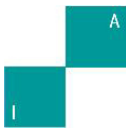
Inteligencia Artificial 16(52) (2013), 16-28



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Merging of Ontologies Using Belief Revision and Defeasible Logic Programming

Sergio Alejandro Gómez, Guillermo Ricardo 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 Ontology merging refers to the process of creating a new ontology from two or more existing ontologies with overlapping parts. The *δ-ontologies* framework allows to reason with possibly inconsistent Description Logic ontologies by interpreting them in Defeasible Logic Programming, which is an approach to common-sense reasoning based on defeasible argumentation and logic programming. In traditional Description Logics, the knowledge expressed by a single ontology is separated into a terminological box and an assertional box. In contrast, in a δ-ontology, the terminological box is separated into an strict box (which must always be consistent) and a defeasible box (that could be inconsistent). Merging defeasible boxes is trivial but merging strict boxes is not since its union could result in an inconsistent strict box. In this article we extend the δ-ontologies framework by combining Argumentation, Belief Revision, and Description Logic ontologies to merge two ontologies such that the union of the strict terminologies could lead to inconsistency. We base our approach on a procedure presented by Falappa *et al.* where part of the inconsistent terminologies are *turned* defeasible by using a kernel revision operator applied to the set union of the ontologies.

Resumen La mezcla de ontologías se refiere al proceso de creación de una nueva ontología a partir de dos más ontologías existentes con partes en común. El marco de las *δ-ontologías* permite razonar con ontologías en Lógica para la Descripción posiblemente inconsistentes al interpretarlas en Programación en Lógica Rebatible, la cual es un acercamiento al razonamiento de sentido común basado en argumentación rebatible y programación en lógica. En las Lógicas para la Descripción tradicionales, el conocimiento expresado por una ontología simple es separado en una caja terminológica y una caja asercional. En contraste, en una δ-ontología, la caja terminológica es separada en una caja estricta (que siempre debe ser consistente) y una caja rebatible (que podría ser inconsistente). Mezclar cajas rebatibles es trivial pero mezclar cajas estrictas no lo es puesto que tal unión podría resultar en una caja estricta inconsistente. En este artículo, extendemos el marco de las δ-ontologías al combinar Argumentación, Revisión de Creencias y Lógicas para la Descripción para mezclar dos ontologías tales que la unión de las terminologías estrictas podría llevar a inconsistencia. Basamos nuestro acercamiento en un procedimiento presentado por Falappa *et al.* donde parte de las terminologías inconsistentes son *convertidas* en rebatibles por medio de un operador de revisión de núcleo aplicado a la unión conjuntista de las ontologías.

Keywords: Ontology merging, Description Logics, Defeasible Logic Programming, Belief Revision

Palabras clave: Mezcla de ontologías, Lógicas para la Descripción, Programación en Lógica Rebatible, Revisión de Creencias

ISSN: 1988-3064(on-line)

1 Introduction

The research topic that focus on the confluence of Description Logics and Argumentation has shown its importance by an ever growing list of research publications (see [14, 33, 4, 31, 2].) Description Logics [3] constitute the semantic substrate of the Web Ontology Language OWL [22], which is at the core of the Semantic Web initiative. The Semantic Web [6] is a vision of the current Web where resources have exact meaning assigned in terms of knowledge bases called *ontologies* [16], enabling agents to reason about them. Argumentation is a form of non-monotonic reasoning that allows to entail consequences from possibly inconsistent knowledge bases [5, 28]. On the other hand, Belief Revision is the process of changing beliefs to take into account a new piece of information [1, 17]. The confluence of Argumentation and Belief Revision is not new, but still can be regarded as a thriving research topic (see [11, 10, 12]).

