A Defeasible Logic Programming Approach to the Integration of Rules and Ontologies*

Sergio Alejandro Gómez† **Carlos Iván Chesñevar**†‡ **Guillermo Ricardo Simari**†

† **Artificial Intelligence Research and Development Laboratory**
Department of Computer Science and Engineering
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
Email: {sag,cic,grs}@cs.uns.edu.ar
‡ **Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina**

ABSTRACT

The Semantic Web is a vision of the current Web where resources have exact meaning assigned in terms of ontologies, thus enabling agents to reason about them. As inconsistencies cannot be treated by standard reasoning approaches, we use Defeasible Logic Programming (DeLP) to reason with possibly inconsistent ontologies. In this article we show how to integrate rules and ontologies in the Semantic Web. We present an approach that can be used to suitably extend the SWRL standard by incorporating classical and default negated literals in Semantic Web rules in the presence of incomplete and possibly inconsistent information. The rules and ontologies will be interpreted as a DeLP program allowing the rules to reason on top of a set of (possibly inconsistent) ontologies.

Keywords: Semantic Web, ontologies, Description Logics, defeasible argumentation, Defeasible Logic Programming

1. INTRODUCTION

The Semantic Web [4] (SW) is a vision of the current Web where resources have exact meaning assigned in terms of ontologies [11], thus enabling agents to reason about them. The Ontology Layer of the SW is well developed with the OWL [14] language whose underlying semantics is based on the *Description Logics* (DL) [2], for which specialized reasoners exist [12].

The goal of the Rule Layer of the SW is to complement ontologies when ontology languages cannot fulfill all of the expressivity requirements for describing a domain. In particular, rules present advantages over plain DL ontologies such as to allow expressing more than one free variable at a time; besides arbitrary OWL classes can be used as predicates in rules, with rules and ontology axioms freely mixed.

The proposed standard for representing rules in the SW is the SWRL language [13]. Despite its maturity, SWRL indirectly allows representing classical negation (by means of the complement operator) and, to the best of our knowledge, does not allow using default negation. We believe that in order to fully exploit the potential of SW rules, it is necessary that SWRL may be able to represent both kinds of negations, a necessity that was pointed out in the beginning of SW research (see [15]).

*This paper is an extended version of the article S.A.Gómez, C.I.Chesñevar, G.R.Simari. Integration of Rules and Ontologies with Defeasible Logic Programming. *XV Congreso Argentino en Ciencias de la Computacin (CACIC 2009)*, Pp. 90–99, 2009.

Although reasoners such as Hermit and Pellet are capable of handling OWL ontologies extended with SWRL rules and provide efficient implementations for detecting inconsistent ontologies, they are incapable of performing inferences upon the latter. In previous works [8, 9], we presented a formalism called δ-ontologies capable of reasoning with potentially inconsistent DL ontologies.

In this article, we extend the δ-ontologies framework to suitably model incomplete and possibly inconsistent sets of rules on top of δ-ontologies. Rules in the proposed extension are to be interpreted as DeLP programs with special primitives to access the knowledge contained in the ontology layer. In this way rules and ontologies are integrated as a single DeLP program upon which queries can be posed. The result of such queries will depend on the content of the rules as well as the contents of the underlying ontologies.

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 building rules on top of ontologies. In Section 5, we show how to interpret default negated literals in SW rules in DeLP. Section 6 discusses implementation issues. Finally Section 7 discusses related work and Section 8 concludes the paper.

2. BACKGROUND

Description Logics

Description Logics (DL) are a well-known family of knowledge representation formalisms [2]. They are based on the notions of *concepts* (unary predicates or classes) and *roles* (binary relations or properties), 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 ($\neg C$), 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 such 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.

One way of providing semantics to DL ontologies is in terms of First-Order Logic by means of a function $\phi(\cdot)$ (see [2]). It is said that Σ is *consistent* if $\phi(\Sigma)$ has a model, otherwise it is *inconsistent*. Determining if an individual a is a member of a concept C w.r.t. Σ is called *instance checking* and it refers to determining if the logical entailment $\phi(\Sigma) \models C(a)$ holds. [1]

Defeasible Logic Programming

Defeasible Logic Programming (DeLP) [7] provides a language for knowledge representation and reasoning that uses *defeasible argumentation* [5, 3] 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 Δ of defeasible rules $P \prec Q_1, \ldots, Q_n$, and a set Π of strict rules $P \leftarrow Q_1, \ldots, Q_n$ 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, the former 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. Thus the answer to a query can be either *Yes*, when there is at least one warranted argument supporting H on the basis of $\mathcal{P}$; *No*, when there is at least one warranted argument supporting $\sim H$ on the basis of $\mathcal{P}$; *Undecided*, when neither H nor $\sim H$ are warranted w.r.t. $\mathcal{P}$, or *Unknown*, if H does not belong to the signature of $\mathcal{P}$.

