Ontology Merging Using Belief Revision and Defeasible Logic Programming[⋆]

Sergio Alejandro Gómez and Guillermo Simari

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

Abstract. We combine argumentation, belief revision and description logic ontologies for extending the δ-ontologies framework to show how to merge two ontologies in which the union of the strict terminologies could lead to inconsistency. To solve this problem, we revisit a procedure presented by Falappa *et al.* in which part of the offending terminologies are turned *defeasible* by using a kernel revision operator.

1 Introduction

The confluence of Description Logics and argumentation is an important research topic as shown by the ever growing list of publications that can be found on the subject [1–5]. Description Logics [6] constitute the semantic substrate of the Web Ontology Language OWL [7], which is at the core of the Semantic Web initiative. The Semantic Web [8] is a vision of the current Web where resources have exact meaning assigned in terms of knowledge bases called *ontologies* [9], enabling agents to reason about them. Argumentation [10, 11] is a form of non-monotonic reasoning that allows to obtain consequences from possibly inconsistent knowledge bases. On the other hand, belief revision is the process of changing beliefs to take into account a new piece of information; in spite of the union of argumentation and belief revision not being new it can be regarded as a live research topic (see [12–14]).

In [1], Gómez *et al.* developped a framework called δ-ontologies that allows to reason in the presence of inconsistent description logic ontologies by using Defeasible Logic Programming [15], which is an argumentative framework based on logic programming. In a δ-ontology the terminology defining the vocabulary is separated in strict and defeasible, the former is inconsistency free but the latter could be not. While joining two defeasible terminologies is trivial, joining two strict terminologies can lead to inconsistency if done careless. In this article,

⋆ This research is funded by the Project *Representación de Conocimiento y Razonamiento Argumentativo: Herramientas Inteligentes para la Web y las Bases de Datos Federadas (24/N030)*, Agencia Nacional de Promoción Científica y Tecnológica and Universidad Nacional del Sur.

we combine argumentation, belief revision and description logic ontologies for extending the δ-ontologies framework to show how to merge two ontologies in which the union of the strict terminologies could lead to inconsistency. To solve this problem, we revisit a procedure presented by Falappa *et al.* [12] in which part of the offending terminologies are turned *defeasible* by using a kernel revision operator.

The rest of this paper is structured as follows. In Section 2 we briefly present the fundamentals of Description Logics and Defeasible Logic Programming. Section 3 briefly recalls the framework of δ-ontologies for reasoning with possibly inconsistent ontologies. In Section 4, we extend the δ-ontologies framework to allow for merging strict terminologies while conserving consistency. Finally Section 5 concludes.

2 Background

2.1 Description Logics

Description Logics (DL) are a well-known family of knowledge representation formalisms [6]. They are based on the notions of *concepts* (unary predicates, classes) and *roles* (binary relations), and are mainly characterized by the constructors that allow complex concepts and roles to be built from atomic ones. Let C and D stand for concepts and R for a role name. Concept descriptions are built from concept names using the constructors conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), negation ($\neg C$), existencial 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 to the basic DL are possible including inverse and transitive roles noted as P^- and P^+, resp.

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 $\langle a, b\rangle : R$, where C is a concept description, R is a role name, and a and b are individual names.

2.2 Defeasible Logic Programming

Defeasible Logic Programming (DeLP) [15] provides a language for knowledge representation and reasoning that uses *defeasible argumentation* [10] to decide between contradictory conclusions through a *dialectical analysis*. Codifying knowledge by means of a DeLP program provides a good trade-off between expressivity and implementability for dealing with incomplete and potentially contradictory information. In a defeasible logic program $\mathcal{P} = (\Pi, \Delta)$, 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. An *argument* $\langle \mathcal{A}, H \rangle$ is a minimal non-contradictory set of ground defeasible clauses $\mathcal{A}$ of Δ that allows to derive a ground literal H possibly using ground rules of Π. Since arguments may be in conflict (concept captured in terms of a logical contradiction), an attack relationship between arguments can be defined. A criterion is usually defined to decide between two conflicting arguments. If the attacking argument is strictly preferred over the attacked one, then it is called a *proper defeater.* If no comparison is possible, or both arguments are equi-preferred, the attacking argument is called a *blocking defeater.* In order to determine whether a given argument $\mathcal{A}$ is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for $\mathcal{A}$, defeaters for these defeaters, and so on, are taken into account. Given a DeLP program $\mathcal{P}$ and a query H, the final answer to H w.r.t. $\mathcal{P}$ takes such dialectical analysis into account. The answer to a query can be: *yes*, *no*, *undecided*, or *unknown.*

3 Reasoning with Inconsistent Ontologies in DeLP

In the presence of inconsistent ontologies, traditional DL reasoners issue an error message and the knowledge engineer must then debug the ontology (*i.e.* making it consistent). In [1], Gómez *et al.* showed how DeLP can be used for coping with inconsistencies in ontologies such that the task of dealing with them is automatically solved by the reasoning system. We recall some of the concepts for making the article more self-contained.