Ontology merging refers to the process of creating a new ontology from two or more existing ontologies that may overlap [21]. An ontology can be inconsistent because the modeling of the knowledge it represents inherently leads to conflict, some other times the inconsistency will come from importing another ontology. Usually, the inconsistencies have to be corrected by the knowledge engineer, but often this cannot be done in this way for the knowledge engineer lacks the expertise or is not authorized to correct the inconsistencies. Regarding this problem, Gómez *et al.* [14] developed the *δ-ontologies* framework that allows to reason in the presence of inconsistent Description Logic ontologies; this system uses an argumentative reasoning framework based on logic programming called Defeasible Logic Programming [13]. In a δ-ontology the terminology defining the vocabulary is separated in *strict* and *defeasible*, where the former stands for a set of strict inclusion axioms which must be inconsistency free, and the latter stands for a set of defeasible axioms which can be potentially inconsistent. While the process of joining two defeasible terminologies is trivial, putting together two strict terminologies can lead to inconsistency if this is done without proper care.

In this article, we combine Argumentation, Belief Revision and Description Logic ontologies to extend the δ-ontologies framework for allowing to merge two ontologies in which the direct union of their respective strict terminologies could lead to an inconsistency. To solve this problem, we revisit a procedure based on a kernel revision operator presented by Falappa *et al.* in [11] for finding the minimal subset of a set of inconsistent logic programming rules and converting that subset in a set of defeasible rules. The common approach in kernel revision is to delete the offending sentences to restore consistency; instead, the mechanism proposed in [11] will restore consistency without losing valuable information, since a weakened form of that information will be retained in the form of defeasible. We use this procedure to merge two δ-ontologies: their defeasible terminologies are combined by performing a set union, but their strict terminologies along with their assertional box are considered through the lens of the method described in [11]; thus, if the set union of the strict terminologies and the assertional boxes is inconsistent a distinguished part of the union is turned into defeasible.

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. In Section 5 we compare this work with related research. Finally, in Section 6 we discuss the presentation.

2 Background

2.1 Description Logics

Description Logics (DL) are a well-known family of knowledge representation formalisms [3]. They are based on the notions of *concepts* (unary predicates, classes) and *roles* (binary relations), 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 ($\neg C$), 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 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* (or terminological box) which introduces the *terminology* and an *Abox* (or assertional box) 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 $a : C$ (meaining a is a member of C) and *role assertions* of the type $\langle a, b\rangle : R$ (meaning a is related to b through R), where C is a concept description, R is a role name, and a and b are individual names.

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. Here, we are only concerned with Abox reasoning, more precisely with *instance checking* [3], that consists of determining if an assertion is entailed from an Abox. For instance, $T \cup A \models a : C$ indicates 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 [3]. 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 exist no models for it.

2.2 Defeasible Logic Programming

Defeasible Logic Programming (DeLP) [13] provides a language for knowledge representation and reasoning that uses *defeasible argumentation* [29] to decide between contradictory conclusions through a *dialectical analysis*. Codifying knowledge as 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 of the form $P \leftarrow Q_1, \ldots, Q_n$, and a set Δ of defeasible rules of the form $P \prec Q_1, \ldots, Q_n$ 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 (expressed in terms of a logical contradiction), an attack relationship between arguments can be defined. A particular criterion is used to decide defeat between two conflicting arguments; *generalized specificity* is a syntactic criterion that favors more informed arguments to less informed ones and arguments with shorter derivations to arguments with longer derivations (see [30] for details). If the attacking argument is strictly preferred over the attacked one, then it is called a *proper defeater*. If no comparison is possible, or both arguments are equi-preferred, the attacking argument is called a *blocking defeater*. To determine whether a given argument $\mathcal{A}$ is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for $\mathcal{A}$, defeaters for these defeaters, and so on, are taken into account. Given a DeLP program $\mathcal{P}$ and a query H, the final answer to H w.r.t. $\mathcal{P}$ 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). 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 [14], Gómez *et al.* showed how DeLP can be used for handling inconsistencies in ontologies in such a way that they are automatically solved by the reasoning system. In that manner we obtain the capability of reasoning with inconsistent ontologies; however, we also lose some expressiveness in the involved ontologies. As Def. 1 shows, certain restrictions will have to be imposed on DL ontologies to make possible to represent them in the DeLP language.

