Uncertainty Reasoning in Description Logics: A Generic Approach*

Volker Haarslev and **Hsueh-Ieng Pai** and **Nematollaah Shiri**
Concordia University
Dept. of Computer Science & Software Engineering
Montreal, Quebec, Canada
{haarslev, hsueh_pa, shiri}@cse.concordia.ca

Abstract

Description Logics (DLs) are gaining more popularity as the foundation of ontology languages for the Semantic Web. As most information in real life is imperfect, there has been an increasing interest recently in extending the expressive power of DLs with uncertainty, for which a number of frameworks have been proposed. In this paper, we introduce an extension of DL which unifies and/or generalizes a number of existing approaches for DLs with uncertainty. For this, we first provide a classification of existing frameworks for DLs with uncertainty on the basis of their underlying certainty formalisms. Using this as a basis, we introduce a generic framework for DL with uncertainty by extending components of the DL framework, i.e., the description language, the knowledge base, and the reasoning services.

Introduction

Uncertainty is a form of deficiency or imperfection commonly found in real-world information/data. A piece of information is uncertain if its truth is not established definitely. Modeling uncertainty and reasoning with it have been challenging issues for over two decades in database and artificial intelligence research. Uncertainty management has attracted the attention of researchers in Description Logics (DLs) (Baader, F. *et al.* 2003) in recent years. To highlight the importance of the family of DLs, we describe its connection with ontologies and Semantic Web as follows.

Ever since Tim Berners-Lee introduced the vision of the Semantic Web (Berners-Lee, T., Hendler, J., & Lassila, O. 2001), attempts have been made on making Web resources more machine-interpretable by giving them a well-defined meaning through semantic mark-ups. One way to encode such semantic mark-ups is using ontologies. An ontology is "an explicit specification of a conceptualization" (Gruber, T. R. 1993). Informally, ontology consists of a set of terms in a domain, the relationship between the terms, and a set of constraints imposed on the way in which those terms can be combined. Constraints such as concept conjunction, disjunction, negation, existential quantifier, and universal quantifier can all be expressed using ontology languages. By explicitly defining the relationships and constraints among the terms, the semantics of the terms can be better defined and understood.

Over the last few years, a number of ontology languages have been developed, most of which have a foundation based on DLs. The family of DLs is a subset of first-order logic (FOL) that is considered to be attractive as it keeps a good compromise between expressive power and computational tractability.

Despite popularity of standard DLs, it has been realized that they are inadequate to model uncertainty. For example, in the medical domain, one might want to express that: "It is very likely that an obese person would have heart disease", where "obese" is a vague concept that may vary across regions or countries, and "likely" shows the uncertain nature of this information. Such expressions cannot be expressed using standard DLs.

Recently, a number of frameworks have been proposed which extend DLs with uncertainty, some of which deal with vagueness while others deal with probabilistic knowledge. We do not intend to discuss which extension is better. In fact, different applications may require different aspects to be modelled, or in some cases, it may even be desired to model different aspects within the same application.

Following the approach of the parametric framework (Lakshmanan, L.V.S. & Shiri, N. 2001), in this paper, we propose a generic DL with uncertainty as a unifying umbrella for several existing frameworks of DLs with uncertainty. This approach not only provides a uniform access over theories that have been expressed using DL with various kinds of uncertainty, but also allows one to study various related problems, such as syntax, semantics, reasoning techniques and optimization, design, and implementation of knowledge bases in a framework-independent manner.

The rest of this paper is organized as follows. Next section provides an overview of the standard DL framework, and presents a classification of existing frameworks of uncertainty in DL. After that, we present our generic framework for DL with uncertainty in detail, and illustrate how it can represent uncertainty and reasoning services in the frameworks considered. Concluding remarks and future directions are presented in the last section.

*This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada, and by ENCS, Concordia University.

Background and Related Work

This section first gives an overview of the classical DL framework. Then, a classification of existing frameworks of uncertainty in DL is presented.

Overview of Classical DL Framework

The classical DL framework consists of three components:

1. *Description Language:* All description languages have elementary descriptions which include atomic concepts (unary predicates) and atomic roles (binary predicates). Complex descriptions can then be built inductively from concept constructors. In this paper, we focus on the description language $\mathcal{ALC}$ (Baader, F. *et al.* 2003). Let C and D be concept descriptions. $\mathcal{ALC}$ includes the atomic concepts A, atomic roles R, top/universal concept $\top$, bottom concept $\bot$, concept negation $\neg C$, concept conjunction $C \sqcap D$, concept disjunction $C \sqcup D$, role value restriction $\forall R.C$ (meaning $\forall y : R(x, y) \rightarrow C(y)$ for x in the domain), and role exists restriction $\exists R.C$ (meaning $\exists y : R(x, y) \wedge C(y)$ for x in the domain).
2. *Knowledge Base:* The knowledge base is composed of both intensional knowledge and extensional knowledge (see Fig. 1). The intensional knowledge includes the Terminological Box (TBox) consisting of a set of terminological axioms that could be concept subsumptions $C \sqsubseteq D$ and/or concept definitions $C \equiv D$ (where C and D are concepts), and the Role Box (RBox) consisting of a set of role axioms that could be role subsumptions $R \sqsubseteq S$ and/or role definitions $R \equiv S$ (where R and S are roles). On the other hand, the extensional knowledge includes the Assertional Box (ABox) consisting of a set of concept assertions $a : C$ (where a is an individual/instance and C is a concept) and/or role assertions $(a, b) : R$ (where a, b are individuals and R is a role).
3. *Reasoning Component:* A DL framework is equipped with reasoning services that enables one to derive implicit knowledge.

Approaches to DL with Uncertainty

On the basis of their mathematical foundation and the type of uncertainty modeled, we can classify existing proposals of DLs with uncertainty into three approaches: fuzzy, probabilistic, and possibilistic approach.

The fuzzy approach, based on fuzzy set theory, deals with the vagueness in the knowledge, where a proposition is true only to some degree. For example, the statement "Jason is obese with degree 0.4" indicates Jason is slightly obese. Here, the value 0.4 is the degree of membership that Jason is in concept obese.

The probabilistic approach, based on the classical probability theory, deals with the uncertainty due to lack of knowledge, where a proposition is either true or false, but one does not know for sure which one is the case. Hence, the certainty value refers to the probability that the proposition is true. For example, one could state that: "The probability that Jason would have heart disease given that he is obese lies in the range $[0.8, 1]$."

Finally, the possibilistic approach, based on possibility theory (Zadeh, L. A. 1978), allows both certainty (necessity measure) and possibility (possibility measure) be handled in the same formalism. For example, by knowing that "Jason's weight is above 80 kg", the proposition "Jason's weight is 80 kg" is necessarily true with certainty 1, while "Jason's weight is 90 kg" is possibly true with certainty 0.5.

Our Generic Framework

To support uncertainty, each component of the DL framework needs to be extended (see Fig. 1). To be more specific, the generic framework consists of:

1. *Description Language with Uncertainty*
2. *Knowledge Bases with Uncertainty*
3. *Reasoning with Uncertainty*

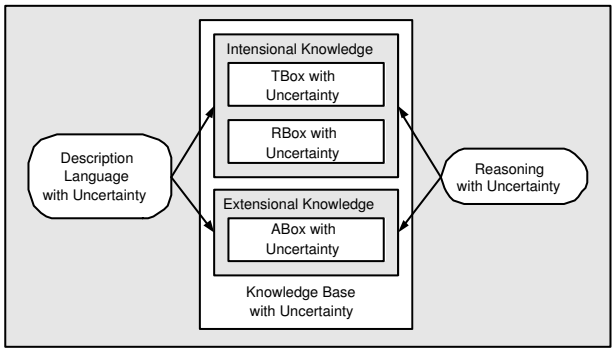

Figure 1: DL Framework with Uncertainty

In what follows, we discuss each of these three components in detail, along with illustrating examples. Note that this paper extends our previous work (Haarslev, V., Pai, H.-I., & Shiri, N. 2005) by presenting uncertainty inference rules for the reasoning component of the framework.

Description Language with Uncertainty

To develop a generic description language with uncertainty, we first need to represent certainty values, and then assign semantics to elements in the extended language.

Representation of Certainty Values. To represent the certainty values, we take a *lattice-based approach*, following the parametric framework (Lakshmanan, L.V.S. & Shiri, N. 2001). That is, we assume that certainty values form a complete lattice shown as $\mathcal{L} = \langle \mathcal{V}, \preceq \rangle$, where $\mathcal{V}$ is the certainty domain, and $\preceq$ is the partial order defined on $\mathcal{V}$. We also use b to denote the bottom or least element in $\mathcal{V}$, and use t to denote the top or greatest value in $\mathcal{V}$. The least upper bound operator (the join operator) in $\mathcal{L}$ is denoted by $\oplus$, its greatest lower bound (the meet operator) is denoted by $\otimes$, and its negation operator is denoted by $\sim$.

The certainty lattice can be used to model both *qualitative* and *quantitative* certainty values. An example for the former is classical logic which uses the binary values $\{0, 1\}$. For the latter, an example would be a family of multi-valued logics such as fuzzy logic which uses $[0, 1]$ as the certainty domain.

Assignment of Semantics to Description Language. The generic framework treats each type of uncertainty formalism as a special case. Hence, it would be restrictive to consider any specific function to describe the semantics of the description language constructors (e.g., fixing *min* as the function to determine the certainty of concept conjunction). An alternative is proposed in our generic framework to allow a user to specify the functions that are appropriate to define the semantics of the description language element at axiom or assertion level. We elaborate more on this later in section "Knowledge Bass with Uncertainty."