Definition 1 (δ-Ontology). *Let C be an $\mathcal{L}_b$-class, D an $\mathcal{L}_h$-class, A, B $\mathcal{L}_{hb}$-classes, P, Q properties, a, b individuals. Let T be a set of inclusion and equality sentences in $\mathcal{L}_{DL}$ 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 1. Figure 1 presents a very simple δ-ontology $\Sigma = (\emptyset, T_D^1, A^1)$ that expresses that every man is a mortal unless he is a Highlander. Socrates is a man and McLeod is both a man and a Highlander.

For assigning semantics to a δ-ontology, two translation functions $\mathcal{T}_\Delta$ and $\mathcal{T}_\Pi$ from DL to DeLP are defined based on the work of [16] (for details see [1]).

Definition 2. ($\mathcal{T}_\Pi^*$ mapping from DL sentences to DeLP strict rules) *Let A, C, D be concepts, X, Y variables, P, Q properties. The $\mathcal{T}_\Pi^* : 2^{\mathcal{L}_{DL}} \rightarrow 2^{\mathcal{L}_{DeLP_\Pi}}$ mapping 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 "$H \leftarrow B_2$".*

Ontology $\Sigma_1 = (\emptyset, T_D^1, A^1)$:

Defeasible terminology T_D^1:	**Assertional box A^1:**
$\mathsf{Man} \sqsubseteq \mathsf{Mortal}$	SOCRATES : Man
$\mathsf{Man} \sqcap \mathsf{Highlander} \sqsubseteq \neg\mathsf{Mortal}$	MCLEOD : Man
	MCLEOD : Highlander

Fig. 1. A very simple δ-ontology

Definition 3 (Transposes of a strict rule). *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 "$\mathfrak{Trans}(r)$", is defined as:*

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

Definition 4 ($\mathcal{T}_\Pi$ mapping from DL sentences to DeLP strict rules). *The mapping from DL ontologies into DeLP strict rules is defined as $\mathcal{T}_\Pi(T) = \mathfrak{Trans}(\mathcal{T}_\Pi^*(T))$.*

Definition 5 (Interpretation of a δ-ontology). *Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology. The* interpretation *of Σ is a DeLP program $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.*

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 $\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A)$.

Definition 6. (Potential, justified and strict membership of an individual to a class) *Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology, C a class name, a an individual, and $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.*

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

Example 2 (Continues Ex. 1). Consider again the δ-ontology Σ_1, it is interpreted as the DeLP program $\mathcal{P}_1$ according to Def. 5 as shown in Fig. 3. From $\mathcal{P}_1$, we

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

Fig. 2. Mapping from DL ontologies to DeLP strict rules

can determine that Socrates justifiedly belongs to the concept Mortal in Σ_1 as there exists a warranted argument structure $\langle \mathcal{A}_1, \mathit{mortal}(\mathit{socrates})\rangle$ where:

$$\mathcal{A}_1 = \left\{ \mathit{mortal}(\mathit{socrates}) \prec\!\!\!- \mathit{man}(\mathit{socrates}) \right\}.$$

Likewise, we can determine that Mcleod justifiedly belongs to the concept $\neg\mathsf{Mortal}$ in Σ_1. We can see that Mcleod potentially belongs to Mortal, as in the case of Socrates, for there is an argument $\langle \mathcal{B}_1, \mathit{mortal}(\mathit{mcleod})\rangle$ where

$$\mathcal{B}_1 = \left\{ \mathit{mortal}(\mathit{mcleod}) \prec\!\!\!- \mathit{man}(\mathit{mcleod}) \right\}.$$

This argument $\mathcal{B}_1$ is however defeated by $\langle \mathcal{B}_1, {\sim}\mathit{mortal}(\mathit{mcleod})\rangle$, where

$$\mathcal{B}_2 = \left\{ {\sim}\mathit{mortal}(\mathit{mcleod}) \prec\!\!\!- \mathit{man}(\mathit{mcleod}), \mathit{highlander}(\mathit{mcleod}) \right\}.$$

Notice that as $\mathcal{B}_2$ is *more specific* (see [17] for details) than $\mathcal{B}_1$, $\mathcal{B}_1$ cannot defeat $\mathcal{B}_2$.

