Uncertainty and Description Logic Programs: A Proposal for Expressing Rules and Uncertainty on Top of Ontologies

Umberto Straccia
ISTI-CNR, Via G. Moruzzi 1, I-56124 Pisa ITALY
Umberto.Straccia@isti.cnr.it

Abstract

Rule-based and object-oriented techniques are rapidly making their way into the infrastructure for representing and reasoning about the Semantic Web and combining these two paradigms emerges as an important objective. We present a new family of representation languages, which extents existing language families for the Semantic Web: namely *Description Logic Programs* (DLPs) and DLPs with *uncertainty* (μDLPs). The former combine the expressive power of description logics (which capture the meaning of the most popular features of structured representation of knowledge) and disjunctive logic programs (powerful rule-based representation languages). The latter are DLPs in which the management of uncertainty is considered as well. We show that μDLPs may be applied in the context of distributed information search in the Semantic Web, where the representation of the inherent uncertainty of the relationships among resource ontologies, to which an automated agent has access to, is required.

Category: F.4.1: Mathematical Logic and Formal Languages: Mathematical Logic: [Logic and constraint programming]

Category: I.2.3: Artificial Intelligence: Deduction and Theorem Proving: [Logic programming]

Terms: Theory

Keywords: Description Logics, Logic programs, uncertainty, Semantic Web

1 Introduction

The *Semantic Web*[1] [7] is widely regarded as the next step in the evolution of the World Wide Web. It aims at enhancing content on the World Wide Web with machine-readable meta-data, that should support *agents* (machines or human users) to richer discovery, data integration, navigation, and automation of tasks. *Ontologies* [34] play

[1] www.semanticweb.org

a key role in the Semantic Web and major effort has been put by the Semantic Web community into this issue. Ontologies provide a source of shared and precisely defined terms that can be used in such meta-data. Informally, an ontology consists typically of a hierarchical description of important concepts in a particular domain, along with the description of the properties (of the instances of) each concept. That is, an ontology defines a representation of a shared conceptualization of a particular world (a conceptual schema). The Semantic Web is envisioned to contain several languages in which languages of increasing power are layered one on top of the other [42, 44]. The *OWL Web Ontology Language* [44] is currently the highest layer of sufficient maturity. OWL has three increasingly expressive sub-languages, namely *OWL Lite*, *OWL DL* and *OWL Full*, where OWL DL corresponds basically to DAML+OIL [41]. OWL Lite and OWL DL are essentially very expressive *Description Logics* (DLs) [4] [2] with an RDF syntax (see, e.g. [44, 39, 41, 42, 78]). DLs provide a simple well-established Tarski-style declarative semantics to capture the meaning of the most popular features of structured representation of knowledge.

Although OWL adds considerable expressive power to the Semantic Web it does have expressive limitations, particularly with respect to what can be said about properties. An intuitive way to overcome to these limitations is to extend current ontology languages with *rules*, i.e. to allow for building rules that use terms specified in ontologies. A first effort in this direction is *RuleML* [9, 8], fostering an XML-based markup language for rules and rule-based systems, while the *OWL Rules Language* [43] is a first proposal for extending OWL by *Horn clause rules*.

As first contribution of this paper, we propose, towards the integration of rules and ontologies in the Semantic Web, a framework for the combination of *Disjunctive Logic Programs* (LPs) under the answer set semantics [30, 56] with description logics. In particular, in this context the main contributions can be summarized as follows: we introduce the notion of *Description Logic Programs* (DLPs), which consists of a knowledge base (ontology) in description logic and a finite set of rules of disjunctive logic programs with explicit and negation as failure. Such rules are similar to the usual rules in logic programming, but may have

1. disjunctions in the head of a rule;
2. negation as failure in the head and the body of a rule;
3. explicit negation in the head and the body of a rule;
4. terms defined in the ontology in the head and the body of a rule and, thus, rules may contribute to the "intentional" definition of terms belonging to the ontology;
5. the body or the head of a rule may be empty and, thus, allows us to express constraints as well as facts.

The combination of DLs with LPs is not new (see, e.g. [20, 25, 33, 36, 43, 55]), however these differ significantly from our work, as we discuss in more detail on the related work later on. Rather than to propose another particular alternative language, our principal goal is to propose a *framework* for a principled integration of both representation paradigms based on description logic and logic programming, to which alternative formalizations may be compared in terms of computational complexity and/or expressive power.

[2] See the DL community home page http://dl.kr.org/.

As the Semantic Web grows in importance, resources will probably start to export their data and/or service descriptions according to some chosen ontology. Resource discovery, searching and resource integration (see, e.g. [3, 102, 54]), thus, become major challenges to information access for which accurate automated tools are desired.

As building an ontology is a time consuming and expensive process, the added value of both ontology-based document and/or service annotation and access to (integration of/search in) ontology-based resources should clearly compensate the enormous labour to construct it. Clearly, while the construction of an ontology and the description of Web services may accepted to be manual, the semantic-annotation of Web documents and the integration/access of heterogeneous resources *should be automatic in the long run and the large scale*. This is, without doubt a challenging and long term issue. Towards this end, we need a suitable representation language and likely, it should be able to deal with the *management of the inherent uncertainty* in the above tasks, as the following case exemplifies.

Let us assume that an agent A has to satisfy an information need Q_A expressed in a query language $\mathcal{L}_A$, whose basic terms belong to an ontology O_A, defined using the ontology language $\mathcal{O}_A$. Assume that there are a large amount of Semantic Web resources $\mathscr{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ accessible to A, where each resource $\mathcal{S}_i$ provides access to its objects by having its own ontology O_i, ontology language $\mathcal{O}_i$ and query language $\mathcal{L}_i$. Then the roughly an agent has to perform the following steps:

1. the agent has to *select* a subset of *relevant* (or, most promising) resources he is aware of, $\mathscr{S}' \subseteq \mathscr{S}$, as it is not reasonable to assume to access to and query all sources (*resource selection*);
2. for every selected resource $\mathcal{S}_i \in \mathscr{S}'$ the agent has to *reformulate* its information need Q_A into the query language $\mathcal{L}_i$ provided by the resource (*schema mapping*);
3. the results from the selected sources have to be merged together (*data fusion*).

That is, an agent must know *where to search, how to query different resources, and how to combine information from diverse resources*. The above problems have already been addressed recently, envisaging fully *automatic* processes, in the context of *Distributed Information Retrieval* and in the database area. Investigations addressed the problem both globally [3, 6, 71, 96, 101], as well as locally in terms of its sub-tasks -resource-selection [11, 28, 62, 74]; schema mapping [69, 70, 72]; data-fusion [12, 22, 75] and all proposed solutions require the management of the inherent uncertainty in the above tasks.

Our second contribution of this paper is the proposal of a framework, in which we extend DLPs towards the management of uncertainty to support reasoning agents in the Semantic Web. In particular, we propose to augment DLPs with annotation terms indicating to which degree an atom is certain, by relying on the work pioneered by [50]. As for DLPs, we define abstract syntax, semantics and discuss computational issues. The extension of DLPs to the management of uncertainty, to the best of our knowledge, has not been investigated yet.

We proceed as follows. We first briefly introduce the main notions related to description logics and disjunctive logic programs, and then show how both can be integrated, defining *Description Logic Programs* (DLPs). We then finally extend DLPs with the management of uncertainty, present related work and conclude.

2 Preliminaries

The OWL ontology languages OWL Lite and OWL DL are very close to expressive DLs, i.e. $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$, respectively [42]. For the sake of ease our presentation, we are not going to consider OWL DL as a whole, but restrict ourself to a significant subset of it. The specific DL we consider is $\mathcal{ALC}$ [79], a significant representative of DLs. $\mathcal{ALC}$ is sufficiently expressive to illustrate the main concepts introduced in this paper. We then recall disjunctive logic programs under the answer set semantics.

2.1 Description logics

2.1.1 Syntax.

We first describe the syntax of $\mathcal{ALC}$. Consider two alphabets of symbols, for *concepts names* (denoted A) and for *roles names* (denoted R) [3]. A *concept* (denoted C or D) of the language $\mathcal{ALC}$ is built inductively from concept names A and role names R according to the following syntax rule:

$$
\begin{array}{rcrl}
C, D & \longrightarrow & \top\mid & \text{(top concept)} \\
 & & \bot\mid & \text{(bottom concept)} \\
 & & A\mid & \text{(concept name)} \\
 & & C \sqcap D\mid & \text{(concept conjunction)} \\
 & & C \sqcup D\mid & \text{(concept disjunction)} \\
 & & \neg C\mid & \text{(concept negation)} \\
 & & \forall R.C\mid & \text{(universal quantification)} \\
 & & \exists R.C & \text{(existential quantification)}\ .
\end{array}
$$

A *terminology*, $\mathcal{T}$, is a finite set of concept inclusions or role inclusions, called *terminological axioms*, τ, where given two concepts C and D, and two role names R and R', a terminological axiom is an expression of the form $C \sqsubseteq D$ (D subsumes C) or of the form $R \sqsubseteq R'$ (R' subsumes R). We write $C = D$ (resp. $R = R'$) is place of the pair of axioms $C \sqsubseteq D$ and $D \sqsubseteq C$ (resp. $R \sqsubseteq R'$ and $R' \sqsubseteq R$).

2.1.2 Semantics.

An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non empty set $\Delta^{\mathcal{I}}$ (called the *domain*) and of an *interpretation function* $\cdot^{\mathcal{I}}$ mapping concepts names into subsets of $\Delta^{\mathcal{I}}$ and roles names into subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation of complex concepts is defined inductively as usual:

$$
\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \forall d'.(d, d') \notin R^{\mathcal{I}} \text{ or } d' \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists d'.(d, d') \in R^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}\ .
\end{array}
$$

[3] Metavariables may have a subscript or a superscript.