In part, our proposal is based in the work of Grosof *et al.*[15], where it is shown how the processing of ontologies can be improved by the use of techniques from logic programming. In particular, they have identified a subset of DL languages that can be effectively mapped into a Horn-clause logics. Below, we recall the fundamental concepts of that proposal to make the article more self-contained.

Definition 1 ($\mathcal{L}_b$, $\mathcal{L}_h$ and $\mathcal{L}_{hb}$ classes) *Let A be an atomic class name, C and D class expressions, and R a property. In the $\mathcal{L}_h$ language, $C \sqcap D$ is a class, and $\forall R.C$ is also a class. Class expressions in $\mathcal{L}_h$ are called* $\mathcal{L}_h$-classes. *In the $\mathcal{L}_b$ language, $C \sqcup D$ is a class, and $\exists R.C$ is a class too. Class expressions in $\mathcal{L}_b$ are called* $\mathcal{L}_b$-classes. *The $\mathcal{L}_{hb}$ language is defined as the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$. Class expressions in $\mathcal{L}_{hb}$ are called* $\mathcal{L}_{hb}$-classes.

Definition 2 (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.*

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

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

Figure 1: A very simple δ-ontology

For assigning semantics to a δ-ontology, we will use two translation functions (based on [15]) from DL to DeLP called $\mathcal{T}_\Delta$ (from *defeasible boxes* to *defeasible rules*) and $\mathcal{T}_\Pi$ (from *strict boxes* to *strict rules*). Further details of these translation functions can be found in [14].

The basic premise for achieving the translation of DL ontologies into DeLP is based on the observation that a DL inclusion axiom "$\mathsf{C} \sqsubseteq \mathsf{D}$" is regarded as a First-Order Logic statement "$(\forall x)(C(x) \rightarrow D(x))$," which in turn is regarded as a Horn-clause "$d(X) \leftarrow c(X)$"[1] First, we assume that in all DL formulas, negation occurs only immediately before atomic formulas. The formula "$\mathsf{C} \sqcap \mathsf{D} \sqsubseteq \mathsf{E}$" is treated as "$e(X) \leftarrow c(X), d(X)$." Furthermore, Lloyd-Topor transformations are used to handle special cases as conjunctions in the head of rules and disjunctions in the body of rules; so "$\mathsf{C} \sqsubseteq \mathsf{D} \sqcap \mathsf{E}$" is interpreted as two rules: "$d(X) \leftarrow c(X)$" and "$e(X) \leftarrow c(X)$", while "$\mathsf{C} \sqcup \mathsf{D} \sqsubseteq \mathsf{E}$" is transformed into: "$e(X) \leftarrow c(X)$" and "$e(X) \leftarrow d(X)$." Likewise axioms of the form "$\exists \mathsf{r}.\mathsf{C} \sqsubseteq \mathsf{D}$" are treated as "$d(X) \leftarrow r(X, Y), c(Y)$." Sbox axioms are considered *strict* and are transformed using $\mathcal{T}_\Pi$, *e.g.*, $\mathcal{T}_\Pi(\mathsf{C} \sqsubseteq \mathsf{D})$ is interpreted as $\{(d(X) \leftarrow c(X)), (\sim c(X) \leftarrow \sim d(X))\}$. Abox assertions are always considered strict, *e.g.*, $\mathcal{T}_\Pi(\mathsf{a} : \mathsf{C})$ is regarded as a fact $c(a)$ and $\mathcal{T}_\Pi(\langle \mathsf{a}, \mathsf{b}\rangle : \mathsf{r})$ as $r(a, b)$. Formally:

Definition 3 ($\mathcal{T}_\Pi^*$ 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 "$H \leftarrow B_2$".*

[1] Following the notation standard in logic programming, constant and predicate names in DeLP rules begin with lowercase letters and variable names begin with an initial uppercase.