3. REASONING WITH INCONSISTENT ONTOLOGIES IN DELP

In the presence of inconsistent ontologies, traditional DL reasoners (such as RACER [12]) 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. In previous works [8, 9], we 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 by translating DL ontologies into DeLP programs. By doing so the capability of reasoning with inconsistent ontologies is gained but also some expressiveness in the involved ontologies is lost as certain restrictions have to be imposed on DL ontologies in order to be expressed in the DeLP language.

Our proposal is based in part in the work of [10] 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.

We therefore translate 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 $\mathcal{P}$ corresponds to the Abox A joined with T_S in Σ, and the set Δ of defeasible rules corresponds to T_D in Σ. This translation is 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 $\Delta = \mathcal{T}_\Delta(T_D)$. Notice that these functions preserve the meaning of the original DL sentences.

As noted in [10], for DL sentences to be mapped into Horn-logic rules, they must satisfy certain constraints. Conjunction and universal restrictions appearing in the right-hand side of inclusion axioms can be mapped to heads of rules (called $\mathcal{L}_h$-classes). In contrast, conjunction, disjunction and existential restriction can be mapped to rule bodies whenever they occur in the left-hand side of inclusion axioms (called $\mathcal{L}_b$-classes). As equality axioms "$C \equiv D$" are interpreted as two inclusion axioms "$C \sqsubseteq D$" and "$D \sqsubseteq C$," they must belong to the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$.

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 *Consider the δ-ontologies $\Sigma_1 = (\emptyset, T_D^1, A^1)$ about swimming and $\Sigma_2 = (T_S^2, T_D^2, A^2)$ about programming both presented in Fig. 1. The defeasible terminology T_D^1 says that both free and scuba divers are divers; saturation divers are scuba divers; somebody who swims a race stroke is usually a race swimmer, and someone who can swim a rescue stroke is normally considered a rescue swimmer. The assertional box A^1 establishes that crawl is a race stroke; side is a rescue stroke; John is able to swim both crawl and side strokes, and finally Paul is a saturation diver. The strict terminology T_S^2 expresses that among programming languages, both logic programming and object-oriented languages can be found. The Dbox T_D^2 says that a programmer is usually somebody who can program in some programming language unless she has failed the elementary*

[1]The symbol $\models$ denotes the usual entailment relation in FOL.

programming course. The Abox A^2 establishes that Prolog is a logic programming language and that John can program in the Prolog programming language; that Java is an object-oriented language and that Mary can program Java code, and that Paul is capable of programming in the Java programming language although he failed the elementary programming course.

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

Defeasible terminology T_D^1**:**
Free_Diver ⊔ Scuba_Diver ⊑ Diver
Saturation_Diver ⊑ Scuba_Diver
∃swims.Race_Stroke ⊑ Race_Swimmer
∃swims.Rescue_Stroke ⊑ Rescue_Swimmer

Assertional box A^1**:**
CRAWL : Race_Stroke
SIDE : Rescue_Stroke
⟨JOHN, CRAWL⟩ : swims
⟨JOHN, SIDE⟩ : swims
PAUL : Saturation_Diver

Programming ontology $\Sigma_2 = (T_S^2, T_D^2, A^2)$**:**

Strict terminology T_S^2**:**
LP_Lang ⊔ OOP_Lang ⊑ Lang

Defeasible terminology T_D^2**:**
∃programs.Lang ⊑ Programmer
∃programs.Lang ⊓ Failed_Prog_101 ⊑ ¬Programmer

Assertional box A^2**:**
PROLOG : LP_Lang
JAVA : OOP_Lang
⟨JOHN, PROLOG⟩ : programs
⟨MARY, JAVA⟩ : programs
⟨PAUL, JAVA⟩ : programs
PAUL : Failed_Prog_101

Figure 1: Ontologies Σ_1 and Σ_2 about swimming and programming, resp.

We now recall the definitions of $\mathcal{T}_\Pi$ and $\mathcal{T}_\Delta$, for details see [8, 9].