OWL DL Abstract Syntax	DL Syntax
Descriptions(C)	
A (URI reference)	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf($C_1 \dots C_n$)	$C_1 \sqcap C_2 \sqcap \dots \sqcap C_n$
unionOf($C_1 \dots C_n$)	$C_1 \sqcup C_2 \sqcup \dots \sqcup C_n$
complementOf(C)	$\neg C$
restriction(R someValuesFrom(C))	$\exists R.C$
restriction(R allValuesFrom(C))	$\forall R.C$
Object properties (R)	
R (URI reference)	R
Axioms	
Class(A partial $C_1 \dots C_n$)	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \dots \sqcap C_n$
Class(A complete $C_1 \dots C_n$)	$A = C_1 \sqcap C_2 \sqcap \dots \sqcap C_n$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses($C_1 \dots C_n$)	$C_1 = C_2 = \dots = C_n$
DisjointClasses($C_1 \dots C_n$)	$C_i \sqcap C_j = \bot, i \neq j$
ObjectProperty(R super(R_1) $\dots$ super(R_n)	$R \sqsubseteq R_i$
domain(C_1) $\dots$ domain(C_m)	$\exists R.\top \sqsubseteq C_i$
range(C_1) $\dots$ range(C_h))	$\top \sqsubseteq \forall R.C_i$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$
EquivalentProperties(R_1 R_2 $\dots$ R_n)	$R_1 = R_2 = \dots = R_n$

Table 1: Some OWL DL constructs and their DL counterpart

An interpretation $\mathcal{I}$ *satisfies* a terminological axiom $C \sqsubseteq D$ (resp. $R \sqsubseteq R'$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$), while $\mathcal{I}$ *satisfies* (is a *model* of) a terminology $\mathcal{T}$ iff $\mathcal{I}$ satisfies each element in $\mathcal{T}$.

The syntax we have presented is in typical DL style. For explicating the relationship between OWL DL and DLs, in Table 1 we report some OWL DL specific constructs, which can be mapped into DL expressions (see, e.g. [42]).

2.2 Disjunctive logic programs

2.2.1 Syntax.

We follow [30, 56]. Consider an arbitrary first order language that contains infinitely many variable symbols, finitely many constants and function symbols and predicate symbols [4]. We denote variables by lower case letters from the end of the alphabet, constants by lowercase letters from the beginning of the alphabet and predicates with capitals (as usual, metavariables may have a subscript or a superscript).

[4]However, often in logic programming function symbols are not considered for computational reasons.

A *term* t is inductively defined as either a constant, a variable or an expression of the form $f(t_1, \ldots, t_n)$, where all t_i are terms and f is a n-ary function symbol.

An *atom* is of the form $P(t_1, \ldots, t_n)$, where all t_i are terms and P is a n-ary predicate symbol. *Ground atoms* are atoms without variables. A *literal* L is either a *positive literal* $L = A$, or a *negative literal* $L = \neg A$, where A is an atom. A *ground literal* is a literal without variables. An *extended literal* is a literal or an expression of the form not(L) ("L is not provable"), where L is a literal. A *ground extended literal* is an extended literal without variables. For a set $\mathcal{X}$ of extended literals, $\mathcal{X}^- = \{\text{not}(L) \mid \text{not}(L) \in X\}$, while $\neg\mathcal{X} = \{\neg L \mid L \in \mathcal{X}, L \text{ literal}\}$, where we define $\neg\neg A = A$.

A *disjunctive logic program* $\mathcal{P}$, is a finite set of *rules* of the form

$$\gamma \leftarrow \delta \ ,$$

where γ and δ are finite sets of extended literals. For ease, we may omit the graph brackets in a rule. γ is called the head of the rules, while δ is called the body. The head is interpreted as the disjunction of its components, while the body is interpreted as a conjunction. A *fact* is a rule with empty body, while a *constraint* is a rule with empty head.

We call programs where for each rule $\gamma^- \cup \delta^- = \emptyset$, *programs without negation as failure (naf)*. Programs, where in each rule $|\gamma| = 1$ are called *normal*. Programs without naf, containing positive literals only, are called *positive*. Programs that do not contain variables are called *ground*.

2.2.2 Semantics.

For a program $\mathcal{P}$, and a (possibly infinite) non-empty set of terms H, such that every term, which may be constructed from the constants an function symbols appearing in $\mathcal{P}$, is in H, we call $\mathcal{P}_H$ the *grounded program* obtained from $\mathcal{P}$ by substituting every variable in $\mathcal{P}$ by every possible term in H. Note that $\mathcal{P}_H$ may contain an infinite number of rules (if H is infinite). The *universe* of a grounded program $\mathcal{P}$ is the (possibly infinite) non-empty set of terms $H_\mathcal{P}$ appearing in $\mathcal{P}$. Note that $H_{\mathcal{P}_H} = H$. The *base* of a grounded program $\mathcal{P}$ is the (possibly infinite) set $\mathcal{B}_\mathcal{P}$ of ground atoms that can be constructed using the predicate symbols in $\mathcal{P}$ with the terms in $H_\mathcal{P}$.

An *interpretation* I of a grounded program $\mathcal{P}$ is any consistent set of literals being a subset of $\mathcal{B}_\mathcal{P} \cup \neg\mathcal{B}_\mathcal{P}$. I *satisfies* a literal L iff $L \in I$. Furthermore, we say that I *satisfies* an extended literal not(L) iff I does not satisfy L, i.e. $L \notin I$.

An interpretation I of a grounded program $\mathcal{P}$ without naf *satisfies* a rule $\gamma \leftarrow \delta$ iff $\gamma \cap I \neq \emptyset$ whenever $\delta \subseteq I$. An interpretation I is a *model* of program $\mathcal{P}$ without naf iff it satisfies every rule in $\mathcal{P}$; I is a *minimal model* of $\mathcal{P}$ iff I is a model of $\mathcal{P}$ and there is no model $J \subset I$ of $\mathcal{P}$.

For a grounded program $\mathcal{P}$ and an interpretation I, the Gelfond-Lifschitz transformation [30, 56], is the program $\mathcal{P}^I$ without naf, obtained by deleting in $\mathcal{P}$,

1. each rule that has not(L) in its body with $L \in I$;
2. each rule that has not(L) in its head with $L \notin I$; and
3. all not(L) in the bodies and heads of the remaining rules.

Finally, an *interpretation* of a program $\mathcal{P}$ (possibly not grounded) is a pair $\mathcal{I} = (H, I)$, such that I is an interpretation of the grounded program $\mathcal{P}_H$. An interpretation $\mathcal{I} = (H, I)$ of a program $\mathcal{P}$ is a *stable model* of $\mathcal{P}$ iff I is a minimal model of $\mathcal{P}_H^I$. It

can easily be shown that, if $\mathcal{I} = (H, I)$ is a stable model of $\mathcal{P}$, then I is a model of $\mathcal{P}_H$ [5].

We say that a program $\mathcal{P}$ *entails* a ground extended literal L, denoted $\mathcal{P} \models L$ iff every stable model of $\mathcal{P}$ satisfies L. Note that $\mathcal{P} \models \text{not}(L)$, where L is a literal, if every stable model satisfying $\mathcal{P}$ *does not* satisfy L.

Notice that the universe H is not fixed to be the *Herbrand universe* of the terms which can be build from the constants and function symbols appearing in $\mathcal{P}$.

3 Description Logic Programs

In this section we introduce *Description Logic Programs* (DLPs), which are a novel combination of description logic terminologies with disjunctive logic programs.

3.1 Syntax.

A *Description Logic Program* (DLP) is a pair $\mathcal{DP} = \langle \mathcal{T}, \mathcal{P} \rangle$, where $\mathcal{T}$ is a terminology and $\mathcal{P}$ is a disjunctive logic program. Role names and concept names appearing in $\mathcal{T}$ may appear in the body as well as in the head of a rule $\gamma \leftarrow \delta \in \mathcal{P}$ and are managed as unary and binary predicates, respectively. Note that usually, concept and role names that appear in $\mathcal{T}$ are not allowed to appear in the head of the rules (except for representing facts) because of the underlying assumption that the terminological component *completely* describes the hierarchical structure in the domain, and, therefore, the rules should not allow to make new inferences about that structure. We do not impose this syntactical restriction, as from a semantics point of view it is unproblematic.

3.2 Semantics.

An *interpretation* for $\mathcal{DP} = \langle \mathcal{T}, \mathcal{P} \rangle$ is a pair $\mathcal{I} = (H, I)$, where $\mathcal{I}$ is an interpretation for $\mathcal{P}$ and $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation for $\mathcal{T}$, where $\Delta^{\mathcal{I}} = H$, and for concept names A and roles names R, $A^{\mathcal{I}} = \{t \mid A(t) \in I\}$ and $R^{\mathcal{I}} = \{(t, t') \mid R(t, t') \in I\}$, respectively. An interpretation $\mathcal{I} = (H, I)$ is a *model* of $\mathcal{DP} = \langle \mathcal{T}, \mathcal{P} \rangle$ iff

1. $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies $\mathcal{T}$;
2. I satisfies $\mathcal{P}_H^I$;
3. I is a as small as possible.

The definition of entailment is as usual.

Example 1 *Consider* $\mathcal{DP} = \langle \mathcal{T}, \mathcal{P} \rangle$*, with*

$$\begin{aligned} \mathcal{T} \quad = \{ \quad & A = \forall R.\neg C \\ & B = \forall R.D \\ & E = C \sqcap D \} \end{aligned}$$

$$\begin{aligned} \mathcal{P} \quad = \{ \quad & P(x) \leftarrow A(x) \\ & P(x) \leftarrow R(x, y), E(y), not(Q(y)) \\ & B(a) \leftarrow \} \ . \end{aligned}$$

[5] Note that $\mathcal{P}_H$ contains grounded extended literals.