To ensure that the combination functions specified by a user make sense, we assume the following properties for various certainty functions to be reasonable. Most of these properties were recalled from (Lakshmanan, L.V.S. & Shiri, N. 2001), and are reasonable and justified when we verify them against existing extensions of DL with uncertainty. To present these properties, we consider the description language constructors in $\mathcal{ALC}$. We assume that the reader has a basic knowledge about $\mathcal{ALC}$.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation, where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ is an interpretation function that maps description language elements to some certainty value in $\mathcal{V}$.

Atomic Concept. The interpretation of an atomic concept A is a certainty value in the certainty domain, i.e., $A^{\mathcal{I}}(a) \in \mathcal{V}$, for all individuals $a \in \Delta^{\mathcal{I}}$. For example, in the fuzzy approach, the interpretation of an atomic concept A is defined as $A^{\mathcal{I}}(a) \in [0, 1]$, that is, the interpretation function assigns to every individual a in the domain, a value in the unit interval that indicates its membership to A.

Atomic Role. Similar to atomic concepts, the interpretation of an atomic role R is a certainty value in the certainty domain, i.e., $R^{\mathcal{I}}(a, b) \in \mathcal{V}$, for all individuals $a, b \in \Delta^{\mathcal{I}}$.

Top/Universal Concept. The interpretation of the top or universal concept $\top$ is the greatest value in $\mathcal{V}$, that is, $\top^{\mathcal{I}} = t$. For instance, $\top$ corresponds to 1 (true) in standard logic with truth values $\{0, 1\}$, as well as in any one of its extensions to certainty domain $[0, 1]$.

Bottom Concept. The interpretation of the concept bottom $\bot$ is the least value in the certainty domain $\mathcal{V}$, that is, $\bot^{\mathcal{I}} = b$. This corresponds to false in standard logic with $\mathcal{V} = \{0, 1\}$, or corresponds to 0 when $\mathcal{V} = [0, 1]$.

Concept Negation. Given a concept C, the interpretation of concept negation $\neg C$ is defined by the negation function $\sim: \mathcal{V} \rightarrow \mathcal{V}$, which satisfies the following properties:

- Boundary Conditions: $\sim b = t$ and $\sim t = b$.
- Double Negation: $\sim(\sim\alpha) = \alpha$, for all $\alpha \in \mathcal{V}$.

In our work, we consider the negation operator $\sim$ in the certainty lattice as the default negation function. Other properties, such as monotonicity (i.e., $\forall\alpha,\ \beta\ \in\ \mathcal{V},\ \sim\alpha \succeq \sim\beta$, whenever $\alpha \preceq \beta$) may be imposed if desired. A common interpretation of $\neg C$ is $1 - C^{\mathcal{I}}(a)$, for all $a \in C$.

Before introducing the properties of combination functions which are appropriate to describe the semantics of concept conjunction and disjunction, we first identify a set of desired properties which an allowable *combination function* f should satisfy. These functions are used to combine a collection of certainty values into one value. We then identify a subset of these properties suitable for describing the semantics of logical formulas on the basis of concept conjunction and disjunction. Note that, since f is used to combine a collection of certainty values into one, we describe f as a binary function from $\mathcal{V} \times \mathcal{V}$ to $\mathcal{V}$. This view is clearly without loss of generality and, at the same time, useful for implementing functions in general.

1. Monotonicity: $f(\alpha_1, \alpha_2) \preceq f(\beta_1, \beta_2)$, whenever $\alpha_i \preceq \beta_i$, for $i = 1, 2$.
2. Bounded Above: $f(\alpha_1, \alpha_2) \preceq \alpha_i$, for $i = 1, 2$.
3. Bounded Below: $f(\alpha_1, \alpha_2) \succeq \alpha_i$, for $i = 1, 2$.
4. Boundary Condition (Above): $\forall\alpha \in \mathcal{V}$, $f(\alpha, b) = \alpha$ and $f(\alpha, t) = t$.
5. Boundary Condition (Below): $\forall\alpha \in \mathcal{V}$, $f(\alpha, t) = \alpha$ and $f(\alpha, b) = b$.
6. Commutativity: $\forall\alpha, \beta \in \mathcal{V}$, $f(\alpha, \beta) = f(\beta, \alpha)$.
7. Associativity: $\forall\alpha, \beta, \delta \in \mathcal{V}$, $f(\alpha, f(\beta, \delta)) = f(f(\alpha, \beta), \delta)$.

