Completion Rules for Uncertainty Reasoning with the Description Logic $\mathcal{ALC}$

Volker Haarslev, Hsueh-Ieng Pai, Nematollaah Shiri

Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, Canada

Abstract. Description Logics (DLs) are gaining more popularity as the foundation of ontology languages for the Semantic Web. On the other hand, uncertainty is a form of deficiency or imperfection commonly found in the real-world information/data. In recent years, there has been an increasing interest in extending the expressive power of DLs to support uncertainty, for which a number of frameworks have been proposed. In this paper, we introduce an extension of DL ($\mathcal{ALC}$) that unifies and/or generalizes a number of existing approaches for DLs with uncertainty. We first provide a classification of the components of existing frameworks for DLs with uncertainty in a generic way. Using this as a basis, we then discuss ways to extend these components with uncertainty, which includes the description language, the knowledge base, and the reasoning services. Detailed explanations and examples are included to describe the proposed completion rules.

1 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 [10]. Modeling uncertainty and reasoning with it have been challenging issues for over two decades in database and artificial intelligence research [2,10,12,13]. In recent years, uncertainty management has attracted the attention of researchers in Description Logics

(DLs) [1]. 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 [3], 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" [5]. Informally, an 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 mostly 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 the 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. It is not our intention to discuss which extension is better. In fact, different applications may require different aspects to be modeled, or in some cases, it may even be desired to model different aspects within the same application [14].

Following the approach of the parametric framework [11], we propose in this paper 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 and semantics of knowledge bases, reasoning techniques, design and implementation of reasoners, and optimization techniques in a framework-independent manner.

The rest of this paper is organized as follows. Sect. 2 provides an overview of the standard DL framework and presents a classification of existing frameworks of uncertainty in DL. In Sect. 3, we present our generic framework for DL with uncertainty in detail along with examples. We discuss how to represent uncertainty knowledge in a general way, as well as how to perform reasoning services. Finally, concluding remarks and future directions are presented in Sect 4.

2 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.

2.1 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 $\mathcal{ALC}$ [1].
2. *Knowledge Base*: The knowledge base is composed of both intensional knowledge and extensional knowledge. The intensional knowledge includes the Terminological Box (TBox) consisting of a set of terminological axioms, and the Role Box (RBox) consisting of a set of role axioms. On the other hand, the extensional knowledge includes the Assertional Box (ABox) consisting of a set of assertions/facts.
3. *Reasoning Component*: A DL framework is equipped with reasoning services that enables one to derive implicit knowledge.

2.2 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 [19], 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 [20], 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.

3 Our DL framework with uncertainty

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*: The syntax and semantics of the description language are extended to express uncertainty.
2. *Knowledge Bases with Uncertainty*: A knowledge base is composed of the intensional knowledge (TBox and RBox) and extensional knowledge, both extended with uncertainty.
3. *Reasoning with Uncertainty*: The DL framework is equipped with reasoning services that take into account the presence of uncertainties in DL theories during the reasoning process.

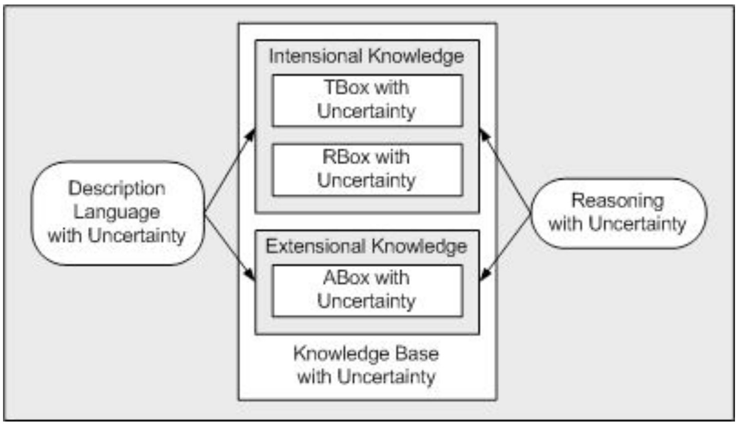

Fig. 1. DL Framework with Uncertainty

In what follows, we discuss each of these three components in detail, along with illustrating examples. Note that this paper extends our previous work [6] by presenting uncertainty inference rules for the reasoning component of the framework.

3.1 Description Language with Uncertainty

To provide a generic extension to a description language, one needs to develop a way to represent certainty values, and assign semantics to each element in the description language.

Representation of Certainty Values

To represent the certainty values, we take a *lattice-based approach* followed in the parametric framework [11]. 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 $\prec$, $\succeq$, $\succ$, and $=$ with their obvious meanings. We 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 the 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 Sect. 3.2.