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

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 $\mathcal{T}_\Pi^* : 2^{\mathcal{L}_{DL}} \longrightarrow 2^{\mathcal{L}_{DeLP_\Pi}}$ mapping is defined in Fig. 3. Besides, intermediate transformations of the form "$(H_1 \wedge H_2) \leftarrow B$" are rewritten as two rules "$H_1 \leftarrow B$" and "$H_2 \leftarrow B$." Similarly transformations of the form "$H_1 \leftarrow H_2 \leftarrow B$" are rewritten as "$H_1 \leftarrow B \wedge H_2$," and rules of the form "$H \leftarrow (B_1 \vee B_2)$" are rewritten as two rules "$H \leftarrow B_1$" and "$H \leftarrow B_2$."*

Definition 4 (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:*

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

Figure 2: Mapping from DL ontologies to DeLP defeasible rules

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

Definition 5 ($\mathcal{T}_\Pi$ mapping from DL sentences to DeLP strict rules) *We define the mapping from DL ontologies into DeLP strict rules 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} \\
 & & \text{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} & \left\{ \begin{array}{l} P(X,Z) \leftarrow \\ \qquad P(X,Y) \wedge P(Y,Z) \end{array} \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}
$$

Figure 3: 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 $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.*

Notice that in order to keep consistency within an argument, we must enforce some internal coherence between the Abox and the Tbox; 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 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, *noted as $\mathfrak{PotentialMember}(a, C, \Sigma)$, 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, *noted as $\mathfrak{JustifiedMember}(a, C, \Sigma)$, 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, *noted as $\mathfrak{StrictMember}(a, C, \Sigma)$, iff there exists an argument $\langle \emptyset, C(a) \rangle$ w.r.t. $\mathcal{P}$.*

Example 2 (Continues Ex. 1) *Consider again the δ-ontologies Σ_1 and Σ_2, they are interpreted as the DeLP programs $\mathcal{P}_1$ and $\mathcal{P}_2$ according to Def. 6 as shown in Fig. 4. From $\mathcal{P}_1$, we can determine that John justifiedly belongs to the concept* Race_Swimmer *in Σ_1 as there exists a warranted argument structure $\langle \mathcal{A}_1, race\_swimmer(john) \rangle$ where:* [2]

$$\mathcal{A}_1 = \left\{ \begin{array}{l} race\_swimmer(john) \prec \\ swims(john, crawl), race\_stroke(crawl) \end{array} \right\}.$$

Likewise, there are warranting arguments $\mathcal{A}_2$ and $\mathcal{A}_3$ for $rescue\_swimmer(john)$ and $diver(paul)$ resp., with:

$$\mathcal{A}_2 = \left\{ \begin{array}{l} rescue\_swimmer(john) \prec \\ swims(john, side), rescue\_stroke(side) \end{array} \right\} \text{ and}$$

$$\mathcal{A}_3 = \left\{ \begin{array}{l} diver(paul) \prec scuba\_diver(paul) \\ scuba\_diver(paul) \prec \\ \quad saturation\_diver(paul) \end{array} \right\}.$$

From $\mathcal{P}_2$ in turn we can conclude that both John and Mary justifiedly belong to the concept Programmer *but Paul justifiedly belongs to the concept* $\neg$Programmer *as there are warranted arguments $\langle \mathcal{B}_1, programmer(john) \rangle$, $\langle \mathcal{B}_2, programmer(mary) \rangle$, and $\langle \mathcal{B}_3, \sim programmer(paul) \rangle$, where:*

$$\mathcal{B}_1 = \left\{ \begin{array}{l} programmer(john) \prec \\ \quad programs(john, prolog) \end{array} \right\},$$

$$\mathcal{B}_2 = \left\{ \begin{array}{l} programmer(mary) \prec \\ \quad programs(mary, java) \end{array} \right\}, \text{ and}$$

$$\mathcal{B}_3 = \left\{ \begin{array}{l} \sim programmer(paul) \prec \\ \quad programs(paul, java), \\ \quad failed\_prog\_101(paul) \end{array} \right\}.$$

Notice that there exists another argument $\langle \mathcal{B}_4, programmer(paul) \rangle$ with $\mathcal{B}_4 = \{programmer(paul) \prec programs(paul, java)\}$ that is defeated by argument $\mathcal{B}_3$.

[2]For space reasons we do not include complete dialectical analyses. See Sect. 6 for checking them on-line.

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

Facts Π_1:
$race\_stroke(crawl)$. $rescue\_stroke(side)$.
$swims(john, crawl)$. $swims(john, side)$.
$saturation\_diver(paul)$.

Defeasible rules Δ_1:
$diver(X) \prec free\_diver(X)$.
$diver(X) \prec scuba\_diver(X)$.
$scuba\_diver(X) \prec saturation\_diver(X)$.
$race\_swimmer(X) \prec swims(X, Y), race\_stroke(Y)$.
$rescue\_swimmer(X) \prec$
$\quad swims(X, Y), rescue\_stroke(Y)$.