Concept Conjunction. Given concepts C and D, the interpretation of concept conjunction $C \sqcap D$ is defined by the conjunction function f_c that should satisfy properties 1, 2, 5, 6, and 7. The monotonicity property is required so that the reasoning is monotone, i.e., whatever that has been proven so far will remain true for the rest of the reasoning process. The bounded value property is included so that the interpretation of the certainty values makes sense. Note that this property also implies the boundary condition (property 5). The commutativity property supports reordering of the arguments of the conjunction operator, and associativity ensures that a different evaluation order of a conjunction of concepts does not change the result. These properties are useful during the runtime evaluation used by the reasoning procedure. Examples of conjunctions include the usual product $\times$ and min functions, and bounded difference defined as $bDiff(\alpha, \beta) = max(0, \alpha + \beta - 1)$.

Concept Disjunction. Given concepts C and D, the interpretation of concept disjunction $C \sqcup D$ is defined by the disjunction function f_d that should satisfy properties 1, 3, 4, 6, and 7. The monotonicity, boundedness, boundary condition, commutativity, and associativity properties are required for similar reasons described in the conjunction case. Some common disjunction functions are: the standard max function, the probability independent function defined as $ind(\alpha, \beta) = \alpha + \beta - \alpha\beta$, and the bounded sum function defined as $bSum(\alpha, \beta) = min(1, \alpha + \beta)$.

Role Value Restriction. Given a role R and a role filler C, the interpretation of the "role value" restriction $\forall R.C$ is defined as follows:

$\forall a \in \Delta^{\mathcal{I}}, (\forall R.C)^{\mathcal{I}}(a) = \otimes_{b \in \Delta^{\mathcal{I}}}\{f_d(\sim R^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b))\}$

The intuition behind this definition is to view $\forall R.C$ as the open first order formula $\forall b.\ R(a, b) \rightarrow C(b)$, where $R(a, b) \rightarrow C(b)$ is equivalent to $\neg R(a, b) \vee C(b)$, and $\forall$ is viewed as a conjunction over certainty values associated with $R(a, b) \rightarrow C(b)$. To be more specific, the semantics of $\neg R(a, b)$ is captured using the negation function $\sim$ as $\sim R^{\mathcal{I}}(a, b)$, the semantics of $\neg R(a, b) \vee C(b)$ is captured using the disjunction function as $f_d(\sim R^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b))$,

and $\forall b$ is captured using the meet operator in the lattice $\otimes_{b\in\Delta^{\mathcal{I}}}$.

Role Exists Restriction. Given a role R and a role filler C, the interpretation of the "role exists" restriction $\exists R.C$ is defined as follows:

$\forall a \in \Delta^{\mathcal{I}}, (\exists R.C)^{\mathcal{I}}(a) = \oplus_{b\in\Delta^{\mathcal{I}}}\{f_c(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))\}$
The intuition here is that we view $\exists R.C$ as the open first order formula $\exists b.\ R(a,b) \wedge C(b)$, where $\exists$ is viewed as a disjunction over the elements of the domain. To be more specific, the semantics of $R(a,b) \wedge C(b)$ is captured using the conjunction function as $f_c(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))$, and $\exists b$ is captured using the join operator in the lattice $\oplus_{b\in\Delta^{\mathcal{I}}}$.

Additional Inter-Constructor Properties. In addition to the aforementioned properties, we further assume that the following inter-constructor properties hold:

- De Morgan's Rule: $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$ and $\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$.
- Pushing Negation In: $\neg\ \exists R.C \equiv \forall R.\neg C$ and $\neg\ \forall R.C \equiv \exists R.\neg C$.

The above two rules are needed to convert a concept description into *negation normal form* (NNF), i.e., the negation operator appears only in front of a concept name. Note that these properties restrict the type of negation, conjunction, and disjunction functions allowed in existing frameworks, and hence in our work.

Knowledge Bases with Uncertainty

As in the classical counterpart, a *knowledge base* Σ in the generic framework is a triple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$, where $\mathcal{T}$ is a TBox, $\mathcal{R}$ is an RBox, and $\mathcal{A}$ is an ABox. To develop a generic extension to the knowledge base, we present syntactical and semantical extensions to both the intensional (TBox and RBox) and extensional knowledge (ABox).

A TBox $\mathcal{T}$ consists of a set of terminological axioms expressed in the form $\langle C \sqsubseteq D, \alpha\rangle\langle f_c, f_d\rangle$ or $\langle C \equiv D, \alpha\rangle\langle f_c, f_d\rangle$, where C and D are concepts, $\alpha \in \mathcal{V}$ is the certainty that the axiom holds, f_c is the conjunction function used as the semantics of concept conjunction and part of the role exists restriction, and f_d is the disjunction function used as the semantics of concept disjunction and part of the role value restriction. As usual, the concept definition $\langle C \equiv D, \alpha\rangle\langle f_c, f_d\rangle$ is defined as $\langle C \sqsubseteq D, \alpha\rangle\ \langle f_c, f_d\rangle$ and $\langle D \sqsubseteq C, \alpha\rangle\langle f_c, f_d\rangle$.