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 [11], 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 the 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:

1. Boundary Conditions: $\sim b = t$ and $\sim t = b$.
2. 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 the 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 × 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:

1. De Morgan's Rule: $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$ and $\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$.
2. Negating Quantifiers Rule: $\neg\exists R.C \equiv \forall R.\neg C$ 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.

3.2 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.

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 $\mathcal{I}$.

To provide a generic extension to the knowledge base, there is a need to give a syntactical and semantical extension to both the intensional (TBox and RBox) and extensional knowledge (ABox).

TBox with Uncertainty

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 $\neg C \sqcup D$. An interpretation $\mathcal{I}$ satisfies $\langle C \sqsubseteq D, \alpha\rangle\langle f_c, f_d\rangle$ iff for all individuals $a \in \Delta^{\mathcal{I}}$, $(f_d\ (\sim C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))) \in \alpha$. By defining the semantics for concept subsumption this way, it also allows us to guarantee that some basic properties hold, such as the Negating Quantifiers Rule described in the previous subsection.

RBox with Uncertainty

The RBox $\mathcal{R}$ 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.

ABox with Uncertainty

An ABox $\mathcal{A}$ consists of a set of assertions of the form $\langle a\text{:C}, \alpha\rangle\langle f_c, f_d\rangle$ or $\langle (a, b)\text{:}R, \alpha\rangle\langle -, -\rangle$, 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 $\langle a\text{:C}, \alpha\rangle\langle f_c, f_d\rangle$ (resp. $\langle (a, b)\text{:}R, \alpha\rangle\langle -, -\rangle$) iff $C^{\mathcal{I}}(a) \in \alpha$ (resp. $R^{\mathcal{I}}(a, b) \in \alpha$).

3.3 Reasoning with Uncertainty

In this section, we describe the reasoning procedure for the generic framework proposed here. Let $\Sigma = \langle \mathcal{T}, \mathcal{A}\rangle$ 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 Σ 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 $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, we are interested in finding the tightest bound for which X is true. As an example, if the certainty values are expressed in a range $[l, u]$, then we would like to find the largest l and the smallest u such that the knowledge base entails X. 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 pre-processing 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 satisfiability 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 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. In what follows, we describe each of these rules in detail.

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 contradictions 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 clash 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. For example, suppose the certainty domain is defined as $\mathcal{V} = \mathcal{C}[0,1]$, meaning the set of closed subintervals $[\alpha, \beta]$ in [0, 1] such that $\alpha \preceq \beta$. If a knowledge base contains both assertions $\langle John{:}Tall, [0.8, 0.9]\rangle$ and $\langle John{:}Tall, [0.2, 0.4]\rangle$, then the last clash trigger will detect such conflicting information in the knowledge base.

Concept Assertion Rule:
if 1. $\langle a : A,\ \Gamma\rangle\langle -, -\rangle \in \mathcal{A}_i^{\mathcal{E}}$, and
2. $(x_{a:A} = \Gamma) \notin \mathcal{C}_j$, and
3. Γ is not the variable $x_{a:A}$
then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{(x_{a:A} = \Gamma)\}$

This rule simply adds the certainty value of each atomic concept assertion to the constraint set $\mathcal{C}_j$. For example, if we have the assertion

$\langle John{:}Tall$, [0.6, 0.9]$\rangle\langle -, -\rangle$ in the ABox, then we add the constraint ($x_{John:Tall}$ = [0.6, 0.9]) to the constraint set $\mathcal{C}_j$.

Role Assertion Rule:

if 1. $\langle (a, b){:}R, \Gamma\rangle\langle -, -\rangle \in \mathcal{A}_i^{\mathcal{E}}$, and

2. $(x_{(a,\, b):R} = \Gamma) \notin \mathcal{C}_j$, and

3. Γ is not the variable $x_{(a,\, b):R}$

then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{(x_{(a,\, b):R} = \Gamma)\}$

Similar to the Concept Assertion Rule, this rule simply adds the certainty value of each atomic role assertion to the constraint set $\mathcal{C}_j$. For example, if we have the assertion $\langle (John, Diabetes){:}hasDisease$, [0.8, 0.9]$\rangle\langle -, -\rangle$ in the ABox, then we add the constraint ($x_{(John,\, Diabetes):\text{hasDisease}}$ = [0.8, 0.9]) to the constraint set $\mathcal{C}_j$.