DeLP program $\mathcal{P}_2 = (\Pi_2, \Delta_2)$ obtained from Σ_2:

Facts and strict rules Π_2:
$lp\_lang(prolog)$. $oop\_lang(java)$.
$programs(john, prolog)$. $programs(mary, java)$.
$programs(paul, java)$. $failed\_prog\_101(paul)$.
$lang(X) \leftarrow lp\_lang(X)$.
$\sim lp\_lang(X) \leftarrow \sim lang(X)$.
$lang(X) \leftarrow oop\_lang(X)$.
$\sim oop\_lang(X) \leftarrow \sim lang(X)$.

Defeasible rules Δ_2:
$programmer(X) \prec programs(X, Y), lang(Y)$.
$\sim programmer(X) \prec$
$\quad programs(X, Y), lang(Y), failed\_prog\_101(X)$.

Figure 4: DeLP programs $\mathcal{P}_1$ and $\mathcal{P}_2$ obtained from ontologies Σ_1 and Σ_2, resp.

4. ADDING RULES ON TOP OF ONTOLOGIES

We now define how to express rules in the Semantic Web with the presence of incompleteness and potential inconsistency. The notions presented will lead to the central definition of *integration system* that joins rules and ontologies making it suitable for a SW setting.

Definition 8 (Strict, justified and potential membership statements) *Let a be an individual name, C a concept name, and Σ a δ-ontology. The expression "$\mathfrak{StrictMember}(a, C, \Sigma)$" is called a* strict membership statement *and queries if "a" strictly belongs to "C" w.r.t. Σ. The expression "$\mathfrak{JustifiedMember}(a, C, \Sigma)$" is called a* justified membership statement *and queries if "a" justifiedly belongs to "C" w.r.t. Σ. The expression "$\mathfrak{PotentialMember}(a, C, \Sigma)$" is called a* potential membership statement *and queries if "a" potentially belongs to "C" w.r.t. Σ.*

Definition 9 (Semantic web strict rule) *A* semantic web strict rule *is an ordered pair, denoted "$B \Longrightarrow H$," whose first member, B, is a finite set of literals or potential membership statements, and whose second member, H, is a literal. A semantic web strict rule with antecedent $\{L_1, \ldots, L_n\}$ and head H will be also written as "$L_1 \wedge \ldots \wedge L_n \Longrightarrow H$."*

Definition 10 (Semantic web defeasible rule) *A* semantic web defeasible rule *is an ordered pair, denoted "$B \succ\!\!= H$," whose first member, B, is a finite set of literals or potential membership statements, and whose second member, H, is a literal. A semantic web defeasible rule with antecedent $\{L_1, \ldots, L_n\}$ and head H will be also written as "$L_1 \wedge \ldots \wedge L_n \succ\!\!= H$."*

Definition 11 (Semantic web program) *Let $\mathcal{S}$ be a set of semantic web strict rules and $\mathcal{D}$ a set of semantic web defeasible rules. A* semantic web program *is a pair $\langle\mathcal{S},\mathcal{D}\rangle$.*

Definition 12 (Integration system) *Let $\mathcal{P}$ be a semantic web program and let $\Sigma_1,\ldots,\Sigma_n$ be n δ-ontologies. An* integration system of rules and ontologies *$\mathcal{I}$ is a pair $\langle\mathcal{P},\{\Sigma_i\}_{i=1,\ldots,n}\rangle$.*

Example 3 *Consider the semantic web program $\mathcal{P} = \langle\mathcal{S},\mathcal{D}\rangle$ presented in Fig. 5, this SW program will be integrated with ontologies Σ_1 and Σ_2 from Ex. 1 into the integration system $\mathcal{I} = \langle\mathcal{P},\{\Sigma_1,\Sigma_2\}\rangle$. In $\mathcal{P}$, the set of semantic web strict rules $\mathcal{S}$ expresses that somebody who potentially belongs to the concept "race swimmer" (resp. "rescue swimmer") in ontology Σ_1 is a race swimmer (resp. rescue swimmer) and that whoever is a potential member of the concept "programmer" in ontology Σ_2 is a computer geek. The set of semantic web defeasible rules $\mathcal{D}$ says that computer geeks are not usually good at sports but expert swimmers normally are; if somebody is either capable of swimming both a race stroke and a rescue stroke the he is often considered an expert swimmer; finally, a diver is usually considered an expert swimmer.*

Set of strict semantic web rules $\mathcal{S}$:

$\mathfrak{PotentialMember}(x, \mathsf{Race\_Swimmer}, \Sigma_1) \Longrightarrow \mathsf{Race\_Swimmer}(x)$
$\mathfrak{PotentialMember}(x, \mathsf{Rescue\_Swimmer}, \Sigma_1) \Longrightarrow \mathsf{Rescue\_Swimmer}(x)$
$\mathfrak{PotentialMember}(x, \mathsf{Programmer}, \Sigma_2) \Longrightarrow \mathsf{Geek}(x)$

Set of defeasible semantic web rules $\mathcal{D}$:

$\mathsf{Geek}(x) \succ\!\!= \neg\mathsf{Good}(x)$
$\mathsf{Swimmer}(x) \succ\!\!= \mathsf{Good}(x)$
$\mathsf{Race\_Swimmer}(x) \wedge \mathsf{Rescue\_Swimmer}(x) \succ\!\!= \mathsf{Swimmer}(x)$
$\mathfrak{PotentialMember}(x, \mathsf{Diver}, \Sigma_1) \succ\!\!= \mathsf{Swimmer}(x)$

Figure 5: Semantic web program $\mathcal{P} = \langle\mathcal{S},\mathcal{D}\rangle$

In order to answer queries posed against an integration system of rules and ontologies, we will interpret integration systems as DeLP programs. We define next the notions of semantic interpretation and answer to a query for an integration system.

Definition 13 (Semantic interpretation) *Let $\mathcal{I} = (\mathcal{P},\{\Sigma_1,\ldots,\Sigma_n\})$ be an integration system such that: $\mathcal{P} = \langle\Pi^{\mathcal{P}},\Delta^{\mathcal{P}}\rangle$, $\Sigma_1 = (T_S^1, T_D^1, A^1)$, …, $\Sigma_n = (T_S^n, T_D^n, A^n)$. The* semantic interpretation of *$\mathcal{I}$, noted as $\mathfrak{Sem}(\mathcal{I})$, is the DeLP program (Π,Δ), where:*

$$\Pi = \Phi(\Pi^{\mathcal{P}}) \cup \bigcup_{i=1,\ldots,n} \mathcal{T}(T_S^i) \cup \bigcup_{i=1,\ldots,n} \mathcal{T}(A^i)$$

and

$$\Delta = \Phi(\Delta^{\mathcal{P}}) \cup \bigcup_{i=1,\ldots,n} \mathcal{T}(T_D^i),$$

and $\Phi(\cdot)$ stands for the interpretation function of Semantic Web rules as defined in Fig. 6.

Definition 14 (Answer to a query in a SW integration system) *Let $\mathcal{I}$ be a SW integration system and L a literal. The answer to the query L, noted as $\mathfrak{Answer}_{\mathcal{I}}(L)$, is defined as:*

- YES *iff the answer to the query L is Yes w.r.t. $\mathfrak{Sem}(\mathcal{I})$;*
- NO *iff the answer to the query $\sim L$ is Yes w.r.t. $\mathfrak{Sem}(\mathcal{I})$, and*
- UNDECIDED *iff the answer to the query L is Undecided. w.r.t. $\mathfrak{Sem}(\mathcal{I})$.*

Example 4 (Continues Ex. 3) *Consider again the integration system $\mathcal{I}$ presented in Ex. 3. When we compute $\mathfrak{Sem}(\mathcal{I})$, we obtain the DeLP program formed by the fragments presented in Fig. 7 along with the ones already presented in Ex. 2. We will show that $\mathfrak{Answer}_{\mathcal{I}}(good(john))$ is* UNDECIDED, *$\mathfrak{Answer}_{\mathcal{I}}(good(mary))$ is* NO, *and $\mathfrak{Answer}_{\mathcal{I}}(good(paul))$ is* YES.

First, we will consider the dialectical analysis for the query "$good(john)$." There exists an argument $\langle\mathcal{C}_1, good(john)\rangle$ where:

$$\mathcal{C}_1 = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \left\{\begin{array}{l} (good(john) \prec swimmer(john)),\\ (swimmer(john) \prec race\_swimmer(john),\\ \quad rescue\_swimmer(john)) \end{array}\right\}.$$

However, there is an argument $\langle\mathcal{C}_2, \sim good(john)\rangle$, that says John is not good at sports as he is a geek (because he is a programmer), that defeats argument $\mathcal{C}_1$, where: $\mathcal{C}_2 = \mathcal{B}_1 \cup \{\sim good(john) \prec geek(john)\}$. Therefore, the answer for the query "$good(john)$" is UNDECIDED.

When we consider the dialectical analysis for determining the answer to the query "$good(mary)$," we find out that there is a warranted argument $\langle\mathcal{B}_2 \cup \{geek(mary) \prec programmer(mary)\}, \sim good(mary)\rangle$.