In order to transform the axiom of the form $\langle C \sqsubseteq D, \alpha\rangle\ \langle f_c, f_d\rangle$ into its normal form, $\langle \top \sqsubseteq \neg C \sqcup D, \alpha\rangle\langle f_c, f_d\rangle$, we restrict the semantics of the concept subsumption to be $f_d(\sim C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$, where $\sim C^{\mathcal{I}}(a)$ captures the semantics of $\neg C$, and f_d captures the semantics of $\sqcup$ in $\neg C \sqcup D$. We say an interpretation $\mathcal{I}$ satisfies $\langle C \sqsubseteq D, \alpha\rangle\langle f_c, f_d\rangle$ iff for all individuals $a \in \Delta^{\mathcal{I}}$, $(f_d(\sim C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))) \in \alpha$. This definition of concept subsumption guarantees some basic properties to hold, such as the De Morgan's and Pushing Negation In rules described above.

The RBox $\mathcal{R}$ of a knowledge base Σ is similar to the TBox except that we have role axioms instead of terminological axioms. In addition, no conjunction or disjunction functions are specified. Since existing DL frameworks with uncertainty do not allow role conjunction or role disjunction, we do not consider them in the generic framework either. We also remark that since this generic framework supports only $\mathcal{ALC}$, no role hierarchy is allowed. However, we include the definition of a RBox here for completeness.

An ABox $\mathcal{A}$ of Σ consists of a set of assertions of the form $\langle a : C, \alpha\rangle\langle f_c, f_d\rangle$ or $\langle (a,b) : R, \alpha\rangle\langle -, -\rangle$, where a and b are individuals, C is a concept, R is a role, $\alpha \in \mathcal{V}$, f_c is the conjunction function, f_d is the disjunction function, and $-$ denotes that the corresponding combination function is not applicable. An interpretation $\mathcal{I}$ satisfies $\langle a : C, \alpha\rangle\langle f_c, f_d\rangle$ (resp. $\langle (a,b) : R, \alpha\rangle\langle -, -\rangle$) iff $C^{\mathcal{I}}(a) \in \alpha$ (resp. $R^{\mathcal{I}}(a,b) \in \alpha$).

An interpretation $\mathcal{I}$ *satisfies* a knowledge base Σ, denoted $\mathcal{I} \models \Sigma$, iff it satisfies each component of Σ. We say that Σ is *satisfiable*, denoted $\Sigma \not\models \bot$, iff there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$. Similarly, Σ is *unsatisfiable*, denoted $\Sigma \models \bot$), iff $\mathcal{I} \not\models \Sigma$, for all interpretations $\mathcal{I}$.

Reasoning with Uncertainty

In this section, we describe the reasoning procedure for the generic framework proposed here. Let $\langle \mathcal{T}, \mathcal{A}\rangle$ be a knowledge base, where $\mathcal{T}$ is an acyclic TBox and $\mathcal{A}$ is an ABox.

Satisfiability Problem: To check if a knowledge base $\langle \mathcal{T}, \mathcal{A}\rangle$ is satisfiable, first apply the pre-processing steps (described below) to remove the TBox, $\mathcal{T}$. Then, initialize the extended ABox, $\mathcal{A}_0^{\mathcal{E}}$, with the resulting ABox (i.e., the one after pre-processing steps are performed), and initialize the constraint set, $\mathcal{C}_0$, to the empty set $\{\}$. After that, apply the completion rules (described below) to transform the ABox into a "simpler" and satisfiability preserving one. The completion rules are applied in arbitrary order as long as possible, until either $\mathcal{A}_i^{\mathcal{E}}$ contains a clash or no further rule could be applied to $\mathcal{A}_i^{\mathcal{E}}$. If $\mathcal{A}_i^{\mathcal{E}}$ contains a clash, the knowledge base is unsatisfiable. Otherwise, an optimization method is applied to solve the system of inequations in $\mathcal{C}_j$. If the system of inequations is unsolvable, the knowledge base is unsatisfiable. Otherwise, the knowledge base is satisfiable.

Entailment Problem: To determine to what degree is an assertion X true, given a knowledge base $\langle \mathcal{T}, \mathcal{A}\rangle$, we are interested in finding the tightest bound for which X is true. To do so, we follow the same procedure as the one for checking satisfiability. However, instead of checking whether the system of inequations is solvable, we apply the optimization method to find the tightest bound for which X is true.

Pre-processing Steps: Before performing any inference procedure on the knowledge base, we do the following preprocessing steps.