To reason with contraposition in DeLP, the function $\mathcal{T}_\Pi$ computes transposition of rules, computing the closure of strict rules [2]. This accounts for taking every strict rule $r = H \leftarrow B_1, \ldots, B_n$ as a material implication in propositional logic which is equivalent to the disjunction $B_1 \vee \ldots \vee B_n \vee \neg H$. From that disjunction different rules of the form $B_1 \vee B_2 \vee B_{i-1} \vee \neg B \vee B_{i+1} \ldots \vee B_n \rightarrow \neg B_i$ can be obtained (which are called the transpositions of r). Formally:

Definition 4 (Transpositions 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 transpositions 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 5 ($\mathcal{T}_\Pi$ mapping from DL sentences to DeLP strict rules) *The mapping from DL ontologies into DeLP strict rules is defined as $\mathcal{T}_\Pi(T) = \mathfrak{Trans}(\mathcal{T}^*_\Pi(T))$.*

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

Figure 2: Mapping from DL ontologies to DeLP strict rules

Definition 6 (Interpretation of a δ-ontology) *Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology. The* interpretation of Σ *is a DeLP program $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.*

[2] To the best of our knowledge this procedure was first proposed by Caminada [7] in regard to rule-based argumentation.

To keep consistency within an argument, some internal coherence between the Abox and the Tbox must be enforced; *i.e.*, 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 7 (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. 6 as shown in Fig. 3. From $\mathcal{P}_1$, we can determine Socrates justifiedly belongs to the concept* Mortal *in Σ_1 as there exists a warranted argument structure $\langle \mathcal{A}_1, mortal(socrates)\rangle$ where:*

$$\mathcal{A}_1 = \{\ mortal(socrates) \prec man(socrates)\ \}$$

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

$$\mathcal{B}_1 = \{\ mortal(mcleod) \prec man(mcleod)\ \}$$

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

$$\mathcal{B}_2 = \{\ \sim mortal(mcleod) \prec man(mcleod), highlander(mcleod)\ \}$$

Notice that as $\mathcal{B}_2$ is more specific than $\mathcal{B}_1$, $\mathcal{B}_1$ cannot defeat $\mathcal{B}_2$ (see Section 2.2).

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

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

Figure 3: DeLP program $\mathcal{P}_1$ obtained from ontology Σ_1

4 Ontology Merging based on Belief Revision Theory

We will now extend the δ-ontologies framework to allow for ontology merging based on belief revision. First, we will recall the fundamentals of ontology merging, then we will review the basics of belief revision, and finally we will present the actual extension along with a running example.

4.1 Fundamentals of Ontology Merging

In the Semantic Web initiative Description Logic ontologies are actually implemented in the Web Ontology Language (OWL). OWL makes an open world assumption; that is, descriptions of resources are not confined to a single file or scope. While class C_1 may be defined originally in ontology Σ_1, it can be extended in other ontologies. The consequences of these additional propositions about C_1 are monotonic. New information cannot retract previous information. New information can be contradictory, but facts and entailments can only be added, never deleted. The possibility of such contradictions is something

the designer of an ontology needs to take into consideration. As it was originally expected by W3C Recommendation, tool support should help detect such cases [22]. In this regard, Pellet [26] notably can detect the source of inconsistency in an ontology. Despite these advances, the user must debug the ontology, making it consistent.

Other form of extending an ontology is by combining two or more ontologies into one, usually known as *integration* [21, 27]. *Combining* refers to the use of two or more ontologies for a task in which their relation is important [21]. Other authors see this last notion as just *using* ontologies [27], *i.e.*, the integration of ontologies into applications. *Merging/integration* is the process of creating a new ontology from two or more existing ontologies with overlapping parts [21]. Pinto *et al.* [27] distinguish between *integration* of ontologies (when building a new ontology reusing other available ontologies) and *merging* of different ontologies about the same subject into a single one that unifies all of them.