Last, let us consider the dialectical tree for the literal "$good(paul)$." There is an argument $\langle\mathcal{D}_1, good(paul)\rangle$, based on the defeasible information that asserts that Paul is an expert swimmer (because he is a saturation diver), with:

$$\mathcal{D}_1 = \mathcal{A}_3 \cup \left\{\begin{array}{l} (good(paul) \prec swimmer(paul)),\\ (swimmer(paul) \prec diver_{\Sigma_2}(paul)) \end{array}\right\}.$$

But argument $\mathcal{D}_1$ is attacked by an argument $\langle\mathcal{D}_2, \sim good(paul)\rangle$, where:

$$\mathcal{D}_2 = \mathcal{B}_4 \cup \left\{\begin{array}{l} (\sim good(paul) \prec geek(paul)),\\ (geek(paul) \prec programmer(paul)) \end{array}\right\}.$$

Nevertheless, as Paul failed the elementary programming course, this argument is defeated by argument $\mathcal{B}_3$ (see Ex. 2), thus reinstating argument $\mathcal{D}_1$.

$$\begin{array}{rcl}
\Phi(B_1 \wedge \ldots \wedge B_n \Longrightarrow A) & =_{df} & \\
 & \Phi(A) \leftarrow \Phi(B_1),\ldots,\Phi(B_1) & \\
\Phi(B_1 \wedge \ldots \wedge B_n \succ\!\!= A) & =_{df} & \\
 & \Phi(A) \prec \Phi(B_1),\ldots,\Phi(B_1) & \\
\Phi(L(x_1,\ldots,x_n)) & =_{df} & L(X_1,\ldots,X_n)\\
\Phi(\neg L(x_1,\ldots,x_n)) & =_{df} & \sim L(X_1,\ldots,X_n)\\
\Phi(\mathfrak{PotentialMember}(a,C,\Sigma)) & =_{df} & C_{\Sigma}(a)
\end{array}$$

Figure 6: Interpretation of Semantic Web rules as DeLP rules

5. ADDING DEFAULT NEGATION TO SW RULES

We now discuss how to add default negation to the approach to Semantic Web rules presented in Sect. 4. A de-

Strict rules $\Pi_{\mathcal{D}}$**:**

$race\_swimmer(X) \leftarrow race\_swimmer_{\Sigma_1}(X).$
$rescue\_swimmer(X) \leftarrow rescue\_swimmer_{\Sigma_1}(X).$
$geek(X) \leftarrow programmer_{\Sigma_2}(X).$

Defeasible rules $\Delta_{\mathcal{D}}$**:**

$\sim good(X) -\!\!\!\prec geek(X).$
$good(X) -\!\!\!\prec swimmer(X).$
$swimmer(X) -\!\!\!\prec$
$\quad race\_swimmer(X), rescue\_swimmer(X).$
$swimmer(X) -\!\!\!\prec diver_{\Sigma_1}(X).$

Figure 7: DeLP program $\mathcal{P}' = (\Pi_{\mathcal{S}}, \Delta_{\mathcal{D}})$ obtained from the interpretation of $\mathcal{P} = \langle \mathcal{S}, \mathcal{D} \rangle$

fault negated literal L is denoted as $\mathsf{not}L$ and is only allowed in the body of SW rules. This negation is interpreted by the DeLP default negation (for details see [7, Sect. 6.1]) and the definition of the interpretation function $\Phi(\cdot)$ for SW rules (which was presented in Def. 13) is extended accordingly by adding the equation

$$\Phi(\mathsf{not}L(x_1, \ldots, x_n)) \quad =_{df} \quad \mathsf{not}L(X_1, \ldots, X_n).$$

We now illustrate the inner workings of this extended approach to SW rules with an example that continues the above presented ones.

Example 5 *Consider again the δ-ontology $\Sigma_1 = (T_S^1, T_D^1, A_1)$ presented in Ex. 1. Let $\Sigma_1' = (T_S^1, T_D^1, A_1 \cup \{(\mathsf{WILLIAM} : \mathsf{Free\_Diver}), (\mathsf{NATALIA} : \mathsf{Free\_Diver})\})$,* i.e. *there are two additional facts stating that both William and Natalia are freedivers. Consider also the δ-ontology Σ_3 presented in Fig. 8; it says that somebody who suffers a swimming disease is injured; someone who went to a see a doctor can usually be consider treated; both swimmer's ear and pink-eye are swimming diseases; William suffers from swimmer's ear; Natalia suffers from pink eye, and it is known that William went to see a doctor.*