1. Replace each axiom of the form $\langle C \equiv D, \alpha\rangle\langle f_c, f_d\rangle$ with the following two equivalent axioms: $\langle C \sqsubseteq D, \alpha\rangle\ \langle f_c, f_d\rangle$ and $\langle D \sqsubseteq C, \alpha\rangle\ \langle f_c, f_d\rangle$.
2. Transform every axiom in the TBox $\mathcal{T}$ into normal form. That is, replace each axiom of the form $\langle C \sqsubseteq D, \alpha\rangle\langle f_c, f_d\rangle$ with $\langle \top \sqsubseteq \neg C \sqcup D, \alpha\rangle\ \langle f_c, f_d\rangle$.
3. Transform every concept (including the ones in TBox and ABox) into negation normal form.

4. For each individual a in the ABox $\mathcal{A}$ and each axiom $\langle \top \sqsubseteq \neg C \sqcup D, \alpha \rangle \ \langle f_c, f_d \rangle$ in the TBox $\mathcal{T}$, add $\langle a : \neg C \sqcup D, \alpha \rangle \ \langle f_c, f_d \rangle$ to $\mathcal{A}$.
5. Apply the clash trigger (described below) to check if the initial knowledge base is inconsistent.

Completion Rules: As in the classical DL, completion rules are a set of consistency preserving transformation rules that allows us to infer implicit knowledge from the explicit one (i.e., the one specified in the original set of assertions specified in the ABox). In our generic framework, we have specified the following completion rules: clash triggers, concept assertion rule, role assertion rule, negation rule, conjunction rule, disjunction rule, role exists restriction rule, and role value restriction rule. Due to the limited space, we describe only some of the rules used in our illustrating example. The concept assertion and role assertion rules are omitted here since they simply add the certainty value of each atomic assertion to the constraint set $\mathcal{C}_j$.

Let α, β be certainty values in the certainty domain. Also let x_X be the variable denoting the certainty of assertion X, and Γ be either a certainty value in the certainty domain or an expression over certainty variables and values. The completion rules are defined as follows.

Clash Triggers:
$\langle a : \bot, t \rangle \langle -, - \rangle \in \mathcal{A}_i^{\mathcal{E}}$
$\langle a : \top, b \rangle \langle -, - \rangle \in \mathcal{A}_i^{\mathcal{E}}$
$\{\langle a : A, \alpha \rangle \langle -, - \rangle, \langle a : A, \beta \rangle \langle -, - \rangle\} \subseteq \mathcal{A}_i^{\mathcal{E}}$,
with $\otimes(\alpha, \beta) = \varnothing$

The purpose of these clash triggers is to detect any possible contractions in the knowledge base. Note that we use $\bot$ as a synonym for $A \sqcap \neg A$, and $\top$ as a synonym for $A \sqcup \neg A$. The last trigger detects the contradiction in terms of the certainty values specified for the same assertion. To be more specific, in case there is no intersection in the certainty values specified for the same assertion, we have conflicting assertions, hence a contradiction is detected.

Negation Rule:
if 1. $\langle a : \neg A, \Gamma \rangle \langle -, - \rangle \in \mathcal{A}_i^{\mathcal{E}}$, and
2. $\langle a : A, \sim\Gamma \rangle \langle -, - \rangle \notin \mathcal{A}_i^{\mathcal{E}}$
then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{ \langle a : A, \sim\Gamma \rangle \langle -, - \rangle \}$

The intuition behind the negation rule is that, if we know an assertion has certainty value Γ, then the certainty of its negation can be obtained by applying the negation operator in the lattice to Γ.

Disjunction Rule:
if $\langle a : C \sqcup D, \Gamma \rangle \langle f_c, f_d \rangle \in \mathcal{A}_i^{\mathcal{E}}$
then for each $\Psi \in \{C, D\}$
if 1. Ψ is atomic, and
2. $\langle a : \Psi, x_{a:\Psi} \rangle \ \langle -, - \rangle \notin \mathcal{A}_i^{\mathcal{E}}$
then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{ \langle a : \Psi, x_{a:\Psi} \rangle \langle -, - \rangle \}$
else if 1. Ψ is not atomic, and
2. $\langle a : \Psi, x_{a:\Psi} \rangle \ \langle f_c, f_d \rangle \notin \mathcal{A}_i^{\mathcal{E}}$
then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{ \langle a : \Psi, x_{a:\Psi} \rangle \langle f_c, f_d \rangle \}$
if $(f_d(x_{a:C}, x_{a:D}) = \Gamma) \notin \mathcal{C}_j$,
then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{ (f_d(x_{a:C}, x_{a:D}) = \Gamma) \}$
if $(f_d(x_{a:C}, x_{a:D}) \ \succeq \ x_{a:\Psi}) \notin \mathcal{C}_j$,
then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{ \ f_d(x_{a:C}, x_{a:D}) \ \succeq \ x_{a:\Psi} \}$

The intuition behind this rule is that, if we know an individual is in $C \sqcup D$, then we know it is in either C, D, or in both. In addition, according the semantics of the description language, we know that the semantics of $a : C \sqcup D$ is defined by applying the disjunction function to the interpretation of $a : C$ and the interpretation of $a : D$. Finally, the last part of the rule re-enforces the "bounded below" property of the disjunction function.