Then, any model $\mathcal{I} = (H, I)$ of $\mathcal{DP}$ is such that $a \in H$, $B(a) \in I$, $Q(a) \notin I$ and either $A(a) \in I$ or $A(a) \notin I$, i.e. $a^{\mathcal{I}} \in A^{\mathcal{I}}$ or $a^{\mathcal{I}} \notin A^{\mathcal{I}}$. In the latter case, $a^{\mathcal{I}} \notin (\forall R.\neg C)^{\mathcal{I}}$, i.e. $a^{\mathcal{I}} \in (\exists R.C)^{\mathcal{I}}$ holds. As $a^{\mathcal{I}} \in B^{\mathcal{I}}$, it follows that $a^{\mathcal{I}} \in (\exists R.(C \sqcap D))^{\mathcal{I}}$ and, thus, $a^{\mathcal{I}} \in (\exists R.E)^{\mathcal{I}}$. As a consequence, any model of $\mathcal{DP}$ is one-to-one to $\mathcal{I}_i = (H, I_i)$ with universe H containing a and $I_1 = \{B(a), A(a), P(a)\}$, while $I_2 = \{B(a), R(a,b), C(b), D(b), E(b), P(a)\}$, for some $b \in H$. Indeed, in case of $\mathcal{I}_1$, we have that $\mathcal{P}_H^{I_1}$ is

$$\begin{array}{lll} \mathcal{P}_H^{I_1} & = \{ & P(a) \leftarrow A(a) \\ & & P(a) \leftarrow R(a,b), E(b) \\ & & B(a) \leftarrow \} \ . \end{array}$$

The it can be verified that

1. *$(\Delta^{I_1}, \cdot^{I_1})$ satisfies $\mathcal{T}$;*
2. *I_1 satisfies $\mathcal{P}_H^{I_1}$;*
3. *I_1 is a as small as possible.*

Notice that indeed I_1 is as small as possible. In fact, the counter example candidate $I' = \{B(a)\} \subset I_1$ is in effect a model of $\mathcal{P}_H^{I'}$, but then $(\Delta^{I'}, \cdot^{I'})$ does not satisfy $\mathcal{T}$. The argument for $\mathcal{I}_2$ is similar. Therefore, it follows that $\mathcal{DP} \models P(a)$.

Note that $\mathcal{DP} \not\models R(a,a)$, as the universe H is not fixed to be the Herbrand universe of $\mathcal{P}$. In case H is requested to be the Herbrand universe of $\mathcal{P}$ then $\mathcal{DP} \models R(a,a)$.

The following example shows a less abstract case based on the same principle of Example 1.

Example 2 *Consider the following database schema of a company database* [6]*. let $\mathcal{DP} = \langle \mathcal{T}, \mathcal{P} \rangle$ be such that*

$$\begin{array}{lll} \mathcal{T} & = \{ & \texttt{Employee} \sqsubseteq (\forall \texttt{OfficeMate}.\texttt{Employee}) \sqcap (\forall \texttt{SuperVised}.\texttt{Manager}) \\ & & \texttt{Manager} \sqsubseteq \texttt{Employee} \\ & & \texttt{Manager} = \texttt{AreaManager} \sqcup \texttt{TopManager} \\ & & \texttt{AreaManager} \sqcap \texttt{TopManager} \sqsubseteq \bot \} \end{array}$$

$$\begin{array}{lll} \mathcal{P} & = \{ & Q(x) \leftarrow \texttt{SuperVised}(x,y), \texttt{TopManager}(y), \texttt{OfficeMate}(y,z), \texttt{AreaManager}(z) \\ & & \texttt{SuperVised}(\texttt{john}, \texttt{jim}) \leftarrow \\ & & \texttt{SuperVised}(\texttt{john}, \texttt{mary}) \leftarrow \\ & & \texttt{OfficeMate}(\texttt{mary}, \texttt{jim}) \leftarrow \\ & & \texttt{OfficeMate}(\texttt{jim}, \texttt{paul}) \leftarrow \\ & & \texttt{Manager}(\texttt{jim}) \leftarrow \\ & & \texttt{TopManager}(\texttt{mary}) \leftarrow \\ & & \texttt{AreaManager}(\texttt{paul}) \leftarrow \} \ . \end{array}$$

$Q(x)$ acts as our query. It turns out that $\mathcal{DP} \models Q(\texttt{john})$, i.e. `john` is an answer to our query. Indeed, similarly as in Example 1, `jim` is either a `TopManager` or an `AreaManager`. In the former case `john` is supervised by the `TopManager jim`, which has the `AreaManager paul` as `OfficeMate`. In the latter case `john` is supervised by the `TopManager mary`, which has the `AreaManager jim` as `OfficeMate`.

[6] This example is taken from http://www.inf.unibz.it/∼franconi/dl/course/.

The following example shows an ontology integration problem as well as the problem of distributed search among different resources.

Example 3 *Informally, an ontology integration system may be seen as a triple* $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ *([54]), where* $\mathcal{G}$ *is the global ontology, while* $\mathcal{S}$ *are the source ontologies and* $\mathcal{M}$ *is mapping between* $\mathcal{S}$ *and* $\mathcal{G}$*. Under this view, let us consider the following scenario. Let us assume that there are two resources* R_1 *and* R_2 *in* $\mathcal{S}$ *based on the following ontologies. Let us assume that the objects in the resources are documents, which may semantically be annotated with terms of the ontologies (see, e.g. [61] for a formal framework), taken from the* Universal Decimal Classification *(UDC) system.* [7] *Essentially,* (i) R_1 *is about movies is general; and* (ii) R_2 *is about movies' comments.*

$$\begin{aligned}
\mathcal{T}_1 \quad = \{ \quad \texttt{M} \quad \sqsubseteq \quad & (\exists \texttt{MTitle.Text}) \sqcap \\
& (\exists \texttt{MYear.Year}) \sqcap \\
& (\exists \texttt{MDirector.Director}) \sqcap \\
& (\exists \texttt{MSummary.Text}) \} \ , \\
\\
\mathcal{T}_2 \quad = \{ \quad \texttt{MC} \quad \sqsubseteq \quad & (\exists \texttt{MovieID.Identifier}) \sqcap \\
& (\exists \texttt{MovieTitle.AnyText}) \sqcap \\
& (\exists \texttt{MovieComment.AnyText} \} \ .
\end{aligned}$$

Let us assume that a search agent is aware of the two resources and has its own ontology (which acts as the global ontology $\mathcal{G}$*):*

$$\begin{aligned}
\mathcal{T} \quad = \{ \qquad \texttt{Movie} \quad \sqsubseteq \quad & (\exists \texttt{HasCode.MovieCode}) \sqcap \\
& (\exists \texttt{HasTitle.TextDescription}) \sqcap \\
& (\exists \texttt{Year.Number}) \sqcap \\
& (\exists \texttt{About.TextDescription}) \\
& (\exists \texttt{HasDirector.MovieDirector}) \\
\\
\texttt{EuropeanDirector} \quad \sqsubseteq \quad & \texttt{MovieDirector} \sqcap \forall \texttt{Born.EuropeanCountry} \\
\\
\texttt{MovieReview} \quad \sqsubseteq \quad & (\exists \texttt{HasCode.MovieCode}) \sqcap \\
& (\exists \texttt{HasTitle.TextDescription}) \sqcap \\
& (\exists \texttt{HasCritique.TextDescription}) \}
\end{aligned}$$

We ask the search agent to search for title and critique of movies in 2003, conducted by European directors. Such a request may be written by means of the rule

$$\begin{aligned}
Q(t, r) \quad \leftarrow \quad & \texttt{Movie}(x), \\
& \texttt{HasTitle}(x, t), \\
& \texttt{Year}(x, 2003), \\
& \texttt{HasDirector}(x, y), \\
& \texttt{EuropeanDirector}(y), \\
& \texttt{MovieReview}(x), \\
& \texttt{HasCritique}(x, r)
\end{aligned}$$

[7] UDC is the world's foremost multilingual classification scheme for all fields of knowledge. See, http://www.udcc.org/.

In order to satisfy the request, the search agent should be able to determine a link between its own terminology and the one of the resources he is aware of. For instance, suppose we were able to determine a mapping (manually, automatically, semi-automatically [72]) between then agents ontology and the resources' ontologies and that it looks like as follows:

1. *the mapping to resource R_1 contains the rules*

$$\begin{aligned}
\texttt{Movie}(x) &\leftarrow \texttt{M}(x)\\
\texttt{HasTitle}(x,y) &\leftarrow \texttt{MTitle}(x,y)\\
\texttt{Year}(x,y) &\leftarrow \texttt{MYear}(x,y)\\
\texttt{HasDirector}(x,y) &\leftarrow \texttt{MDirector}(x,y)\\
\texttt{About}(x,y) &\leftarrow \texttt{MSummary}(x,y)\\
\texttt{MovieReview}(x) &\leftarrow \texttt{M}(x)\\
\texttt{HasCritique}(x,y) &\leftarrow \texttt{MSummary}(x,y)
\end{aligned}$$

2. *the mapping to resource R_2 contains the rules*

$$\begin{aligned}
\texttt{Movie}(x) &\leftarrow \texttt{MC}(x)\\
\texttt{HasCode}(x,y) &\leftarrow \texttt{MovieID}(x,y)\\
\texttt{HasTitle}(x,y) &\leftarrow \texttt{MovieTitle}(x,y)\\
\texttt{About}(x,y) &\leftarrow \texttt{MovieComment}(x,y)\\
\texttt{MovieReview}(x) &\leftarrow \texttt{MC}(x)\\
\texttt{HasCritique}(x,y) &\leftarrow \texttt{MovieComment}(x,y)\ .
\end{aligned}$$

It is easily verified that the above mapping suffices to solve the query in terms of calls to the resources R_i.