4 Ontology Merging based on Belief Revision Theory

We now extend the δ-ontologies framework to allow for ontology merging based on belief revision. First the fundamentals of belief revision are presented. Next we present the actual extension along with a running example.

DeLP program $\mathcal{P}_1 = (\Pi_1, \Delta_1)$ obtained from Σ_1:

Facts Π_1:	**Defeasible rules Δ_1:**
$man(socrates).$	$mortal(X) \mathrel{-\!\!\!\prec} man(X).$
$man(mcleod).$	$\sim mortal(X) \mathrel{-\!\!\!\prec} man(X), highlander(X).$
$highlander(mcleod).$	

Fig. 3. DeLP program $\mathcal{P}_1$ obtained from ontology Σ_1

4.1 Fundamentals of Belief Revision

Belief revision is the process of changing beliefs to take into account a new piece of information. Two kinds of changes are usually distinguished [18, 19]: *update*, in which new information must be considered with respect to a set of old beliefs, then update refers to the operation of changing the old beliefs to take into account the change; and *revision*, where there are old beliefs and new information; in this case the new information is considered more reliable, then revision is the process of inserting the new information into the set of old beliefs without generating an inconsistency. Belief revision should produce minimal change, *i.e.* the knowledge before and after the change should be as similar as possible.

Given two set of sentences K and A, a *revision operator* $K \circ A$ is a function that maps K and A to a new set of sentences. In particular, in Falappa *et al.* [12] the mechanism of a revision operator $K \circ A$ by a set of sentences with partial acceptance is defined as follows: first, the input set A is initially accepted, and, second, all posible inconsistencies of $K \cup A$ are removed. The mechanism of this operator consists of adding A to K and then eliminating from the result all possible inconsistencies by means of an incision function that makes a “cut” over each minimally inconsistent subset of $K \cup A$.

In [12], beliefs are split into two distinguished sets: (i) *particular beliefs* K_P, that are represented by ground facts, and (ii) *general beliefs* K_G, that are represented by closed material implications. Thus, each belief base K has the form $K_P \cup K_G$ where $K_P \cap K_G = \emptyset$. When doing a kernel revision by a set of sentences, an incision function is needed to make a cut upon every set; *i.e.* it is necessary to determine which beliefs must be given up in the revision process. They consider two possible policies: discard particular beliefs, and discard general beliefs. In the latter, at least one sentence is discarded. Thus [12] propose a refined characterization of revision by preserving retracted beliefs with a different status: retracted general beliefs are preserved as *defeasible rules*. They also introduce a revision operator that generates defeasible conditionals from a revision operator upon belief bases represented in a first order language. It may be the case that in the revision process a conditional sentence of the form $(\forall(X))(\alpha(X) \rightarrow \beta(X))$ has to to be eliminated. This can occur because new incoming information results in an inconsistency. One of the following cases may occur: (i) there exists some individual satisfying α but not satisfying β, and (ii) there exists some individual satisfying $\neg\beta$ but not satisfying $\neg\alpha$. Eliminating $(\forall(X))(\alpha(X) \rightarrow \beta(X))$ from the knowledge base produces too much loss of information. As an alternative, Falappa *et al.* propose a transformation to change it into $\beta \prec \alpha$. Formally:

Definition 7 (Positive/negative transformation [12]). *Let* $\delta = (\forall X_1 \ldots X_n)(\alpha \rightarrow \beta)$ *be a material implication in DeLP. A positive transformation of* δ*, noted by* $T^+(\delta)$*, is a sentence of the form* $\beta \prec \alpha$*; a negative transformation of* δ*, noted by* $T^-(\delta)$*, is a sentence of the form* $\neg\beta \prec \neg\alpha$*.*