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. For example, if the certainty domain is $\mathcal{V} = \mathcal{C}[0,1]$, and the negation operator is defined as $\sim([\alpha, \beta] = [1 - \beta, 1 - \alpha]$. Then, if we have the assertion $\langle John : \neg Tall$, [0.4, 0.8]$\rangle\langle -, -\rangle$ in the ABox, we could also infer that $\langle John : Tall$, [0.2, 0.6]$\rangle\langle -, -\rangle$.

Conjunction Rule:

if $\langle a : C \sqcap D, \Gamma\rangle\langle f_c, f_d\rangle \in \mathcal{A}_i^{\mathcal{E}}$

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 $(f_c\,(x_{a:C}, x_{a:D}) = \Gamma) \notin \mathcal{C}_j$,

then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{(f_c\,(x_{a:C}, x_{a:D}) = \Gamma)\}$

if $(f_c\,(x_{a:C}, x_{a:D}) \preceq x_{a:\Psi}) \notin \mathcal{C}_j$, then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{(f_c\,(x_{a:C}, x_{a:D}) \preceq x_{a:\Psi})\}$

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

For example, if we have the assertion $\langle John{:}Tall \sqcap Thin, [0.6, 0.8]\rangle\langle min, ma\rangle$ in the ABox, then we could infer that $\langle John{:}Tall, x_{John:Tall}\rangle\langle -, -\rangle$ and $\langle John{:}Thin, x_{John:Thin}\rangle\langle -, -\rangle$, with the constraint $(min\,(x_{John:Tall}, x_{John:Thin}) = [0.6, 0.8])$ satisfied. In addition, based on the property of the conjunction function, we also know that $min\,(x_{John:Tall}, x_{John:Thin}) \preceq$ both $x_{John:Tall}$ and $x_{John:Thin}$.

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 $C \sqcup D$, 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 $C \sqcup D$ 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.

For example, if we have $\langle John{:}Rich \sqcup CarFanatic, [0.6, 0.8]\rangle\langle min, max\rangle$ in the ABox, then we could infer $\langle John{:}Rich, x_{John:Rich}\rangle\langle -, -\rangle$ and $\langle John{:}CarFanatic, x_{John:CarFanatic}\rangle\langle -, -\rangle$, with the constraint $(max\,(x_{John:Rich}, x_{John:CarFanatic}) = [0.6, 0.8])$ satisfied. In addition, based on the property of the disjunction function, we also know that $max\,(x_{John:Rich}, x_{John:CarFanatic}) \succeq$ both $x_{John:Rich}$ and $x_{John:CarFanatic}$.

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. That is, 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 $\oplus_{b\in\Delta^{\mathcal{I}}}$.

For example, if the join operator is *sup* (supremum), and we have the assertion $\langle John{:}\exists hasDisease.Diabetes, [0.4, 0.6]\rangle\langle min, max\rangle$ in the ABox. Then, we could infer that $\langle(John, d1){:}hasDisease, x_{(John,\,d1):hasDisease}\rangle\langle -, -\rangle$ and $\langle d1{:}Diabetes, x_{d1:Diabetes}\rangle\langle -, -\rangle$, where *d1* is a new individual. In addition, the constraints $(min\ (x_{(John,\ d1):hasDisease}, x_{d1:Diabetes}) = x_{John:\exists hasDisease.Diabetes})$ and $(x_{John:\exists hasDisease.Diabetes} = [0.4, 0.6])$ must be satisfied. Now, suppose we have yet another assertion $\langle John{:}\exists hasDisease.Diabetes, [0.5, 0.9]\rangle\langle min, max\rangle$ in the ABox. Then, when we apply Role Exists Restriction Rule, we will not generate a new individual. Instead, we simply replace the constraint $(x_{John:\exists hasDisease.Diabetes} = [0.4, 0.6])$ in $\mathcal{C}_j$ with the constraint $(x_{John:\exists hasDisease.Diabetes} = sup\ ([0.5, 0.9], [0.4, 0.6]))$, where *sup* is the join operator in the lattice. This new constraint takes into account the certainty value of the current assertion as well as that of the previous assertion.