Less obvious in the above example, is how to solve queries asking a search agent, for instance, to find movies conducted by European directors "about the difficulties of life" or the critiques addressing the "social impact of a movie". It is evident that any relevant answer (a retrieved object) provided by a search agent to such requests is not a matter of just "crisp" yes or no decisions (a document satisfies a request or it does not satisfy it), but rather the answer is accompanied with a degree of certainty (the so-called *Retrieval Status Value* (RSV), in Information Retrieval terms [77, 5]), which is the agent's degree of believing an object to satisfy the request, and the list of answers is ordered according to their RSV. This is exactly what happens in current Web search engines and we do not expect that this will change either in case Web objects are semantically annotated as well. Furthermore, as described in the introduction, an agent must automatically know (*i*) where to search; (*ii*) how to query different resources; and (*iii*) how to combine information from diverse resources. As information resources continue to proliferate, these problems of resource selection, schema mapping and data fusion become major obstacles to information access for which accurate automated tools are desired (they are an ineffective manual task). Any successful solution to distributed search (of documents and services) in the Semantic

Web, like in Information Retrieval, should envisage a fully *automatic* process in the *large scale*. Towards this end, we need a suitable representation language and likely, it should be able to deal with the management of the inherent uncertainty in the above tasks. A preliminary language proposal towards this direction is the subject of the next section.

4 Description Logic Programs with Uncertainty

We are going now to define an extension of DLPs towards the management of uncertainty.

Classically, n-ary predicates may be seen as functions from their domain into $\{0, 1\}$, where 0 stands for $false$, while 1 stands for $true$. We extend this notion by mapping n-ary predicates into functions from their domain into the real unit interval $[0, 1]$. The associated value $c \in [0, 1]$ indicates to which *extent* a predicate is $true$.

4.1 Syntax.

From a syntax point of view, terminological axioms remain unchanged, i.e. of the form $C \sqsubseteq D$ (resp. $R \sqsubseteq R'$) or of the form $C = D$ (resp. $R = R'$). Unlike classical DLs, we will be able to specify to which extent $c \in [0, 1]$ (how certain it is that) a constant a is an instance of the concept C.

For rules, we have major modifications, which are inspired on the so-called *Generalized Annotated Logic Programs* (GAP) framework of Kifer and Subrahmanian [50]. So, let us define an *annotation function* of arity n to be a total and computable function [8] $f\colon ([0, 1])^n \to [0, 1]$. Assume a new alphabet of *annotation variables*, which will denote a value in $[0, 1]$ and can only appear in so-called *annotation terms*. An *annotation item*, κ, is defined inductively: (i) as a real $c \in [0, 1]$, or as an annotation variable (ν), or (ii) is of the form $f(\kappa_1, \ldots, \kappa_n)$, where f is an n-ary annotation function and all κ_i are annotation items. An *annotation term*, λ, is of the form $[\kappa, \kappa']$, where κ and κ' are annotation items. Annotation terms are supposed to denote subintervals of $[0, 1]$. Now, let L be a literal and λ an annotation term.

A *μliteral*, denoted μL, is of the form $L\colon \lambda$. The intended meaning is that "the certainty of L lies in the interval λ". An *extended μliteral* is of the form $\mathrm{not}(\mu L)$, where μL is a μliteral. The intended meaning of $\mathrm{not}(L\colon \lambda)$ is that "it is not provable that the certainty of L lies in the interval λ". For instance, following [61, 82], an image `i` may be annotated with the μliteral $\texttt{About}(\texttt{i}, \texttt{landscape})\colon [0.7, 1]$ indicating the extent to which an automated image annotation (classification) tool (see, e.g. [31, 93]) is certain that the image `i` is about a `landscape`.

A *μdisjunctive logic program* (or, simply μprogram), $\mu\mathcal{P}$, is a finite set of *μrules* of the form $\gamma \leftarrow \delta$, where γ and δ are finite sets of extended μliterals. Finally, a *μDescription Logic Program* (μDLP), denoted $\mu\mathcal{D}P$, is a pair $\langle \mathcal{T}, \mu\mathcal{P} \rangle$, where $\mathcal{T}$ is a terminology and $\mu\mathcal{P}$ is a μprogram.

Example 4 *Consider Example 3. Let us assume that the resource R_1 annotates the movies with* Dublin Core [9] *metadata records. Roughly a Dublin Core metadata record contains 15 attributes describing a document and consists of the attributes title, creator, subject, description, publisher, contributor, date, type, format, identifier, source,*

[8] The result of f is computable in a finite amount of time.

[9] http://dublincore.org/.

language, relation, coverage and rights. We may thus generate a schema mapping rule relating the term `About` *of the agent's ontology to* R_1 *as follows:*

$$\begin{aligned} \texttt{Map}(\texttt{about}, \texttt{title}) : [0.6, 1] \quad &\leftarrow \\ \texttt{Map}(\texttt{about}, \texttt{subject}) : [0.7, 1] \quad &\leftarrow \\ \texttt{Map}(\texttt{about}, \texttt{description}) : [0.9, 1] \quad &\leftarrow \end{aligned}$$

$$\begin{aligned} \texttt{About}(x, y) : [\nu, 1] \quad &\leftarrow \quad \texttt{Map}(\texttt{about}, \texttt{title}) : [\nu, 1], \texttt{Title}(x, y) : [\nu', 1] \\ \texttt{About}(x, y) : [v, 1] \quad &\leftarrow \quad \texttt{Map}(\texttt{about}, \texttt{subject}) : [\nu, 1], \texttt{Subject}(x, y) : [\nu', 1] \\ \texttt{About}(x, y) : [\nu, 1] \quad &\leftarrow \quad \texttt{Map}(\texttt{about}, \texttt{description}) : [\nu, 1], \texttt{Description}(x, y) : [\nu', 1] \end{aligned}$$

The first three facts represent the relationships (with their degree of certainty) an automated schema mapping tool was able to discover among the property `about` *and the properties* `title, subject` *and* `description`*. The second set of rules uses these discovered relationships to establish the mapping among the agent's ontology and the one of the resource* R_1*. Furthermore, keyword search may be supported by using rules like*

$$\texttt{SearchMovieAbout}(x, y) : [\nu \cdot \nu', 1] \quad \leftarrow \quad \texttt{About}(x, y') : [\nu, 1], \texttt{Sim}(y, y') : [\nu', 1]$$

In the above rule, the variable x *denotes a movie,* y *denotes the text to be searched for,* y' *denotes the text describing the content of the movie* x*, while* $\texttt{Sim}(y, y')$ *is a built-in predicate computing the similarity (using usual text similarity measures [5]) among the text denoted by* y *and* y' *and* ν' *is the computed RSV. Note that such a rule takes into account both the uncertainty of the schema mapping as well as the uncertainty of the "aboutness".*

4.2 Semantics.

We proceed as follows. We specify the semantics of the terminological component first and then the one of the rule component and conclude with defining the models of a μDLP.

The interpretation of concepts and roles and the semantics of terminological axioms is as in [83]. Formally, a *μinterpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function* mapping

- a concept C into a function $C^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \to [0, 1]$; and
- a role R into a function $R^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$.

If C is a concept then $C^{\mathcal{I}}$ will naturally be interpreted as the *membership degree function* of the concept C w.r.t. $\mathcal{I}$, i.e. if $d \in \Delta^{\mathcal{I}}$ is an object of the domain $\Delta^{\mathcal{I}}$ then $C^{\mathcal{I}}(d)$ gives us the certainty degree of being the object d an instance of the concept C under the μinterpretation $\mathcal{I}$. Similarly for roles.

The interpretation function $\cdot^{\mathcal{I}}$ has also to satisfy the following equations: for all $d \in \Delta^{\mathcal{I}}$,

$$
\begin{array}{rcl}
\top^{\mathcal{I}}(d) & = & 1 \\
\bot^{\mathcal{I}}(d) & = & 0 \\
(C \sqcap D)^{\mathcal{I}}(d) & = & \min(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d)) \\
(C \sqcup D)^{\mathcal{I}}(d) & = & \max(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d)) \\
(\neg C)^{\mathcal{I}}(d) & = & 1 - C^{\mathcal{I}}(d) \\
(\forall R.C)^{\mathcal{I}}(d) & = & \inf_{d' \in \Delta^{\mathcal{I}}}\{\max(1 - R^{\mathcal{I}}(d, d'), C^{\mathcal{I}}(d'))\} \\
(\exists R.C)^{\mathcal{I}}(d) & = & \sup_{d' \in \Delta^{\mathcal{I}}}\{\min(R^{\mathcal{I}}(d, d'), C^{\mathcal{I}}(d'))\} \ .
\end{array}
$$

These equations are the standard interpretation of conjunction, disjunction, negation and quantification, respectively.

Concerning terminological axioms, a μinterpretation $\mathcal{I}$ *satisfies* $C \sqsubseteq D$ iff for all $d \in \Delta^{\mathcal{I}}, C^{\mathcal{I}}(d) \leq D^{\mathcal{I}}(d)$. Similarly, μinterpretation $\mathcal{I}$ *satisfies* $R \sqsubseteq R'$ iff for all $\{d, d'\} \subseteq \Delta^{\mathcal{I}}, R^{\mathcal{I}}(d, d') \leq R'^{\mathcal{I}}(d, d')$. Finally, μinterpretation $\mathcal{I}$ *satisfies* (is a *model* of) a terminology $\mathcal{T}$ iff $\mathcal{I}$ satisfies each element in it, which concludes the semantics for terminological components.

Concerning μprograms we have the following definitions. In order to avoid straightforward repetition, if not stated otherwise, definitions related to μdisjunctive logic programs, parallels those for disjunctive logic programs. Furthermore, in grounding a μliteral $L\colon\lambda$, we assume that the annotation term λ is grounded as well, i.e. annotation variables are replaced with values in $[0, 1]$ and annotation items of the form $f(\kappa_1, \ldots, \kappa_n)$ are replaced with the result of the computation of $f(\kappa_1, \ldots, \kappa_n)$. Note that a grounded μprogram $\mu\mathcal{P}$ may contain an infinite number of rules due to the grounding of annotation terms. For a grounded μprogram $\mu\mathcal{P}$, $\mathcal{B}_{\mu\mathcal{P}}$ is the set of ground atoms A that can be constructed using the predicate symbols in $\mu\mathcal{P}$, where A is grounded with the terms in $H_{\mu\mathcal{P}}$ (annotations terms are not considered).