Definition 8 (Kernel (partial meet) composed revision [12]). *Let* (K, Δ) *be a knowledge structure,*[1] $(\circ)$ *an operator of kernel (partial meet) revision by a set of sentences for* K *and* A *a set of sentences. The* kernel (partial meet) composed revision *of* (K, Δ) *w.r.t.* A *is defined as:* $(K, \Delta) \star A = (K', \Delta')$ *such that* $K' = K \circ A$ *and* $\Delta' = \Delta \cup \Delta'_1 \cup \Delta'_2$ *where:*

$$\Delta'_1 = \{ \alpha \prec true | \alpha \in (K_P \setminus K \circ A) \}$$
$$\Delta'_2 = \{ T^+(\alpha) | \alpha \in (K_G \setminus K \circ A) \} \cup \{ T^-(\alpha) | \alpha \in (K_G \setminus K \circ A) \}.$$

The set K' contains the revised undefeasible beliefs, Δ'_1 is the transformation in defeasible rules of particular beliefs (also called *presumptions* [15, Section 6]) eliminated from K whereas Δ'_2 is the transformation of general beliefs eliminated from K into defeasible rules.

4.2 Merging of δ-ontologies using Belief Revision

We now adapt the reasoning framework for δ-ontologies to use it in ontology merging. Merging is the process of creating a new ontology from two or more existing ontologies with overlapping parts [20]. Suppose we have two *strict* ontologies[2] that we desire to reason with at the same time. However accepting both ontologies at once may generate inconsistencies. The simplest solution is two consider the ontologies as *defeasible information.* This solution is too simplistic, a smarter approach consists of *transforming* into defeasible the part of the ontologies producing the inconsistency letting the part which is not in conflict as is.

For simplicity, in the following discussion we assume *unique name assumption* (UNA). If UNA could not be assumed, it would be always possible to use an ontology integration schema based on a mapping function as it was presented in Gómez *et al.* [1].

Definition 9 (Merged δ-ontology). *Let* Σ_1 *and* Σ_2 *be two* δ*-ontologies. The merged ontology between* Σ_1 *and* Σ_2 *is noted as* $\Sigma_1 \oplus \Sigma_2$*.*

In the same way as with single δ-ontologies, the merge of two δ-ontologies will be interpreted as a DeLP program.

[1] In [12], a DeLP program composed of material implications instead of derivation rules is called a *knowledge structure.*

[2] This situation can be modeled by two δ-ontologies with non-empty Sbox, and empty Dbox and non-empty Abox.

Definition 10 (Interpretation of a merged δ-ontology). *Let Σ_1 and Σ_2 be two δ-ontologies such that $\Sigma_1 = (T_S^1, T_D^1, A^1)$ and $\Sigma_2 = (T_S^2, T_D^2, A^2)$. The interpretation of the merged δ-ontology $\Sigma_1 \oplus \Sigma_2$, noted as $\mathcal{T}(\Sigma_1 \oplus \Sigma_2)$, is defined as the DeLP program (Π, Δ) where*

$$\begin{aligned}
\Pi_1 &= \mathcal{T}_\Pi(T_S^1) \cup \mathcal{T}_\Pi(A^1);\\
\Delta_1 &= \mathcal{T}_\Delta(T_D^1);\\
\Pi_2 &= \mathcal{T}_\Pi(T_S^2) \cup \mathcal{T}_\Pi(A^2);\\
\Delta_2 &= \mathcal{T}_\Delta(T_D^2);\\
(\Pi, \Delta') &= (\Pi_1, \Delta_1) \star \Pi_2, \text{ and}\\
\Delta &= \Delta_1 \cup \Delta_2 \cup \Delta'.
\end{aligned}$$

We now extend the reasoning tasks over Aboxes for the case of a merged ontology. In particular, we define the *instance checking operation* for merged ontologies.

Definition 11 (Instance checking for a merged δ-ontology). *Let Σ_1 and Σ_2 be two δ-ontologies. let C a concept name and a an individual name.*

- *The individual a is a potential member of the concept C w.r.t. $\Sigma_1 \oplus \Sigma_2$ iff there exists an argument $\langle \mathcal{A}, C(a) \rangle$ w.r.t. $\mathcal{T}(\Sigma_1 \oplus \Sigma_2)$.*
- *The individual a is a justified member of the concept C w.r.t. $\Sigma_1 \oplus \Sigma_2$ iff there exists a warranted argument $\langle \mathcal{A}, C(a) \rangle$ w.r.t. $\mathcal{T}(\Sigma_1 \oplus \Sigma_2)$.*
- *The individual a is an strict member of the concept C w.r.t. $\Sigma_1 \oplus \Sigma_2$ iff there exists an argument $\langle \emptyset, C(a) \rangle$ w.r.t. $\mathcal{T}(\Sigma_1 \oplus \Sigma_2)$.*
- *The individual a is an indeterminate member of the concept C w.r.t. $\Sigma_1 \oplus \Sigma_2$ iff there is no argument for the literal $C(a)$ w.r.t. $\mathcal{T}(\Sigma_1 \oplus \Sigma_2)$.*