Let $\mathcal{P}_5$ be a new SW program obtained from the SW program $\mathcal{P}$ (presented in Ex. 3) by adding an extra SW defeasible rule indicating that some injured swimmer that it is not know if he was treated is supposed to be unathletic. Formally, $\mathcal{P}_5 = \langle \Pi^{\mathcal{P}}, \Delta^{\mathcal{P}} \cup \{\mathsf{Swimmer}(x), \mathsf{Injured}(x), \mathsf{notTreated}(x) \succ\!\!== \neg\mathsf{Good}(x)\}\rangle$. Let $\mathcal{I}_5 = \langle \mathcal{P}_5, \{\Sigma_1, \Sigma_2, \Sigma_3\}\rangle$. We will present how the answers to queries w.r.t. $\mathfrak{Sem}(\mathcal{I}_5)$ are computed, in particular we will see that $\mathfrak{Answer}_{\mathcal{I}_5}(good(william))$ is Yes *while $\mathfrak{Answer}_{\mathcal{I}_5}(good(natalia))$ is* No. *When we consider the DeLP code in Fig. 9 along with the DeLP code already presented in Ex. 4, we see there is an argument $\langle \mathcal{E}_1, good(william)\rangle$ supporting that William is athletic because he is a free-diver; formally,*

$$\begin{array}{lcl} \mathcal{E}_1 & = & \{\ (good(william) -\!\!\!\prec swimmer(william))\ \} \cup \mathcal{D}_1 \\ \mathcal{D}_1 & = & \left\{ \begin{array}{l} (swimmer(william) -\!\!\!\prec diver(william)), \\ (diver(william) -\!\!\!\prec free\_diver(william)) \end{array} \right\}. \end{array}$$

There is also an argument $\langle \mathcal{G}_1, \sim good(william)\rangle$ expressing that William is not good at sports because he is injured (as he suffers from swimmer's ear); more formally,

$$\mathcal{G}_1 = \mathcal{D}1 \cup \left\{ \begin{array}{l} (\sim good(william) -\!\!\!\prec \\ \quad swimmer(william), \\ \quad injured(william), \\ \quad \mathsf{not}treated(william)), \\ (injured(william) -\!\!\!\prec \\ \quad suffers(william, swimmers\_ear), \\ \quad swimmingDisease(swimmers\_ear)) \end{array} \right\}.$$

However, the argument $\mathcal{G}_1$ is defeated by $\langle \mathcal{H}_1, treated(william)\rangle$ expressing that William was treated (as he went to see a doctor), thus reinstating $\mathcal{E}_1$:

$$\mathcal{H}_1 = \left\{ \begin{array}{l} treated(william) -\!\!\!\prec \\ \quad went\_to\_see\_a\_doctor(william) \end{array} \right\}.$$

When we consider the arguments concerning Natalia's case, the following dialectical analysis arises. First, there is an argument $\langle \mathcal{E}_2, good(natalia)\rangle$ expressing that Natalia is athletic because she is a swimmer (she is a freediver and freedivers are a kind of diver that it is ultimately a swimmer); this argument is defeated by another argument $\langle \mathcal{D}_2, \sim good(natalia)\rangle$ saying that Natalia is not good at sports as she is an injured swimmer (she suffers from a pink eye condition). In turn the latter argument does not have any defeaters and it is thus warranted.

Strict terminology T_S^3**:**
$\exists$suffers.SwimmingDisease $\sqsubseteq$ Injured

Defeasible terminology T_D^3**:**
WentToSeeADoctor $\sqsubseteq$ Treated

Abox A^3**:**
SWIMMERS_EAR : SwimmingDisease
PINK_EYE : SwimmingDisease
$\langle$WILLIAM, SWIMMERS_EAR$\rangle$: suffers
$\langle$NATALIA, PINK_EYE$\rangle$: suffers
WILLIAM : WentToSeeADoctor

Figure 8: Ontology $\Sigma_3 = (T_S^3, T_D^3, A^3)$ about swimming injuries

New facts obtained from extra assertions added to Σ_1**:**
$free\_diver(william). \quad free\_diver(natalia).$

Defeasible rule obtained from extra defeasible SW rule:
$\sim good(X) -\!\!\!\prec swimmer(X), injured(X), \mathsf{not}treated(X).$

Set of strict rules Π^3 **obtained form** T_S^3**:**
$injured(X) \leftarrow suffers(X, Y), swimmingDisease(Y).$

Set of defeasible rules Δ^3 **obtained form** T_D^3**:**
$treated(X) -\!\!\!\prec went\_to\_see\_a\_doctor(X).$

Set of facts obtained from A_3**:**
$swimmingDisease(swimmers\_ear).$
$swimmingDisease(pink\_eye).$
$suffers(william, swimmers\_ear).$
$suffers(natalia, pink\_eye).$
$went\_to\_see\_a\_doctor(william).$