An *μinterpretation* I of a grounded μprogram $\mu\mathcal{P}$ is any (possibly partial) function $I\colon\mathcal{B}_{\mu\mathcal{P}} \rightarrow [0, 1]$ (*some ground atoms may be left unspecified*). The set of defined atoms in I is denoted $def(I)$. In the following, whenever we write $I(A)$, we assume that $A \in def(I)$. We extend I to literals $L = \neg A$ in the obvious way: $I(L) = 1 - I(A)$. A μinterpretation I *satisfies* a ground μliteral $L\colon\lambda$ iff $I(L) \in \lambda$. Note that we can always assume that μliterals are positive, by replacing $\neg A\colon[\kappa_1, \kappa_2]$ with $A\colon[\neg\kappa_2, \neg\kappa_1]$. Like for disjunctive logic programs, we say that I *satisfies* an extended μliteral not$(L\colon\lambda)$ iff I does not satisfy $L\colon\lambda$, i.e. $I(L) \notin \lambda$.

An μinterpretation I of a grounded program $\mu\mathcal{P}$ without naf *satisfies* a rule $\gamma \leftarrow \delta$ iff if I satisfies every μliteral in δ then I satisfies some μliteral in γ. An μinterpretation I is a *μmodel* of program $\mu\mathcal{P}$ without naf iff it satisfies every rule in $\mu\mathcal{P}$.

Example 5 *Consider the grounded μprogram $\mu\mathcal{P}$ without naf, with two facts*

$$
\begin{array}{l}
A\colon[0.2, 0.7], B\colon[0.3, 0.6] \leftarrow \\
C\colon[0.1, 0.3] \leftarrow
\end{array}
$$

Let us consider the following two partial functions $\mathscr{I}_1$ and $\mathscr{I}_2$, assigning to atoms intervals and defined as follows:

$$
\begin{array}{rcl}
\mathscr{I}_1(A) & = & [0.2, 0.7] \\
\mathscr{I}_1(C) & = & [0.1, 0.3]
\end{array}
$$

$$\begin{aligned} \mathscr{I}_2(B) &= [0.3, 0.6] \\ \mathscr{I}_2(C) &= [0.1, 0.3] \ . \end{aligned}$$

It is easily verified that for each μmodel I of $\mu\mathcal{P}$, we have that for some $i = 1, 2$, $def(\mathscr{I}_i) \subseteq def(I)$ and for all ground atoms $P \in def(\mathscr{I}_i), I(P) \in \mathscr{I}_i(P)$. Essentially, the $\mathscr{I}_i$ are minimal in terms of the atoms defined and the intervals are the 'most precise' intervals that can be inferred.

In the following we formally define the above concept of "interval interpretation". Let $\mathcal{C}[0, 1]$ be the set of all closed sub-intervals of $[0, 1]$. We also add $\emptyset$ to $\mathcal{C}[0, 1]$, called the empty interval. We will use it to manage inconsistencies among intervals. For two intervals σ_1 and σ_1 in $\mathcal{C}[0, 1]$, we define

$$\sigma_1 \preceq_p \sigma_2 \text{ iff } \sigma_2 \subseteq \sigma_1 \ .$$

Similarly, $\sigma_1 \prec_p \sigma_2$ iff $\sigma_2 \subset \sigma_1$ Furthermore, we define $\neg[c, c'] = [1 - c', 1 - c]$. The $\preceq_p$-least interval is $[0, 1]$, the $\preceq_p$-greatest interval is $\emptyset$.

An *interval interpretation* $\mathscr{I}$ of a grounded program $\mu\mathcal{P}$ without naf, is a (possibly partial) function $\mathscr{I}\colon \mathcal{B}_{\mu\mathcal{P}} \to \mathcal{C}[0, 1]$. An interval interpretation $\mathscr{I}$ is a representative of a whole family of μinterpretations I: we write $I \in \mathscr{I}$ iff $def(I) = def(\mathscr{I})$ and for all $A \in def(\mathscr{I})$, $I(A) \in \mathscr{I}(A)$ (see Example 5). We extend interval interpretations $\mathscr{I}$ to ground literals $L = \neg A$ as usual: $\mathscr{I}(L) = \neg\mathscr{I}(A)$.

For interval interpretations $\mathscr{I}_1$ and $\mathscr{I}_2$,

$$\begin{aligned} \mathscr{I}_1 \preceq_p \mathscr{I}_2 \quad \text{iff} \quad & def(\mathscr{I}_1) \subseteq def(\mathscr{I}_2) \\ & \text{and for all } A \in def(\mathscr{I}_1), \mathscr{I}_1(A) \preceq_p \mathscr{I}_2(A) \ . \end{aligned}$$

$\mathscr{I}_1 \prec_p \mathscr{I}_2$ iff $\mathscr{I}_1 \preceq_p \mathscr{I}_2$ and either $def(\mathscr{I}_1) \subset def(\mathscr{I}_2)$ or for some $A \in def(\mathscr{I}_1), \mathscr{I}_1(A) \prec_p \mathscr{I}_2(A)$. The $\preceq_p$-greatest interval interpretation, $\mathscr{I}_\top$, assigns to all ground atoms in $\mathcal{B}_{\mu\mathcal{P}}$ the empty interval, while the $\preceq_p$-least interval interpretation, $\mathscr{I}_\bot$ is undefined on all ground atoms in $\mathcal{B}_{\mu\mathcal{P}}$, i.e. $def(\mathscr{I}_\bot) = \emptyset$, meaning essentially that the certainty of all the atoms is *unknown*.

An interval interpretation $\mathscr{I}$ *satisfies* a ground μliteral $L\colon \lambda$ iff $\lambda \preceq_p \mathscr{I}(L)$. An interval interpretation $\mathscr{I}$ of a grounded program $\mu\mathcal{P}$ without naf *satisfies* a rule $\gamma \leftarrow \delta$ iff if $\mathscr{I}$ satisfies every μliteral in δ then $\mathscr{I}$ satisfies some μliteral in γ. An interval interpretation $\mathscr{I}$ is an *interval model* of program $\mu\mathcal{P}$ without naf iff it satisfies every rule in $\mu\mathcal{P}$. Furthermore, $\mathscr{I}$ is *minimal* as well iff there is no interval model $\mathscr{I}' \prec_p \mathscr{I}$ of $\mu\mathcal{P}$. For instance, in Example 5, $\mathscr{I}_1$ and $\mathscr{I}_2$ are the only two minimal interval models of $\mu\mathcal{P}$.

Like in [50], it can easily be shown that if $\mu\mathcal{P}$ is normal grounded program without naf, then there is an unique minimal interval model for $\mu\mathcal{P}$, which is the $\preceq_p$-least fixed-point, $lfp(T_{\mu\mathcal{P}})$, of the following $T_{\mu\mathcal{P}}$ monotone operator: for all $A \in \mathcal{B}_{\mu\mathcal{P}}$,

$$T_{\mu\mathcal{P}}(\mathscr{I})(A) = \bigcap\{\lambda \mid A\colon \lambda \leftarrow \delta \in \mu\mathcal{P}, \mathscr{I} \text{ satisfies each } \mu\text{literal in } \delta\} \ .$$

Note that $T_{\mu\mathcal{P}}$ is not continuous. In fact the grounded μprogram without naf, containing all ground instances of the rules

$$
\begin{array}{rcl}
A\colon [0,1] & \leftarrow & \\
A\colon [\frac{\nu+1}{2}, \nu'] & \leftarrow & A\colon [\nu, \nu'] \\
B\colon [1,1] & \leftarrow & A\colon [1,1]
\end{array}
$$

has unique minimal interval model $\mathscr{I}(A) = \mathscr{I}(B) = [1,1]$, which is obtained after $\omega+1$ $T_{\mu\mathcal{P}}$ iterations over $\mathscr{I}_\perp$, where ω is the first limit ordinal. Note also that the least and unique interval model $\mathscr{I}$ of the μprogram $\{(A\colon [0,0] \leftarrow), (A\colon [1,1] \leftarrow)\}$ assigns to A the empty set $\emptyset$, which indicates that on A the μprogram is inconsistent and, thus, has no μmodel. We may use this property to manage inconsistencies of this kind, by allowing interpretations to map ground atoms into a new symbol, indicating that the μprogram is inconsistent on that atom. We will not investigate this issue further in this paper.

Given a grounded μprogram (possibly with naf) $\mu\mathcal{P}$ and an μinterpretation I for $\mu\mathcal{P}$, the Gelfond-Lifschitz transformation, is the grounded positive μprogram $\mu\mathcal{P}^{\mathcal{I}}$, obtained by deleting in $\mu\mathcal{P}$,

1. each rule that has not(μL) in its body and I satisfies μL;
2. each rule that has not(μL) in its head and I does not satisfy μL;
3. all not(μL) in the bodies and heads of the remaining rules.

Finally, an *μinterpretation* of a μprogram $\mathcal{P}$ (possibly not grounded) is a pair $\mathcal{I} = (H, I)$, such that I is an μinterpretation of the grounded program $\mu\mathcal{P}_H$. An *μinterpretation* $\mathcal{I} = (H, I)$ of a μprogram $\mu\mathcal{P}$ is a *stable μmodel* of $\mu\mathcal{P}$ iff $I \in \mathscr{I}$ for a minimal interval model $\mathscr{I}$ of $\mu\mathcal{P}_H^I$. Finally, we say that a program $\mu\mathcal{P}$ *entails* a ground extended μliteral μL, denoted $\mu\mathcal{P} \models \mu L$ iff every stable μmodel of $\mu\mathcal{P}$ satisfies μL. For instance, in Example 5, we have that any stable μmodel I of $\mu\mathcal{P}$ is such that $I \in \mathscr{I}_1$ or $I \in \mathscr{I}_2$.