Definition 12 (Set of justified and strict answers). *Let Σ be a δ-ontology, a an individual and p a concept. The set of justified answers of Σ is the set of literals $p(a)$ such that a belongs justifiedly to p. The set of strict answers of Σ stands for all the literals $p(a)$ such that a belongs strictly to p.*

Example 3. Suppose we are given the δ-ontology $\Sigma_1 = (T_S^1, \emptyset, A^1)$, where:

$$T_S^1 = \left\{ \begin{array}{l} \mathsf{Penguin} \sqsubseteq \mathsf{Bird} \\ \mathsf{Bird} \sqsubseteq \mathsf{Flies} \end{array} \right\}, \text{ and}$$
$$A^1 = \left\{ \begin{array}{l} \mathsf{TWEETY} : \mathsf{Bird} \\ \mathsf{OPUS} : \mathsf{Penguin} \end{array} \right\}.$$

The set of strict answers of this ontology is the set of literals: $\{bird(tweety)$, $penguin(opus)$, $bird(opus)$, $flies(tweety)$, $flies(opus)\}$.

Let us suppose that we receive another δ-ontology $\Sigma_2 = (T_S^2, \emptyset, A^2)$, viewed as an explanation for "$\mathsf{OPUS} : \neg\mathsf{Flies}$", where:

$$T_S^2 = \left\{ \mathsf{Bird} \sqcap \mathsf{Penguin} \sqsubseteq \neg\mathsf{Flies} \right\}, \text{ and}$$
$$A^2 = \left\{ \begin{array}{l} \mathsf{OPUS} : \mathsf{Bird} \\ \mathsf{OPUS} : \mathsf{Penguin} \end{array} \right\}.$$

Suppose that we now desire to find the DeLP program $\mathcal{P} = (\Pi, \Delta) = \mathcal{T}(\Sigma_1 \oplus \Sigma_2)$ that interprets δ-ontology which merges Σ_1 and Σ_2. When we compute the interpretation of the merged ontology, we must perform a kernel revision by a set of sentences. We need to find the minimally inconsistent subsets of the set of DeLP sentences: $\mathcal{T}_\Pi(A^1) \cup \mathcal{T}_\Pi(T_S^1) \cup \mathcal{T}_\Pi(A^2) \cup \mathcal{T}_\Pi(T_S^2)$. The two sets in this condition are:

1. $\mathfrak{Trans}(\left\{\begin{array}{l} \mathit{bird}(\mathit{opus}), \\ \mathit{penguin}(\mathit{opus}), \\ (\mathit{flies}(X) \;\leftarrow\; \mathit{bird}(X), \mathit{penguin}(X)), \\ (\mathit{flies}(X) \;\leftarrow\; \mathit{bird}(X)) \end{array}\right\})$, and
2. $\mathfrak{Trans}(\left\{\begin{array}{l} \mathit{penguin}(X), \\ (\mathit{bird}(X) \;\leftarrow\; \mathit{penguin}(X)), \\ (\mathit{flies}(X) \;\leftarrow\; \mathit{bird}(X)), \\ ({\sim}\mathit{flies}(X) \;\leftarrow\; \mathit{bird}(X), \mathit{penguin}(X)) \end{array}\right\})$.

To discard general beliefs, we must discard at least one sentence in each set above. As the sentence "$\mathit{flies}(X) \;\leftarrow\; \mathit{bird}(X)$" is in both sets, it can be discarded. The set Π of strict rules of the revised ontology is then composed by:

$$\Pi = \left\{\begin{array}{l} \mathit{bird}(\mathit{tweety}), \\ \mathit{bird}(\mathit{opus}), \\ \mathit{penguin}(\mathit{opus}) \end{array}\right\} \cup \mathfrak{Trans}(\{\, \mathit{bird}(X) \;\leftarrow\; \mathit{penguin}(X) \,\}) \cup$$
$$\mathfrak{Trans}(\{\, {\sim}\mathit{flies}(X) \;\leftarrow\; \mathit{bird}(X), \mathit{penguin}(X) \,\}).$$

