou, G., Billington, D., & Maher, M. (1998). Normal forms for defeasible logic. In *Proceedings of international joint conference and symposium on logic programming* (pp. 160–174). MIT Press.

Antoniou, G., Maher, M. J., & Billington, D. (2000). Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming, 42*, 47–57.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (Eds.). (2003). The description logic handbook – Theory, implementation and applications. Cambridge University Press.

Bench-Capon, T. J. M., & Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial Intelligence, 171*, 619–641.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The Semantic Web', Scient. American.

Brena, R., Chesñevar, C., & Aguirre, J. (2006). Argumentation-supported information distribution in a multiagent system for knowledge management. In *Proceedings of ArgMAS 2005 (Utrecht, Netherlands, July 2005) In LNCS 4049* (pp. 279–296). Springer-Verlag.

Brewka, G., Dix, J., & Konolige, K. (1997). *Non monotonic reasoning. An overview*. Stanford, USA: CSLI Publications.

Calvanese, D., Giacomo, G.D. & Lenzerini, M. (2001). A framework for ontology integration. In *First semantic web working symposium* (pp. 303–316).

Caminada, M. (2008). On the issue of contraposition of defeasible rules. In P. Besnard, S. Doutre, & A. Hunter (Eds.), *COMMA. Frontiers in Artificial Intelligence and Applications* (Vol. 172, pp. 109–115). IOS Press.

Carbogim, D., Robertson, D., & Lee, J. (2000). Argument-based applications to knowledge engineering. *The Knowledge Engineering Review, 15*(2), 119–149.

Cecchi, L., Fillottrani, P. & Simari, G. (2006). On complexity of DeLP through game semantics. In J. Dix & A. Hunter, (Eds.). *11th International workshop on nonmonotonic reasoning* (pp. 386–394).

Chesñevar, C. & Maguitman, A., (2004a). An argumentative approach to assessing natural language usage based on the web corpus. In *Proceedings of the 16th ECAI Conference, Valencia, Spain* (pp. 581–585).

Chesñevar, C. & Maguitman, A., (2004b). ARGUENET: An Argument-Based Recommender System for Solving Web Search Queries. In *Proceedings of the 2nd IEEE International IS-2004 Conference* (pp. 282–287) Varna, Bulgaria.

Chesñevar, C., Brena, R., & Aguirre, J. (2005a). Knowledge distribution in large organizations using defeasible logic programming. In *Proceedings of the 18th Canadian conference on AI (LNCS, Vol. 3501)* (pp. 244–256). Springer-Verlag.

Chesñevar, C., Brena, R. & Aguirre, J., (2005b). Modelling power and trust for knowledge distribution: An argumentative approach. In *LNAI Springer Series Proceedings of the 3rd Mexican International Conference on Artificial Intelligence – MICAI 2005, vol. 3789* (pp. 98–108).

Chesñevar, C., Maguitman, A., & Loui, R. (2000). Logical models of argument. *ACM Computing Surveys, 32*(4), 337–383.

Chesñevar, C., Maguitman, A., & Simari, G. (2006). Argument-based critics and recommenders: A qualitative perspective on user support systems. *Data & Knowledge Engineering (DKE), 59*(2), 293–319.

Chesñevar, C., Simari, G., Alsinet, T. & Godo, L., 2004. A logic programming framework for possibilistic argumentation with vague knowledge. In *Proceedings of the international conference in uncertainty in artificial intelligence (UAI 2004)*, Banff, Canada (pp. 76–84).

Chesñevar, C., Simari, G., Godo, L. & Alsinet, T., 2005. Argument-based expansion operators in possibilistic defeasible logic programming: Characterization and logical properties. In *LNAI/LNCS Springer Series, Vol. 3571 (Proceedings of the 8th ECSQARU Intl. Conference, Barcelona, Spain)* (pp. 353–365).

Dimopoulos, Y., & Kakas, A. (1995). Logic programming without negation as failure. In J. Lloyd (Ed.), *Logic programming* (pp. 369–383). Cambridge, MA: MIT Press.

Eiter, T., Lukasiewicz, T., Schindlauer, R. & Tompits, H., 2004. Combining Answer Set Programming with Description Logics for the Semantic Web. KR 2004 (pp. 141–151).

García, A., & Simari, G. (2004). Defeasible logic programming an argumentative approach. *Theory and Practice of Logic Programming, 4*(1), 95–138.

Gómez, S. & Chesñevar, C., 2004. A hybrid approach to pattern classification using neural networks and defeasible argumentation. In *Proceedings of 17th international FLAIRS conference Miami, Florida, USA, American association for artificial intelligence* (pp. 393–398).

Gómez, S. A., Chesñevar, C. I., & Simari, G. R. (2008). Defeasible reasoning in web forms through argumentation. *International Journal of Information Technology & Decision Making, 7*, 71–101.

Gómez, S., Chesñevar, C., & Simari, G. (2010). Reasoning with inconsistent ontologies through argumentation. *Applied Artificial Intelligence, 1*(24), 102–148.

Grosof, B., Horrocks, I., Volz, R. & Decker, S., 2003. Logic programs: Combining logic programs with description logics. In *WWW2003*, May 20–24, Budapest, Hungary.

Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition, 5*(2), 199–220.

Haarslev, V. & Möller, R., 2001. RACER System Description, Technical report, University of Hamburg, Computer Science Department.