Example 6 *Consider the following μprograms $\mu\mathcal{P}_1, \mu\mathcal{P}_2, , \mu\mathcal{P}_3$ and $\mu\mathcal{P}_4$, where*

$$
\begin{array}{ll}
\mu\mathcal{P}_1 = \{r_1, r_2\}\ , & \mu\mathcal{P}_2 = \{r_1, r_3\} \\
\mu\mathcal{P}_3 = \{r_1, r_4\}\ , & \mu\mathcal{P}_4 = \{r_1, r_5\}
\end{array}
$$

and the rules r_i are

$$
\begin{array}{rcl}
r_1 : A\colon [0.6, 0.8] & \leftarrow & \\
r_2 : B\colon [0.4, 0.5] & \leftarrow & \mathit{not}(A\colon [0.2, 0.3]) \\
r_3 : B\colon [0.4, 0.5] & \leftarrow & \mathit{not}(A\colon [0.2, 0.7]) \\
r_4 : B\colon [\nu, \nu'] & \leftarrow & A\colon [\nu, \nu'] \\
r_5 : B\colon [\nu, \nu'] & \leftarrow & \mathit{not}(A\colon [\nu, \nu'])\ .
\end{array}
$$

It can be verified that for any stable μmodel I of $\mu\mathcal{P}_i$ we have that

1. *for $\mu\mathcal{P}_1$, $I(A) \in [0.6, 0.8]$ and $I(B) \in [0.4, 0.5]$. Therefore, $\mu\mathcal{P}_1 \models B\colon [0.4, 0.5]$;*
2. *for $\mu\mathcal{P}_2$, $I(A) \in [0.6, 0.8]$ and if $I(A) \in (0.7, 0.8]$ then $I(B) \in [0.4, 0.5]$ else I is undefined on B. Therefore, $\mu\mathcal{P}_2 \not\models B\colon [c, c']$, for any $c, c' \in [0, 1]$. But, $\mu\mathcal{P}_2 \models \mathit{not}(B\colon [c, c'])$ if $[c, c'] \cap [0.4, 0.5] = \emptyset$;*
3. *for $\mu\mathcal{P}_3$, $I(A) \in [0.6, 0.8]$ and $I(B) \in [0.6, 0.8]$. Therefore, $\mu\mathcal{P}_3 \models B\colon [0.6, 0.8]$.*

4. *for $\mu\mathcal{P}_4$, first note that for any I satisfying r_1, $I(A) \in [0.6, 0.8]$ holds. Therefore, $\mu\mathcal{P}_4^I = \{r_1\} \cup \{(B{:}\,[c, c'] \leftarrow) | I(A) \not\in [c, c'], c \leq c' \in [0, 1]\}$, whose least interval model $\mathscr{I}$ is such that $\mathscr{I}(A) = [0.4, 0.5]$ and $\mathscr{I}(B) = \emptyset$. So, $\mu\mathcal{P}_4$ has no stable model.*

We are ready now to define the semantics of μDLPs. An interpretation for a μDLP $\mu\mathcal{D}P$ is a pair $\mathcal{I} = (H, I)$, where $\mathcal{I}$ is a μinterpretation for the μprogram $\mu\mathcal{P}$ and $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a μinterpretation for the terminology $\mathcal{T}$, where $\Delta^{\mathcal{I}} = H$, and for concept names A and roles names R, $A^{\mathcal{I}}(t) = I(A(t))$ and $R^{\mathcal{I}}(t) = I(R(t, t'))$ $(t, t' \in H)$, respectively.

An interpretation $\mathcal{I} = (H, I)$ for a μDLP $\mu\mathcal{D}P = \langle \mathcal{T}, \mu\mathcal{P} \rangle$ is a *μmodel* of $\mu\mathcal{D}P$ iff

1. $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of the terminology $\mathcal{T}$;
2. $I \in \mathscr{I}$ and $\mathscr{I}$ is an interval model $\mathscr{I}$ of $\mu\mathcal{P}_H^I$;
3. $\mathscr{I}$ is a as small as possible.

Entailment is defined as usual.

Example 7 *Let us consider the DLP of Example 1, but extended with uncertainty as follows. $\mu\mathcal{D}P = \langle \mathcal{T}, \mathcal{P} \rangle$, where*

$$\begin{aligned} \mathcal{T} \quad &= \{ \quad A = \forall R.\neg C \\ &\qquad\; B = \forall R.D \\ &\qquad\; E = C \sqcap D\} \end{aligned}$$

$$\begin{aligned} \mu\mathcal{P} \quad &= \{ \quad B(a){:}\,[0.7, 1] \leftarrow \\ &\qquad\; P(x){:}\,[\nu, \nu'] \leftarrow A(x){:}\,[\nu, \nu'] \\ &\qquad\; P(x){:}\,[\nu, \nu'] \leftarrow R(x, y){:}\,[\nu, \nu'], E(y){:}\,[\nu, \nu'], \mathit{not}(R(y){:}\,[0, 1, 0.2])\} \end{aligned}$$

Consider the atom $P(a)$. Then, by reasoning by cases like in Example 1, it follows that $\mu\mathcal{D}P \models P(a){:}\,[0.5, 1]$. In fact, the μmodels $\mathcal{I} = (H, I)$ of $\mu\mathcal{D}P$ are such that $a \in H$ and either $def(I) = \{B(a), A(a), P(a)\}$ or $def(I) = \{B(a), R(a, b), C(b), D(b), E(b), P(b)\}$, for some $b \in H$. Furthermore, in either case we have $I(P(a)) \geq \max(c, \min(0.7, 1-c))$, for any $c \in [0, 1]$. As a consequence, for any μmodel $\mathcal{I} = (H, I)$ of $\mu\mathcal{D}P$, we have $I(P(a)) \geq 0.5$.

5 Reasoning support

5.1 Description logic programs

Reasoning in full DLPs is a difficult task, even for a small fragments of it [10]. In fact, from the results in [55], in which DLs are combined with function-free Horn rules, from [55, Theorem 5.1] it follows easily that the entailment problem in DLPs is undecidable. For instance, it suffices to allow "acyclic" terminologies with concept definitions only, i.e. of the form $A = C$ and there is no concept defined in terms of

[10] See for instance, [4] as a source of complexity results for DLs, [17, 23] as a source of complexity results for LPs.

it self, the $\forall R.C$ constructor and recursive function-free *Horn rules*, i.e. rules without naf, to guarantee undecidability.

In case we restrict our attention to DLPs in which Horn rules are considered only, we may easily translate DLPs into First-Order Logic (FOL) and use FOL theorem provers, like Vampire [76] to reason in the resulting first order theory. The use of FOL ensures that no restriction on the form of Horn rules and on the terminology is required and that all inferences were sound. In case decidability is desired, we have to restrict the form or the expressiveness of DLPs. [33, 55] provide many concrete suggestions in the restriction of the form of terminological axioms as well as of the form of rules to fit either within a tractable or decidable fragment of DLPs. On the other hand, [25] suggests to use a weaker semantics to be used to get decidable reasoning (see the section about related work).

In case of DLPs with naf, [25] shows concretely how to combine normal logic programs with naf with DLs, by weakening the semantics. In particular, concept and role predicates appearing in rules can be managed as system calls to a DL reasoner (in particular, RACER [11]). On the other hand, if we rely on the semantics given in this paper, of course the entailment problem is undecidable as well. Towards the support of decidable entailment, e.g. [84] shows that under certain reasonable circumstances we may reduce the entailment problem in DLPs into an entailment problem for disjunctive logic programs. This has the considerable practical advantage that deciding an entailment problem in DLPs may be deferred to a disjunctive logic program reasoner like DLV [19] or smodels [68].

5.2 Description logic programs with uncertainty

Reasoning in μDLP is further complicated by the introduction of annotation terms. For instance, the simple example showing the non-continuity of the $T_{\mu\mathcal{P}}$ immediate consequence operator is indicating an undecidability result for normal μprograms without naf, while their non-annotated counterpart is decidable (see, e.g. [24]). As a consequence, similarly as for DLPs, we have either to restrict the form or the expressiveness of μDLPs. An interesting source for suggestions, but applicable to normal rules without naf only, can be found in [50]. In the case of μDLPs *with naf*, another approach is shown in [84], where in particular it is shown that under certain restricted circumstances we may reduce the entailment problem in μDLPs into an entailment problem for disjunctive logic programs and, thus, state of the art disjunctive logic program reasoner may be applied. Of course, further investigations are required in this field.

6 Related work

While the combination of DLs with LPs is not new, to best of our knowledge, the integration of the management of uncertainty as well, has not been investigated yet.

Concerning μDLPs, the terminological component mainly relies on the work of Straccia [83], but it allows only a restricted form of terminological axioms. Concerning the rule component, as anticipated, it is inspired by the Generalized Annotated Logic Programming framework of Kifer and Subrahmanian [50], but extends it in two

[11] http://www.cs.concordia.ca/∼haarslev/racer/

directions: disjunctions may appear in the head of rules; negation-as-failure and explicitly negated atoms may appear in rules. The semantics we devised in that case closely relates the one of [65], but there a probabilistic setting has been considered.

Concerning the combination of DLs with LPs, they do not consider function symbols and, the works can roughly be divided into (*i*) hybrid approaches, which use description logics to specify structural constraints in the bodies of logic programs rules; and (*ii*) approaches that reduce description logic inference to logic programming. The basic idea behind (*i*) is to combine the semantic and computational strengths of the two systems, while the main rationale of (*ii*) is to use powerful logic programming technology for inference in description logics.

In the former case fall approaches like [20, 55, 43]. In particular, [20] combines plain Datalog (no disjunction and negation) with the description logic $\mathcal{ALC}$, where the integration lies in using concepts from the terminological component as constraints in rule bodies. It also presents a technique for answering conjunctive queries (existentially quantified conjunctions of atoms) with such constraints, where SLD resolution is integrated with an inference method for $\mathcal{ALC}$. More related to our approach is [55], which combines Horn rules with the description logic $\mathcal{ALCNR}$, where concepts and roles may appear in the body of Horn rules. Like in [20], [55] devices an SLD resolution integrated with an inference method for $\mathcal{ALCNR}$. Unfortunately, this approach is not easy applicable to full DLPs as negation-as-failure is present and rules heads are disjunctions. However, it may be applied to special cases, for which top-down resolution methods for the logic programming component are known (see, e.g. [14, 15, 73, 95]). Finally, [43] is similar to [55], except that the atoms of a rule refer to OWL classes and properties rather than to $\mathcal{ALCNR}$.