In this case, the set of strict answers of the merged ontology $\Sigma_1 \oplus \Sigma_2$ is $\{\mathit{bird}(\mathit{tweety}), \mathit{bird}(\mathit{opus}), \mathit{penguin}(\mathit{opus}), {\sim}\mathit{flies}(\mathit{opus})\}$. Nevertheless, the set of deleted sentences are not completely forgotten but stored as defeasible rules. That is, the set Δ of defeasible rules in the interpretation of the merged δ-ontology is $\Delta = \{(\mathit{flies}(X) \prec \mathit{bird}(X)), ({\sim}\, \mathit{bird}(X) \prec {\sim}\mathit{flies}(X))\}$. Then the set of justified answers of $\Sigma_1 \oplus \Sigma_2$ is $\{\mathit{bird}(\mathit{tweety})$, $\mathit{bird}(\mathit{opus})$, $\mathit{penguin}(\mathit{opus})$, ${\sim}\mathit{flies}(\mathit{opus})$, $\mathit{flies}(\mathit{tweety})\}$. Notice that the literal "$\mathit{flies}(\mathit{tweety})$" is present in the set of justified answers but it is not in the set of strict answers; *i.e.* we are now able to conclude that the individual Tweety is a justified member of the concept Flies.

5 Conclusion

We have presented an approach for merging ontologies based on Belief Revision and Defeasible Logic Programming. We have combined argumentation, belief revision and description logic ontologies for extending the δ-ontologies framework and thus showing how to merge two ontologies in which the union of the strict terminologies could lead to inconsistency. To solve this problem, we revisited a procedure presented by [12] in which part of the offending terminologies are turned *defeasible* by using a kernel revision operator. We have presented a framework for characterizing the behavior of the proposed approach and an example scenario. Future work includes characterizing mathematical properties of the approach.

References

1. Gómez, S.A., Chesñevar, C.I., Simari, G.R.: Reasoning with Inconsistent Ontologies Through Argumentation. Applied Artificial Intelligence **1**(24) (2010) 102–148
2. Zhang, X., Zhang, Z., Lin, Z.: An Argumentative Semantics for Paraconsistent Reasoning in Description Logic ALC (2009)
3. Bassiliades, N., Antoniou, G., Vlahavas, I.P.: DR-DEVICE: A Defeasible Logic System for the Semantic Web. In: PPSWR 2004. (2004) 134–148
4. Wang, K., Billington, D., Blee, J., Antoniou, G.: Combining Description Logic and Defeasible Logic for the Semantic Web. In: RuleML 2004. (2004) 170–181
5. Grigoris Antoniou, Antonis Bikakis, G.W.: A System for Nonmonotonic Rules on the Web. In: RuleML 2004. (2004) 23–36
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook – Theory, Implementation and Applications. Cambridge University Press (2003)
7. McGuiness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview (2004)
8. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scient. American (2001)
9. Gruber, T.R.: A translation approach to portable ontologies. Knowledge Acquisition **5**(2) (1993) 199–220
10. Chesñevar, C.I., Maguitman, A., Loui, R.: Logical Models of Argument. ACM Computing Surveys **32**(4) (December 2000) 337–383
11. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artif. Intell. **171**(10-15) (2007) 619–641
12. Falappa, M.A., Kern-Isberner, G., Simari, G.R.: Explanations, Belief Revision and Defeasible Reasoning. Artificial Intelligence **141** (2002) 1–28
13. Falappa, M.A., García, A.J., Kern-Isberner, G., Simari, G.R.: On the evolving relation between Belief Revision and Argumentation. The Knowledge Engineering Review **26**(1) (2011) 35–43
14. Falappa, M.A., Kern-Isberner, G., Simari, G.R.: Belief Revision and Argumentation Theory. In Rahwan, I., Simari, G.R., eds.: Argumentation in Artificial Intelligence, Springer (2009) 341–360
15. García, A., Simari, G.: Defeasible Logic Programming an Argumentative Approach. Theory and Prac. of Logic Program. **4**(1) (2004) 95–138
16. 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)
17. Stolzenburg, F., García, A., Chesñevar, C., Simari, G.: Computing Generalized Specificity. J. of N.Classical Logics **13**(1) (2003) 87–113
18. Alchourron, C., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet functions for contraction and revision. Journal of Symbolic Logic (50) (1985) 510–530
19. Falappa, M.A.: Teoría de Cambio de Creencias y sus Aplicaciones sobre Estados de Conocimiento. PhD thesis, Universidad Nacional del Sur (1999)
20. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., Uschold, M., eds.: Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA (August 4–5, 2001)