Haase, P., & Motik, B. (2005). A mapping system for the integration of OWL-DL ontologies. In A. Hahn, S. Abels, & L. Haak (Eds.), *IHIS 05: Proceedings of the first international workshop on interoperability of heterogeneous information systems* (pp. 9–16). ACM Press.

Heymans, S. & Vermeir, D., 2002. A Defeasible Ontology Language, CoopIS/DOA/ODBASE 2002 (pp. 1033–1046).

Horridge, M., Parsia, B. & Sattler, U., 2008. Laconic and Precise Justifications in OWL, In '*Proceedings of the international semantic web conference (ISWC 2008)*'.

Huang, Z., van Harmelen, F. & ten Teije, A., 2005, Reasoning with Inconsistent Ontologies. In L.P. Kaelbling & A. Saffiotti, (Eds.), *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, Edinburgh, Scotland (pp. 454–459).

Imam, F., MacCaull, W. & Kennedy, M., 2007. Merging healthcare ontologies: Inconsistency tolerance and implementation issues In *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS '07)* (pp. 74–85).

Janjua, N. K., & Hussain, F. K. (2012). Web@idss argumentation-enabled web-based idss for reasoning over incomplete and conflicting information. *Knowledge-Based Systems, 32*(0), 9–27.

Kakas, A. C., & Toni, F. (1999). Computing argumentation in logic programming. *Journal of Logic and Computation, 9*(4), 515–562.

Kakas, A. C., Mancarella, P., & Dung, P. M. (1994). The acceptability semantics for logic programs. In *Proceedings of the 11th international conference on logic programming* (pp. 504–519). Santa Margherita, Italy: MIT Press.

Klein, M., 2001. Combining and relating ontologies: An analysis of problems and solutions. In A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt & M. Uschold (Eds.), *Workshop on ontologies and information sharing, IJCAI'01*, Seattle, USA.

Laera, L., Tamma, V., Euzenat, J., Bench-Capon, T., & Payne, T. (2006). Reaching agreement over ontology alignments. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA. Lecture Notes in Computer Science* (Vol. 4273). Berlin/ Heidelberg: Springer.

Lenzerini, M., 2002. Data integration: A theoretical perspective, In *Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2002*, Madison, Winsconsin, USA.

Lloyd, J. (1987). *Foundations of logic programming*. Springer-Verlag.

Modgil, S., Toni, F., Bex, F., Bratko, I., Chesñevar, C., Dvořák, W., et al. (2013). The added value of argumentation: examples and challenges. In O. Sascha (Ed.). *Handbook of agreement technologies* (Vol. 8). New York, USA: Springer Verlag.

Nute, D. (1988). Defeasible reasoning. In J. H. Fetzer (Ed.), *Aspects of artificial intelligence* (pp. 251–288). Norwell, MA: Kluwer Academic Publishers..

Nute, D. (1992). Basic defeasible logic. In L. Fariñas del Cerro (Ed.), *Intensional logics for programming*. Oxford: Clarendon Press.

Parsia, B. & Sirin, E., 2004. Pellet: An OWL DL Reasoner. In *3rd International semantic web conference (ISWC2004)*.

Parsons, S., Sierrra, C., & Jennings, N. (1998). Agents that reason and negotiate by arguing. *Journal of Logic and Computation, 8*, 261–292.

Pollock, J. (1974). *Knowledge and justification*. Princeton.

Pollock, J. L. (1987). Defeasible reasoning. *Cognitive Science, 11*, 481–518.

Pollock, J. L. (1995). *Cognitive carpentry: A blueprint for how to build a person*. Bradford/MIT Press.

Prakken, H., & Sartor, G. (2002). The role of logic in computational models of legal argument – A critical survey. In A. Kakas & F. Sadri (Eds.), *Computational logic: logic programming and beyond* (pp. 342–380). Springer.

Prakken, H., & Vreeswijk, G. (2002). Logical systems for defeasible argumentation. In D. Gabbay & F. Guenther (Eds.), *Handbook of philosophical logic* (pp. 219–318). Kluwer Academic Publishers.

Rahwan, I., & Simari, G. R. (2009). *Argumentation in artificial intelligence*. Springer.

Rahwan, I., Ramchurn, S. D., Jennings, N. R., Mcburney, P., Parsons, S., & Sonenberg, L. (2003). Argumentation-based negotiation. *Knowledge Engineering Review, 18*(4), 343–375.

Ribeiro, M. M., & Wassermann, R. (2009). Base revision for ontology debugging. *Journal of Logic and Computation, 19*(5), 721–743.

Sierra, C., & Noriega, P. (2002). Agent-mediated interaction from auctions to negotiation and argumentation. In *Foundations and applications of multi-agent systems. LNCS series* (Vol. 2403, pp. 27–48). Springer.

Simari, G., & Loui, R. (1992). A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence, 53*, 125–157.

Stolzenburg, F., García, A., Chesñevar, C., & Simari, G. (2003). Computing generalized specificity. *Journal of Applied Non-Classical Logics, 13*(1), 87–113.

Verheij, B. (2005). *Virtual arguments. On the design of argument assistants for lawyers and other arguers*. The Hague: Asser Press.

Wasserman, P. D. (1989). *Neural computing. Theory and practice*. Van Nostrand Reinhold.

Williams, M. & Hunter, A., 2007. Harnessing ontologies for argument-based decision-making in breast cancer. In *Proceedings of the international conference on tools with AI (ICTAI'07)* (pp. 254–261).

Zhang, P., Sun, J., & Chen, H. (2005). Frame-based argumentation for group decision task generation and identification. *Decision Support Systems, 39*, 643–659.