In the latter case (reducing description logic inference to function-free logic programming) fall approaches like [91, 1, 87, 33, 36, 63]. We remark that (*i*) [91] reduces knowledge bases in the DL $\mathcal{ALCN}$ [10] ($\mathcal{ALC}$ with number restrictions) to open logic programs; (*i*) [1, 87] reduce reasoning in the DL $\mathcal{ALCQI}$ [89, 88] ($\mathcal{ALC}$ with qualified number restrictions and inverse roles) to query answering from answer sets of normal logic programs; (*iii*) [33] shows how a subset of the DL $\mathcal{SHOIQ}$ [40, 39, 44] can be reduced to a subset of Horn programs (positive normal programs); (*iv*) [36] reduces the DL $\mathcal{SHIF}$ with transitive role closure to disjunctive logic programs (in [84] we use a similar reduction); and [63] reduces reasoning in the DL $\mathcal{SHIQ}^-$ to reasoning in disjunctive datalog programs by translating DL expressions into FOL clauses, saturates the clauses by resolution and translates the saturated clauses to disjunctive datalog programs.

Finally, not in the above classification fall approaches like [2, 25, 98, 99]. Roughly, [25] combines the DL $\mathcal{SHOIN}(D)$, which is $\mathcal{SHOIN}$ with concrete domains (e.g. strings, integers, reals), with normal logic programs. The main characteristics of [25] lies in the use of a weaker semantics to be used to get decidable reasoning (but, e.g. it does not answer positively to the query of Example 2). Indeed, concept and role predicates appearing in rules can be managed as system calls to a DL reasoner. [2] proposes defeasible reasoning in normal logic programs combined with a reduction of a subset of OWL Lite to normal logic programs in similar manner as [33]. Finally, [98] (and similarly in [99]) is based on *F-logic* [48, 49].

7 Conclusions

Towards the integration of rules and ontologies in the Semantic Web, we have proposed a framework in order to combine logic programming under the answer set semantics with description logics, which stand behind ontology languages like DAML+OIL, OWL Lite and OWL DL. We have defined the new family of Description Logic Programs (DLPs), which combine DLs with Disjunctive Logic Programs under the answer set semantics: concepts, roles, explicit negation and negation as failure may appear both in the head as well as in the body of rules. We defined their abstract syntax and semantics and discussed related computational issues. While the combination of DLs with positive logic programs (Horn logics) is not new, the integration of DLs with disjunctive logic programs has not been addressed yet. We hope that DLPs may become the reference language of the combination of the two representation paradigm to which alternative formalizations may be compared.

We further have shown that in the realistic setting of distributed search in the Semantic Web, where an agent is asked for searching relevant objects from the numerous information resources he has access to, there is an implicit requirement of uncertainty management at various levels: document content representation (annotation), document retrieval, automated resource selection, automated schema mapping and retrieval results merging. We have proposed a framework to extend DLPs towards this direction by proposing to augment DLPs with annotation terms indicating to which degree a literal is certain. As for DLPs, we defined their abstract syntax, semantics and discussed some computational issues.

The main direction for future work involves the computational aspect. Currently, there are three alternatives worth to be investigated to provide reasoning support to (subsets of) DLPs: (*i*) translating DLPs into disjunctive logic programming [91, 1, 87, 33, 36, 63]; (*ii*) weakening the semantics such that ontology concepts and properties are managed like procedural attachments [25, 64]; or (*iii*) to rely on methods, which combine logic programming SLD refutation with DLs tableaux like refutation methods [20, 55].

While there is already substantial work from which the computational issues addressed by DLPs may take advantage of, little is known concerning μDLPs and, thus, this may be an appealing area of discovery.

It might well be of interest to consider different approaches towards the management of uncertainty in DLPs, by relying on alternative notions of uncertainty. For instance, related to DLs, we may mention

probability theory: [32, 35, 46, 51, 80];

possibility theory [38];

fuzzy theory: [16, 37, 83, 90, 100];

multi-valued theory: [83, 85]

which have to be coupled with the corresponding rule component based on

probability theory: [18, 29, 52, 58, 65, 66, 67, 97];

possibility theory [21];

fuzzy theory: [13, 45, 81, 92, 94];

multi-valued theory: [26, 27, 47, 50, 53, 57, 59, 60, 86].

References

[1] Guray Alsaç and Chitta Baral. Reasoning in description logics using declarative logic programming. Technical report, Department of Computer Science and Engineering, Arizona State University, USA, 1997.

[2] G. Antoniou. Nonmonotonic rule systems on top of ontology layers. In *Proceedings of the 1st International Semantic Web Conference (ISWC-02)*, number 2663 in Lecture Notes in Computer Science, pages 394–398, Sardinia, Italia, 2002. Springer Verlag.

[3] A. Aslam, Javed and Mark Montague. Models of metasearch. In *Proceedings of the 24rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-01)*, pages 276–284, New Orleans, USA, 2001.

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

[5] Ricardo A. Baeza-Yates, R. Baeza-Yates, and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[6] Christoph Baumgarten. A probabilistic solution to the selection and fusion problem in distribute information retrieval. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-99)*, pages 246–256, Berkeley, CA USA, 1999.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *The Scientific American*, 284(5):34–43, 2001.

[8] H. Boley, B. Grosof, M. Sintek, S. Tabet, and G. Wagner. RuleML design. Sept. 2002. http://www.dfki.uni-kl.de/ruleml/indesign.html.

[9] Harold Boley, Said Tabet, and Gerd Wagner. Design rationale of RuleML: A markup language for semantic web rules. In *Proceedings of the 1st Semantic Web Working Symposium (SWWS-01)*, pages 105–113, Stanford, 2001.

[10] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. In *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*, pages 704–709, Chambery, France, 1993. Morgan Kaufmann, Los Altos.

[11] Jamie Callan. Distributed information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, Hingham,MA, USA, 2000.

[12] Jamie Callan, Zhihong Lu, and Bruce W. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-95)*, pages 21–28, Seattle, WA, 1995.

[13] True H. Cao. Annotated fuzzy logic programs. *Fuzzy Sets and Systems*, 113(2):277–298, 2000.

[14] Weidong Chen, Terrance Swift, and David Scott Warren. Efficient top-down computation of queries under the well-founded semantics. *Journal of Logic Programming*, 24(3):161–199, 1995.

[15] Weidong Chen and David S. Warren. Tabled evaluation with delaying for general logic programs. *Journal of the ACM*, 43(1):20–74, 1996.

[16] Rita Maria da Silva, Antonio Eduardo C. Pereira, and Marcio Andrade Netto. A system of knowledge representation based on formulae of predicate calculus whose variables are annotated by expressions of a fuzzy terminological logic. In *Proc. of the 5th Int. Conf. on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-94)*, number 945 in Lecture Notes in Computer Science. Springer-Verlag, 1994.

[17] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[18] Alex Dekhtyar and V.S. Subrahmanian. Hybrid probabilistic programs. In *Proc. of the 13th Int. Conf. on Logic Programming (ICLP-97)*, Leuven, Belgium, 1997. The MIT Press.

[19] Tina Dell'Armi, Wolfgang Faber, Giuseppe Ielpa, Christoph Koch, Nicola Leone, Simona Perri, and Gerald Pfeifer. System description: DLV. In *6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, number 2173 in Lecture Notes in Artificial Intelligence, pages 409–412, 2001.

[20] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[21] Didier Dubois, Jérome Lang, and Henri Prade. Towards possibilistic logic programming. In *Proc. of the 8th Int. Conf. on Logic Programming (ICLP-91)*, pages 581–595. The MIT Press, 1991.

[22] Cynthia Dwork, Ravi Kumar, moni Noar, and D. Sivakumar. Rank aggregation methods for the web. In *10th International Conference on the World Wide Web*, pages 613–622. ACM Press and Addison Wesley, 2001.

[23] Thomas Eiter and Georg Gottlob. Expressiveness of stable model semantics for disjunctive logic programs with functions. *Journal of Logic Programming*, 33(2):167–178, 1997.

[24] Thomas Eiter, Nicola Leone, and Domenico Saccá. Expressive power and complexity of partial models for disjunctive deductive databases. *Theoretical Computer Science*, 206(1–2):181–218, 1998.

[25] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-04)*. AAAI Press, 2004.

[26] Gonzalo Escalada-Imaz and Felip Manyà. Efficient interpretation of propositional multiple-valued logic programs. In *Proc. of the 5th Int. Conf. on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-94)*, number 945 in Lecture Notes in Computer Science, pages 428–439. Springer-Verlag, 1994.

[27] Melvin Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11:91–116, 1991.

[28] Norbert Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 3(17):229–249, 1999.

[29] Norbert Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110, 2000.

[30] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[31] Th. Gevers and A.W.M. Smeulders. Content-based image retrieval: An overview. In G. Medioni and S. B. Kang, editors, *Emerging Topics in Computer Vision*, page To appear. Prentice Hall, 2004.

[32] T. Rosalba Giugno and Thomas Lukasiewicz. P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, number 2424 in Lecture Notes in Artificial Intelligence, pages –. Springer-Verlag, 2002.

[33] Benjamin Grosof and Ian Horrocks. Description logics programs: Combining logic programs with description logics. In *The Twelfth International World Wide Web Conference (WWW-03)*, pages –, Budapest, Hungary, 2003.

[34] N. Guarino and R. Poli. Formal ontology in conceptual analysis and knowledge representation. *International Journal of Human and Computer Studies*, 43(5/6):625–640, 1995.

[35] Jochen Heinsohn. Probabilistic description logics. In R. Lopez de Mantara and D. Pool, editors, *Proceedings of the 10th Conference on Uncertainty in Artificila Intelligence*, pages 311–318, 1994.