Integrated ontologies could not be in agreement; thus, the process of *aligning* brings two or more ontologies into mutual agreement, making them consistent and coherent [21]. To do this a map must be built, thus *mapping* consists of relating similar concepts or relations from different sources to each other using an equivalence relation [21]. *Articulation* is the point of linkage between two aligned ontologies [21]. Articulation points can have the semantics equivalent, subsumes (`is-a`), property (`part-of` and/or `has-knowledge-of`) [24, 23].

Changes to an ontology result in the production of another ontology. *Transforming* consists of changing the semantics of an ontology slightly to make it suitable for a purpose different than the original one. A *version* is the result of a change to an ontology. *Versioning* is a mechanism for keeping track between old and new evolved ontologies.

4.2 Fundamentals of Belief Revision

Belief revision is the process of changing beliefs to take into account a new piece of information. Two types of change are usually distinguished [1, 9]: *update* and *revision*. In *update*, 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 in *revision*, there are also old beliefs and new information; but, in this case the new information is considered more reliable, then the process of revision inserts the new information into the set of old beliefs without generating an inconsistency. Belief revision should produce minimal change, *i.e.*, the knowledge changed should be as minimal as possible.

Given two sets 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.* [11] 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 using an incision function that makes a "cut" over each minimally inconsistent subset of $K \cup A$. Formally:

Definition 8 (α-Kernel [11]) *Let K be a set of sentences and α a sentence. Then $K^{\perp\!\!\!\perp}\alpha =_{df} \{K' | (K' \subseteq K) \wedge (K' \vdash \alpha) \wedge ((K" \subset K') \Rightarrow (K" \nvdash \alpha))\}$.*

Definition 9 (External incision function [11]) *Let K be a set of sentences. An* external incision function *for K is a function $(\sigma) : 2^{2^{\mathcal{L}}} \mapsto 2^{\mathcal{L}}$, such that for any set $A \subseteq \mathcal{L}$:*

1. *$\sigma((K \cup A))^{\perp\!\!\!\perp}\perp) \subseteq \bigcup((K \cup A^{\perp\!\!\!\perp}\perp)$, and*
2. *If $X \in (K \cup A)^{\perp\!\!\!\perp}\perp$ and $X \neq \emptyset$, then $(X \cap (K \cup A)^{\perp\!\!\!\perp}\perp) \neq \emptyset$.*

Definition 10 (Kernel revision by a set of sentences [11]) *Let K and A be sets of sentences and (σ) be an external incision function for K. The operator $(\circ)$ of* kernel revision by a set of sentences *$((\circ) : 2^{\mathcal{L}} \mapsto 2^{\mathcal{L}})$ is defined as $K \circ A = (K \cup A) \setminus \sigma((K \cup A)^{\perp\!\!\!\perp}\perp)$.*

In [11], 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, in [11] proposal, 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 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 11 (Positive/negative transformation [11]) *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 12 (Kernel (partial meet) composed revision [11]) *Let (K, Δ) be a knowledge structure,*[3] *$(\circ)$ 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:*

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

The set K' contains the revised undefeasible beliefs, Δ'_1 is the transformation in defeasible rules of particular beliefs (also called *presumptions* [13, Section 6]) eliminated from K whereas Δ'_2 is the transformation of general beliefs eliminated from K into defeasible rules.

4.3 Merging of δ-ontologies using Belief Revision

Now, we will adapt the reasoning framework for δ-ontologies for its use in ontology merging. Merging is the process of creating a new ontology from two or more existing ontologies, possibly with overlapping parts [21]. Suppose we have two *strict* ontologies[4] that we want to reason with simultaneously; however, just putting both ontologies together may generate inconsistencies. The simplest solution is to consider the ontologies as modeling *defeasible knowledge*; but this solution is too simplistic, and a smarter approach consists of *transforming* the part of the ontologies producing the inconsistency into defeasible, leaving untouched the part which is not in conflict.