Role Exists Restriction Rule:
if $\langle a : \exists R.C, \Gamma \rangle \langle f_c, f_d \rangle \in \mathcal{A}_i^{\mathcal{E}}$
then if there exists no individual b such that
$(f_c(x_{(a,b):R}, x_{b:C}) = x_{a:\exists R.C}) \in \mathcal{C}_j$
then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{ \langle (a, b) : R, x_{(a,b):R} \rangle \langle -, - \rangle \}$
if C is atomic
then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{ \langle b : C, x_{b:C} \rangle \langle -, - \rangle \}$
else $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{ \langle b : C, x_{b:C} \rangle \langle f_c, f_d \rangle \}$
where b is a new individual
$\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{ (f_c(x_{(a,b):R}, x_{b:C}) = x_{a:\exists R.C}) \}$
if Γ is not the variable $x_{a:\exists R.C}$
then if $(x_{a:\exists R.C} = \Gamma') \in \mathcal{C}_j$
then if 1. $\Gamma \neq \Gamma'$, and
2. Γ is not an element in Γ'
then $(x_{a:\exists R.C} = \Gamma') \leftarrow (x_{a:\exists R.C} = \oplus(\Gamma, \Gamma'))$
where $\oplus$ is the join operator of the lattice and $\leftarrow$ means whatever is on the LHS is replaced by the RHS
else $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{ (x_{a:\exists R.C} = \Gamma) \}$

The intuition behind this rule is that we view $\exists R.C$ as the open first order formula $\exists b.\ R(a, b) \wedge C(b)$, where $\exists$ is viewed as a disjunction over the elements of the domain. To be more specific, the semantics of $R(a, b) \wedge C(b)$ is captured using the conjunction function as $f_c(R^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b))$, and $\exists b$ is captured using the join operator in the lattice $\oplus_{b \in \Delta^{\mathcal{I}}}$.

Example

The fuzzy $\mathcal{ALC}$ proposed in (Tresp, C. & Molitor, R. 1998; Sánchez, D. & Tettamanzi, A. G. B. 2004; Straccia, U. 1998; 2001) can be represented in the generic framework by setting the certainty lattice as $\mathcal{L} = \langle \mathcal{V}, \leq \rangle$, where $\mathcal{V} = \mathcal{C}[0, 1]$ is the set of closed subintervals $[\alpha, \beta]$ in [0, 1] such that $\alpha \preceq \beta$. The negation operator is defined as $\sim([\alpha, \beta]) = [1 - \beta, 1 - \alpha]$, the conjunction function is min, and the disjunction function is max. In (Tresp, C. & Molitor, R. 1998; Sánchez, D. & Tettamanzi, A. G. B. 2004; Straccia, U. 2001), the meet operator is inf (infimum), and the join operator is sup (supremum). On the other hand, (Straccia, U. 1998) uses min as the meet operator, and max as the join operator. As an example, suppose we have the following fuzzy knowledge base:

$\mathcal{T}$=$\{\langle \exists owns.Porsche \sqsubseteq Rich \sqcup CarFanatic, [0.8,1]\rangle$
$\langle min, max\rangle$,
$\langle Rich \sqsubseteq Golfer, [0.7,1]\rangle\langle -, max\rangle\}$
$\mathcal{A}$=$\{\langle Tom : \exists owns.Porsche, [0.9,1]\rangle\langle min, -\rangle$,
$\langle Tom : \neg CarFanatic, [0.6,1]\rangle\langle -, -\rangle\}$

Then, we could first remove all the axioms in $\mathcal{T}$, add the corresponding assertions to the ABox $\mathcal{A}$, and initialize the extended ABox to be:

$\mathcal{A}_0^{\mathcal{E}}$ = $\{\langle Tom : \exists owns.Porsche, [0.9,1]\rangle\langle min, -\rangle$,
$\langle Tom : \neg CarFanatic, [0.6,1]\rangle\langle -, -\rangle$,
$\langle Tom : (\forall owns.\neg Porsche) \sqcup (Rich \sqcup CarFanatic), [0.8,1]\rangle\langle min, max\rangle$,
$\langle Tom : \neg Rich \sqcup Golfer, [0.7,1]\rangle\langle -, max\rangle\}$

and $\mathcal{C}_0 = \{\}$. Now, we are ready to apply the completion rules to construct the model. Due to the limited space, we show only how to apply Role Exists Restriction Rule to the first assertion. According to this assertion, Tom must own at least one $Porsche$, with certainty more than 0.9. Indeed, when we apply the Role Exists Restriction Rule to this assertion, we get:

$\mathcal{A}_1^{\mathcal{E}} = \mathcal{A}_0^{\mathcal{E}} \cup \{\langle (Tom, p_1) : owns, x_{(Tom,p_1):owns}\rangle\langle -, -\rangle$,
$\langle p_1 : Porsche, x_{p_1:Porsche}\rangle\langle -, -\rangle\}$
where p_1 is a new individual

$\mathcal{C}_1 = \mathcal{C}_0 \cup \{(min(x_{(Tom,p_1):owns}, x_{p_1:Porsche}) = x_{Tom:\exists owns.Porsche})\}$

$\mathcal{C}_2 = \mathcal{C}_1 \cup \{(x_{Tom:\exists owns.Porsche} = [0.9,1])\}$

Note that in case we have the same assertion in the knowledge base but with a different certainty value, say,

$\langle Tom : \exists owns.Porsche, [1,1]\rangle\langle min, -\rangle$

Then, we are not going to add yet more constraints to $\mathcal{C}_j$. Instead, we replace the constraint in $\mathcal{C}_2$, $x_{Tom:\exists owns.Porsche} = [0.9,1]$ with $x_{Tom:\exists owns.Porsche} = sup([1,1],[0.9,1])$, where sup is the join operator of the lattice $\mathcal{L}$.

After applying Role Exists Restriction Rule to the first assertion, we can continue applying other completion rules to the rest of assertions in the extended ABox until either we get a clash or no further rule could be applied. If a clash is obtained, the knowledge base is inconsistent. Otherwise, an optimization method is applied to check if the system of inequations is solvable, or to find the tightest bound for which an assertion is true.

Now, suppose we want to reason about the same knowledge base using basic probability instead of fuzzy logic. Then, we may replace the conjunction function in the knowledge base with the algebraic product ($\times(\alpha,\beta) = \alpha\beta$), and the disjunction function with the independent function ($ind(\alpha,\beta) = \alpha + \beta - \alpha\beta$) if desired. For example, the terminological axiom: $\langle \exists owns.Porsche \sqsubseteq Rich \sqcup CarFanatic, [0.8,1]\rangle\langle \times, ind\rangle$ asserts that the probability that someone owns Porsche is Rich or CarFanatic is at least 0.8. Once the knowledge base is defined and the pre-processing steps are followed, the appropriate completion rules can be applied to perform the desired inference. Note that, since reasoning with probability requires extra information/knowledge about the events and facts in the world (Σ), we are investigating ways to model knowledge base with more general probability theory, such as positive/negative correlation, ignorance, and conditional independence.

It is important to note that, unlike other proposals which support only one form of uncertainty for the entire knowledge base, our framework allows the user to specify different combination functions (f_c, f_d) for each of the axioms and assertions in the knowledge base. For example, for a given knowledge base, an axiom may use $\langle min, max\rangle$ as the combination functions, while another axiom may use $\langle \times, ind\rangle$. This is in addition to the fact that our generic framework can simulate the computation of many DLs with uncertainty, each having a different underlying certainty formalism.

Conclusions and Future Work

We introduced a generic framework which allows us to represent several existing extensions of DLs with uncertainty in a uniformed manner. In particular, we abstracted away the underlying notion of uncertainty (fuzzy logic, probability, possibilistic logic), the way in which the constructors of the description language are interpreted (by flexibly defining the conjunction and disjunction functions), and the way in which the inference procedure proceeds. An implementation of the proposed generic framework is underway. As future works, we plan to study the complexity of the proposed reasoning procedure, extend the generic framework to a more expressive portion of DL (e.g., $\mathcal{SHOIN}$), and also study optimization techniques for extended framework.

References

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American* 284(5).

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

Haarslev, V.; Pai, H.-I.; and Shiri, N. 2005. A generic framework for description logics with uncertainty. In *Proceedings of URSW*, 77–86.

Lakshmanan, L.V.S., and Shiri, N. 2001. A parametric approach to deductive databases with uncertainty. *IEEE TKDE* 13(4):554–570.

Sánchez, D., and Tettamanzi, A. G. B. 2004. Generalizing quantification in fuzzy description logics. In *Proceedings of Fuzzy Days-04*. Springer-Verlag.

Straccia, U. 1998. A fuzzy description logic. In *Proceedings of AAAI-98*, 594–599. Menlo Park, CA, USA: AAAI Press.

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

Tresp, C., and Molitor, R. 1998. A description logic for vague knowledge. In *Proceedings of ECAI-98*, 361–365. Brighton, UK: John Wiley and Sons.

Zadeh, L. A. 1978. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* 1(1):3–28.