[36] Stijn Heymans and Dirk Vermeir. Integrating description logics and answer set programming. In *Principles and Practice of Semantic Web Reasoning (PPSWR-03)*, number 2901 in Lecture Notes in Computer Science, pages 146–159, Mumbai, India, 2003. Springer Verlag.

[37] Steffen Hölldobler, Tran Dinh Khang, and Hans-Peter Störr. A fuzzy description logic with hedges as concept modifiers. In Nguyen Hoang Phuong, Hung T. Nguyen, Nguyen Cat Ho, and Pratit Santiprabhob, editors, *Proceedings InTech/VJFuzzy'2002*, pages 25–34, Hanoi, Vietnam, 2002. Institute of Information Technology, Vietnam Center for Natural Science and Technology, Science and Technics Publishing House, Hanoi, Vietnam.

[38] Bernhard Hollunder. An alternative proof method for possibilistic logic and its application to terminological logics. In *10th Annual Conference on Uncertainty in Artificial Intelligence*, Seattle, Washington, 1994. R. Lopez de Mantaras and D. Pool.

[39] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[40] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[41] Ian Horrocks. DAML+OIL: a description logic for the semantic web. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 25(1):4–9, March 2002.

[42] Ian Horrocks and Peter F. Patel-Schneider. Three theses of representation in the semantic web. In *The Twelfth International World Wide Web Conference (WWW-03)*, Budapest, Hungary, 2003.

[43] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW-04)*. ACM, 2004.

[44] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[45] Mitsuru Ishizuka and Naoki Kanai. Prolog-ELF: incorporating fuzzy logic. In *Proc. of the 9th Int. Joint Conf. on Artificial Intelligence (IJCAI-85)*, pages 701–703, Los Angeles, CA, 1985.

[46] Manfred Jäger. Probabilistic reasoning in terminological logics. In *Proceedings of KR-94, 5-th International Conference on Principles of Knowledge Representation and Reasoning*, pages 305–316, Bonn, FRG, 1994.

[47] M. Kifer and Ai Li. On the semantics of rule-based expert systems with uncertainty. In *Proc. of the Int. Conf. on Database Theory (ICDT-88)*, number 326 in Lecture Notes in Computer Science, pages 102–117. Springer-Verlag, 1988.

[48] Michael Kifer and Georg Lausen. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, pages 134–146. ACM Press, 1989.

[49] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of Object-Oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[50] Michael Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12:335–367, 1992.

[51] Daphne Koller, Alon Levy, and Avi Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proc. of the 14th Nat. Conf. on Artificial Intelligence (AAAI-97)*, pages 390–397, 1997.

[52] Laks V.S. Lakshmanan and Nematollaah Shiri. Probabilistic deductive databases. In *Int'l Logic Programming Symposium*, pages 254–268, 1994.

[53] Laks V.S. Lakshmanan and Nematollaah Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.

[54] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS-02)*, pages 233–246. ACM Press, 2002.

[55] Alon Y. Levy and Marie-Christine Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.

[56] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39–54, 2002.

[57] James J. Lu. Logic programming with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.

[58] Thomas Lukasiewicz. Probabilistic logic programming. In *Proc. of the 13th European Conf. on Artificial Intelligence (ECAI-98)*, pages 388–392, Brighton (England), August 1998.

[59] Cristinel Mateis. Extending disjunctive logic programming by t-norms. In *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-99)*, number 1730 in Lecture Notes in Computer Science, pages 290–304. Springer-Verlag, 1999.

[60] Cristinel Mateis. Quantitative disjunctive logic programming: Semantics and computation. *AI Communications*, 13:225–248, 2000.

[61] Carlo Meghini, Fabrizio Sebastiani, and Umberto Straccia. A model of multimedia information retrieval. *Journal of the ACM*, 48(5):909–970, 2001.

[62] W Meng, K.L Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. Determining text databases to search in the internet. In *Proc. of the 24th Int. Conf. on Very Large Data Bases (VLDB-98)*, pages 14–25, New York, USA, 1998.

[63] B. Motik, U. Sattler, and U. Hustadt. Reducing $\mathcal{SHIQ}^-$ description logic to disjunctive datalog programs. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-04)*. AAAI Press, 2004.

[64] Karen L. Myers. Hybrid reasoning using universal attachment. *Artificial Intelligence Journal*, (67):329–375, 1994.

[65] Raymond Ng and V.S. Subrahmanian. Stable model semantics for probabilistic deductive databases. In Zbigniew W. Ras and Maria Zemenkova, editors, *Proc. of the 6th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-91)*, number 542 in Lecture Notes in Artificial Intelligence, pages 163–171. Springer-Verlag, 1991.

[66] Raymond Ng and V.S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1993.

[67] Raymond Ng and V.S. Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. *Journal of Automated Reasoning*, 10(3):191–235, 1993.

[68] I. Niemelä and P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning*, pages 420–429. Springer-Verlag, 1997.

[69] H. Nottelmann and N. Fuhr. Learning probabilistic datalog rules for information classification and transformation. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM-01)*, pages 387–394, New York, 2001. ACM.

[70] H. Nottelmann and N. Fuhr. Combining DAML+OIL, XSLT and probabilistic logics for uncertain schema mappings in MIND. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL-03)*, 2003.

[71] Allison L. Powell, James C. French, and Jamie Callan. The impact of database selection on distributed searching. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR-00)*, pages 232–239, Athens, Greece, 2000.

[72] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[73] Prasad Rao, Konstantinos F. Sagonas, Terrance Swift, David Scott Warren, and Juliana Freire. XSB: A system for effciently computing WFS. In *Proceedings of Logic Programming and Non-monotonic Reasoning (LPNMR-97)*, number 1265 in Lecture Notes in Computer Science, pages 431–441. Springer-Verlag, 1997.

[74] Yves Rasolofo, Faïza Abbaci, and Jacques Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *Proc. of the 10th Int. Conf. on Information and Knowledge Management (CIKM-01)*, 2001.

[75] M. Elena Renda and Umberto Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *Proc. 18th Annual ACM Symposium on Applied Computing (SAC-03)*, pages 841–846, Melbourne, Florida, USA, 2003. ACM.

[76] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2):91–110, 2002.

[77] Gerard Salton. *Automatic Text Processing: The Tranformation, Analysis, and Retrieval of Information by Computer*. McGraw-Hill, New York, 1989.

[78] U. Sattler. Description logics for ontologies. In *Proc. of the International Conference on Conceptual Structures (ICCS 2003)*, "Lecture Notes in Artificial Intelligence", 2003. To appear.

[79] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[80] Fabrizio Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 122–130, Dublin, IRL, 1994. Published by Springer Verlag, Heidelberg, FRG.

[81] Ehud Y. Shapiro. Logic programs with uncertainties: A tool for implementing rule-based systems. In *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence (IJCAI-83)*, pages 529–532, 1983.

[82] Umberto Straccia. *Foundations of a Logic based approach to Multimedia Document Retrieval*. PhD thesis, Department of Computer Science, University of Dortmund, Dortmund, Germany, June 1999.

[83] Umberto Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[84] Umberto Straccia. Uncertainty in description logic programs. Technical Report ISTI-2004-TR-01, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy, 2004.

[85] Umberto Straccia. Uncertainty in description logics: a lattice-based approach. In *Proceedings of the 10th International Conference on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-04)*, 2004.

[86] V.S. Subramanian. On the semantics of quantitative logic programs. In *Proc. 4th IEEE Symp. on Logic Programming*, pages 173–182. Computer Society Press, 1987.

[87] Terrance Swift. Deduction in ontologies via ASP. In *Proceedings of the 7th International Conference in Logic Programming and Nonmonotonic Reasoning (LPNMR-04)*, number 2923 in Lecture Notes in Artificial Intelligence, pages 275–288, Fort Lauderdale, FL, USA, 2004. Springer Verlag.

[88] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[89] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH-Aachen, Germany, 2001.

[90] C. Tresp and R. Molitor. A description logic for vague knowledge. In *Proc. of the 13th European Conf. on Artificial Intelligence (ECAI-98)*, Brighton (England), August 1998.

[91] K. van Belleghem, M. Denecker, and D. de Schreye. A strong correspondence between description logics and open logic programs. In *Proc. of the 13th Int. Conf. on Logic Programming (ICLP-97)*, pages 346–360. The MIT Press, 1997.

[92] M.H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 4(1):37–53, 1986.

[93] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[94] Gerd Wagner. Negation in fuzzy and possibilistic logic programs. In T. Martin and F. Arcelli, editors, *Logic programming and Soft Computing*. Research Studies Press, 1998.

[95] Kewen Wang. A top-down procedure for disjunctive well-founded semantics. In *Proceedings of International Joint Conference on Automated Reasoning (IJCAR-01)*, number 2083 in LNAI, Siene, Italy, 2001. Springer.

[96] Zonghuan Wu, Weiyi Meng, Clement Yu, and Zhuogang Li. Towards a highly-scalable and effective metasearch engine. In *10th International Conference on the World Wide Web*, pages 386–395. ACM Press and Addison Wesley, 2001.

[97] Beat Wüttrich. Probabilistic knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):691–698, 1995.

[98] Guizhen Yang and Michael Kifer. Inheritance and rules in object-oriented semantic web languages. In *In Second International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML)*, Sanibel Island, Florida, 2003.

[99] Guizhen Yang, Michael Kifer, and Chang Zhao. Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE-03)*, Catania, Sicily, Italy, 2003.

[100] John Yen. Generalizing term subsumption languages to fuzzy logic. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 472–477, Sydney, Australia, 1991.

[101] Clement Yu, Weiyi Meng, King-Lup, Wensheng Wu, and Naphtali Rishe. Efficient and effective metasearch for a large number of text databases. In *Proc. of the 8th Int. Conf. on Information and Knowledge Management (CIKM-99)*, pages 217–224, 1999.

[102] Osmar R. Zaiane. From resource discovery to knowledge discovery on the internet. Technical report, Simon Fraser University, Canada, 1998.