For simplicity, in the following discussion we will assume the *unique name assumption* (UNA). If UNA could not be assumed, it would always be possible to use an ontology integration schema based on a mapping function as it was presented in Gómez *et al.* [14].

Notation: Let Σ_1 and Σ_2 be two δ-ontologies. The *merged ontology* resulting of merging 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.

Definition 13 (Interpretation of a merged δ-ontology) *Let Σ_1 and Σ_2 be two δ-ontologies such that $\Sigma_1 = (T^1_S, T^1_D, A^1)$ and $\Sigma_2 = (T^2_S, T^2_D, A^2)$. The interpretation of the merged δ-ontology $\Sigma_1 \oplus \Sigma_2$,*

[3] In [11], a DeLP program composed of material implications instead of derivation rules is called a *knowledge structure*.
[4] This situation can be modeled by two δ-ontologies with non-empty Sbox, and empty Dbox and non-empty Abox.

noted as $\mathcal{T}(\Sigma_1 \oplus \Sigma_2)$*, is defined as the DeLP program* (Π, Δ) *where*

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

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

Definition 14 (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 15 (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:*

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

The set of strict answers of this ontology is the set of literals:

$$\mathfrak{StrictAnswers}(\Sigma_1) = \left\{ \begin{array}{l} \mathit{bird}(\mathit{tweety}),\\ \mathit{penguin}(\mathit{opus}),\\ \mathit{bird}(\mathit{opus}),\\ \mathit{flies}(\mathit{tweety}),\\ \mathit{flies}(\mathit{opus}) \end{array} \right\}$$

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:*

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

Now, suppose that we want to find the DeLP program $\mathcal{P} = (\Pi, \Delta) = \mathcal{T}(\Sigma_1 \oplus \Sigma_2)$ *that interprets the* 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* S_1 *and* S_2 *in this condition are:*

$$S_1 = \mathfrak{Trans}(\left\{ \begin{array}{l} bird(opus), \\ penguin(opus), \\ (flies(X) \leftarrow bird(X), penguin(X)), \\ (flies(X) \leftarrow bird(X)) \end{array} \right\}) \quad and$$

$$S_2 = \mathfrak{Trans}(\left\{ \begin{array}{l} penguin(X), \\ (bird(X) \leftarrow penguin(X)), \\ (flies(X) \leftarrow bird(X)), \\ (\sim flies(X) \leftarrow bird(X), penguin(X)) \end{array} \right\})$$

To discard general beliefs, we must discard at least one sentence in each set above. As the sentence "$flies(X) \leftarrow bird(X)$*" is in* $S_1 \cap S_2$*, it can be discarded. The set* Π *of strict rules of the revised ontology is then composed by:*

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

In this case, the set of strict answers of the merged ontology $\Sigma_1 \oplus \Sigma_2$ *is*

$$\mathfrak{StrictAnswers}(\Sigma_1 \oplus \Sigma_2) = \left\{ \begin{array}{l} bird(tweety), \\ bird(opus), \\ penguin(opus), \\ \sim flies(opus) \end{array} \right\}.$$

Nevertheless, the set of deleted sentences is 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*

$$\Delta = \left\{ \begin{array}{l} flies(X) \prec bird(X), \\ \sim bird(X) \prec \sim flies(X) \end{array} \right\}.$$

Then the set of justified answers of $\Sigma_1 \oplus \Sigma_2$ *is*

$$\mathfrak{JustifiedAnswers}(\Sigma_1 \oplus \Sigma_2) = \left\{ \begin{array}{l} bird(tweety), \\ bird(opus), \\ penguin(opus), \\ \sim flies(opus), \\ flies(tweety) \end{array} \right\}.$$

Notice that the literal "$flies(tweety)$*" 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 Related Work