Figure 9: Remaining portion of the semantic interpretation of $\mathcal{I}_5$

6. IMPLEMENTATION ISSUES

As performing defeasible argumentation is a computationally complex task, an abstract machine called JAM (Justification Abstract Machine) has been specially developed for an efficient implementation of DeLP [7]. JAM provides an argument-based extension of the traditional WAM (Warren's Abstract Machine) for Prolog. A full-fledged implementation of DeLP is available online,[3] including facilities for visualizing arguments and dialectical trees. The DeLP programs presented in this article can be downloaded and tried on-line.[4]

[3]See http://lidia.cs.uns.edu.ar/delp_client.
[4]See http://cs.uns.edu.ar/~sag/papers/programjcst2010.delp.

7. RELATED WORK

Eiter *et al.* [6] 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 as we do. However, in contrast to our approach, they are not able to handle inconsistencies neither in the ontologies nor in the rule bases.

Williams and Hunter [16] 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.

In the classification of systems for combining rules and ontologies presented in [1], two kinds are distinguished: hybrid and homogeneous. Our approach can be considered homogeneous in the sense that it does not distinguish between predicates in rules and ontologies and uses DeLP as a specialized reasoner.

8. CONCLUSIONS AND FUTURE WORK

We have presented a novel approach for combining rules and ontologies in the Semantic Web that could be used as a basis for suitably extending the current standard SWRL for representing rules on top of OWL ontologies. The proposed approach allows to add incomplete and possibly inconsistent rules on top of also possibly inconsistent ontologies by interpreting them as DeLP programs. We have presented a framework for characterizing the behavior of the proposed approach and an example scenario including both classical and negated literals in rules.

As future work, the formal properties arising from the approach must be characterized. Other research issue is related to the inclusion of both strict and justified membership statements in Semantic Web rules as in this work we have only considered the inclusion of potential membership statements. Our current research efforts are directed toward solving these issues.

Acknowledgments: This research is founded by Universidad Nacional del Sur (UNS), Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 No. 13.096, PICT 2003 No. 15.043, PAV 2004 076), and by Projects PIP 112-200801-02798 (CONICET, Argentina), TIN2006-15662-C02-01 (MEC, Spain), PGI 24/ZN10 (SGCyT, UNS, Argentina).

References

[1] Grigoris Antoniou, Carlos Viegas Damasio, Benjamin Grosof, Ian Horrocks, Michael Kiefer, Jan Maluszynski, and Peter F. Patel-Schneider. Combining Rules and Ontologies. A survey. REWERSE 2005, 2005.

[2] 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.

[3] T. J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artif. Intell.*, 171(10-15):619–641, 2007.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scient. American*, 2001.

[5] Carlos Iván Chesñevar, Ana Maguitman, and Ronald Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.

[6] 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.

[7] A. García and G. Simari. Defeasible Logic Programming an Argumentative Approach. *Theory and Prac. of Logic Program.*, 4(1):95–138, 2004.

[8] Sergio Alejandro Gómez, Carlos Iván Chesñevar, and Guillermo Ricardo Simari. An Argumentative Approach to Reasoning with Inconsistent Ontologies. In Thomas Meyer and Mehmet A. Orgun, editors, *Proc. of the Knowledge Representation in Ontologies Workshop (KROW 2008)*, volume CPRIT 90, pages 11–20, Sydney, Australia, 2008.

[9] Sergio Alejandro Gómez, Carlos Iván Chesñevar, and Guillermo Ricardo Simari. Reasoning with Inconsistent Ontologies Through Argumentation. *Applied Artificial Intelligence*, 1(24):102–148, 2010.

[10] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logics. *WWW2003, May 20-24, Budapest, Hungary*, 2003.

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

[12] Volker Haarslev and Ralf Möller. RACER System Description. Technical report, University of Hamburg, Computer Science Department, 2001.

[13] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. National Research Council of Canada, Network Inference, and Stanford University, 2004.

[14] Deborah L. McGuiness and Frank van Harmelen. OWL Web Ontology Language Overview, 2004. http://www.w3.org/TR/owl-features/.

[15] Gerd Wagner. Web Rules Need Two Kinds of Negation. In N. Henze F. Bry and J. Maluszynski, editors, *Proc. of the 1st International Workshop, PPSW3 '03. Springer-Verlag LNCS 2901*, 2003.

[16] M. Williams and A. Hunter. Harnessing ontologies for argument-based decision-making in breast cancer. *Proc. of the Intl. Conf. on Tools with AI (ICTAI'07)*, pages 254–261, 2007.