To the best of our knowledge, the treatment of DL ontologies in Prolog first appeared in Grosof *et al.* [15]. They showed how to inter-operate, semantically and inferentially, between the leading Semantic Web approaches to rules (RuleML Logic Programs) and ontologies (OWL DL) by analyzing their expressive intersection. They defined 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 showed 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 [15] 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 has been addressed in related non-monotonic approaches including the addition of a preference order on the axioms [18], imposing answer set programming rules on top of ontologies Eiter *et al.* [8], using Paraconsistent Logics [19] or Belief Revision [32] to determine a consistent subset of an inconsistent ontology. In this regard, our approach determines the inconsistent subset of an ontology and transforms it into defeasible axioms, without completely losing valuable information that otherwise would have not been considered in future queries to the ontology.

In [18], Heymans and Vermier extended the DL $\mathcal{SHOQ}(D)$ with a preference order on the axioms. With this strict partial order certain axioms can be overruled when defeated with more preferred ones. They also imposed a preferred model semantics, introducing non-monotonicity into $\mathcal{SHOQ}(D)$. As in [18], we allow to perform inferences from inconsistent ontologies by considering subsets (arguments) of the original ontology. In [18] a hard-coded comparison criterion on DL axioms is imposed. 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. For that reason, we believe our approach can be considered more declarative; in particular, the comparison criterion in DeLP is modular, so that rule comparison could also be adopted [13].

Eiter *et al.* [8] propose a combination of logic programming under answer set semantics with the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$; this combination allows for building rules on top of ontologies. Notice that our approach can be extended to have this feature on top of two merged ontologies.

Huang *et al.* [19] 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.

Imam *et al.* [20] discuss various implementation issues for the development of a prototype merging system which will provide an inconsistency-tolerant reasoning mechanism applicable to the health-care domain. Their approach, which is based on paraconsistent logic, is similar to ours in the sense that does not lose information.

To debug an inconsistent ontology, Moguillansky *et al.* [25] proposed an acceptability semantics for arguments to obtain a related maximal consistent ontology. They showed how to obtain arguments from a DL ontology and how to use an abstract argumentation framework to reason with a possibly inconsistent ontology; despite our work can be considered in the same line of research, it is based solely on the merging of ontologies in the language of DeLP.

6 Conclusion and Future Work

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 an inconsistency. As δ-ontologies are interpreted as Defeasible Logic Programming rules, to solve this problem, we applied a procedure in which part of an inconsistent set of logic programming rules are transformed into *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 interesting formal properties of the approach as well as testing it on concrete inconsistent OWL ontologies.

Acknowledgements

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)*, Universidad Nacional del Sur in Bahía Blanca, Argentina.

References

[1] Carlos E. Alchourron, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, (50):510–530, 1985.

[2] Grigoris Antoniou, Antonis Bikakis, and Gerd Wagner. A System for Nonmonotonic Rules on the Web. In *RuleML 2004*, pages 23–36, 2004. doi: 10.1007/978-3-540-30504-0_3.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook – Theory, Implementation and Applications*. Cambridge University Press, 2003.

[4] Nick Bassiliades, Grigoris Antoniou, and Ioannis P. Vlahavas. DR-DEVICE: A Defeasible Logic System for the Semantic Web. In *PPSWR 2004*, pages 134–148, 2004.

[5] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007. doi: 10.1016/j.artint.2007.05.001.

[6] Timothy Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 2001.

[7] Martin Caminada. On the Issue of Contraposition of Defeasible Rules. *COMMA 2008*, pages 109–115, 2008.

[8] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. *KR 2004*, pages 141–151, 2004.

[9] Marcelo A. Falappa. *Teoría de Cambio de Creencias y sus Aplicaciones sobre Estados de Conocimiento*. PhD thesis, Universidad Nacional del Sur, 1999.

[10] Marcelo A. Falappa, Alejandro J. García, Gabriele Kern-Isberner, and Guillermo R. Simari. On the evolving relation between Belief Revision and Argumentation. *The Knowledge Engineering Review*, 26(1):35–43, 2011. doi: 10.1017/S0269888910000391.

[11] Marcelo A. Falappa, Gabriele Kern-Isberner, and Guillermo R. Simari. Explanations, Belief Revision and Defeasible Reasoning. *Artificial Intelligence*, 141:1–28, 2002.

[12] Marcelo A. Falappa, Gabriele Kern-Isberner, and Guillermo R. Simari. Belief Revision and Argumentation Theory. In Iyad Rahwan and Guillermo R. Simari, editors, *Argumentation in Artificial Intelligence*, pages 341–360. Springer, 2009. doi: 10.1007/978-0-387-98197-0_17.

[13] Alejandro J. García and Guillermo R. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004. doi: 10.1.1.59.6130.

[14] Sergio A. Gómez, Carlos I. Chesñevar, and Guillermo R. Simari. Reasoning with Inconsistent Ontologies Through Argumentation. *Applied Artificial Intelligence*, 1(24):102–148, 2010. doi: 10.1080/08839510903448692.

[15] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logics. *WWW2003, Hungary*, 2003.

[16] Thomas R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[17] Sven Ove Hansson. *A Textbook of Belief Dymanics: Theory Change and Database Updating*. Kluwer Academic Publishers, 1999.

[18] Stijn Heymans and Dirk Vermeir. A Defeasible Ontology Language. *On the Move to Meaningful Internet Systems - Confederated Int. Conf. DOA, CoopIS and ODBASE*, pages 1033–1046, 2002.

[19] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with Inconsistent Ontologies. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 454–459, Edinburgh, Scotland, August 2005.

[20] Fahim T. Imam, Wendy MacCaull, and Margaret A. Kennedy. Merging Healthcare Ontologies: Inconsistency Tolerance and Implementation Issues. *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS '07)*, pages 74–85, 2007.

[21] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In Asuncion Gomez-Perez, Michael Gruninger, Heiner Stuckenschmidt, and Michael Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, August 4–5, 2001.

[22] Deborah L. McGuiness and Frank van Harmelen. OWL Web Ontology Language Overview, 2004.

[23] Prasenjit Mitra. *An Algebraic Framework for the Interoperation of Ontologies*. PhD thesis, Department of Electrical Engineering, 2004.

[24] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. *Lecture Notes in Computer Science*, 1777:86+, 2000.

[25] Martín O. Moguillansky, Nicolás D. Rotstein, and Marcelo A. Falappa. Generalized Abstract Argumentation: A First-order Machinery towards Ontology Debugging. *Inteligencia Artificial*, 46:17–33, 2010.

[26] Bijan Parsia and Evren Sirin. Pellet: An OWL DL Reasoner. In *3rd International Semantic Web Conference (ISWC2004)*, 2004.

[27] H. Sofia Pinto, Asunción Gómez-Pérez, and João P. Martins. Some issues on ontology integration. In *Proceedings of the IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, pages 7.1–7.12, 1999.

[28] Iyad Rahwan and Guillermo R. Simari. *Argumentation in Artificial Intelligence*. Springer, 2009.

[29] Guillermo R. Simari and Ronald P. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53:125–157, 1992.

[30] Frieder Stolzenburg, Alejandro J. García, Carlos I. Chesñevar, and Guillermo R. Simari. Computing Generalized Specificity. *J. of N.Classical Logics*, 13(1):87–113, 2003.

[31] Kewen Wang, David Billington, Jeff Blee, and Grigoris Antoniou. Combining Description Logic and Defeasible Logic for the Semantic Web. In *RuleML 2004*, pages 170–181, 2004.

[32] Philip D. Wasserman. *Neural Computing. Theory and Practice*. Van Nostrand Reinhold, 1989.

[33] Xiaowang Zhang, Zhihu Zhang, and Zuoquan Lin. An Argumentative Semantics for Paraconsistent Reasoning in Description Logic ALC, 2009.