Role Value Restriction Rule:

if $\{\langle a : \forall R.C, \Gamma\rangle\langle f_c, f_d\rangle, \langle (a, b){:}R, \Gamma'\rangle\langle -, -\rangle\} \subseteq \mathcal{A}_i^{\mathcal{E}}$
then if 1. C is atomic, and
  2. $\langle b : C, x_{b:C}\rangle\langle -, -\rangle \notin \mathcal{A}_i^{\mathcal{E}}$
  then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{\langle b : C, x_{b:C}\rangle\langle -, -\rangle\}$
  else if 1. C is not atomic, and
    2. $\langle b : C, x_{b:C}\rangle\langle f_c, f_d\rangle \notin \mathcal{A}_i^{\mathcal{E}}$
    then $\mathcal{A}_{i+1}^{\mathcal{E}} = \mathcal{A}_i^{\mathcal{E}} \cup \{\langle b : C, x_{b:C}\rangle\langle f_c, f_d\rangle\}$
  if $(f_d\ (\sim x_{(a,b):R}, x_{b:C}) = x_{a:\forall R.C}) \notin \mathcal{C}_j$
  then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{(f_d\ (\sim x_{(a,b):R}, x_{b:C}) = x_{a:\forall R.C})\}$
  if Γ is not the variable $x_{a:\forall R.C}$
  then if $(x_{a:\forall R.C} = \Gamma'') \in \mathcal{C}_j$
    then if 1. $\Gamma \neq \Gamma''$, and
      2. Γ is not an element in Γ''
      then $(x_{a:\forall R.C} = \Gamma'') \leftarrow (x_{a:\forall R.C} = \otimes(\Gamma, \Gamma''))$
        where $\otimes$ is the meet operator of the lattice and
          $\leftarrow$ means whatever is on the LHS is
          replaced by the RHS
      else $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \{(x_{a:\forall R.C} = \Gamma)\}$

The intuition behind this rule 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 $R(a, b) \rightarrow C(b)$. That is, 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 $f_d\ (\sim R^{\mathcal{I}}(a, b)$, and $\forall b$ is captured using the meet operator in the lattice $\otimes_{b \in \Delta^{\mathcal{I}}}$.

For example, if the meet operator is *inf* (infimum), and we have assertions $\langle John{:}\forall hasPet.Dog$, [0.4, 0.6]$\rangle\langle min, max\rangle$ and $\langle (John, d1){:}hasPet$, [0.5, 0.8]$\rangle\langle -, -\rangle$ in the ABox. Then, we could infer that $\langle d1{:}Dog, x_{d1:Dog}\rangle\langle -, -\rangle$. In addition, the constraints $(max\ (\sim x_{(John,d1):hasPet}, x_{d1:Dog}) = x_{John:\forall hasPet.Dog})$ and $(x_{John:\forall hasPet.Dog} = [0.4, 0.6])$ must be satisfied. Now, suppose we have yet another assertion $\langle John{:}\forall hasPet.Dog$, [0.5, 0.9]$\rangle\langle min, max\rangle$ in the ABox. Then, when we apply Role Value Restriction Rule, we simply replace the constraint $(x_{John:\forall hasPet.Dog} = [0.4, 0.6])$ in $\mathcal{C}_j$ with the new constraint $(x_{John:\forall hasPet.Dog} = inf\ ([0.5, 0.9], [0.4, 0.6]))$, where *inf* is the meet operator in the lattice. Note that the new constraint takes into account the certainty value of the current assertion as well as that of the previous assertion.

3.4 Illustrative Example

Most of the proposed fuzzy DLs ("most" because our framework supports only $\mathcal{ALC}$) can be represented in the generic framework by setting the certainty lattice as $\mathcal{L} = \langle \mathcal{V}, \preceq \rangle$, 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 in this case is defined as $\sim([\alpha, \beta]) = [1 - \beta, 1 - \alpha]\}$. In [7,15,17,18], the meet operator is *inf* (infimum) and the join operator is *sup* (supremum). On the other hand, in [16], *min* is used as the meet operator, and *max* is used as the join operator. The conjunction function used in all these proposals is *min*, whereas the disjunction function used is *max*. As an example, suppose we have the following fuzzy knowledge base:

$\mathcal{T} = \{\langle \exists owns.Porsche \sqsubseteq (Rich \sqcup CarFanatic), [0.8, 1]\rangle\langle min, max\rangle,$
$\quad \langle Rich \sqsubseteq Golfer, [0.7, 1]\rangle\langle -, max\rangle\}$

$\mathcal{A} = \{\langle Tom : \exists owns.Porsche, [0.9, 1]\rangle\langle min, -\rangle,$
$\quad \langle Tom : \neg CarFanatic, [0.6, 1]\rangle\langle -, -\rangle\}$

Then, we first transform all the axioms into normal form:

$\mathcal{T} = \{\langle \top \sqsubseteq ((\forall owns.\neg Porsche) \sqcup (Rich \sqcup CarFanatic)), [0.8,1]\rangle\langle min, max\rangle,$
$\quad \langle \top \sqsubseteq (\neg Rich \sqcup Golfer), [0.7, 1]\rangle\langle -, max\rangle\}$

After that, we could remove the axioms in the TBox $\mathcal{T}$ by adding the corresponding assertions to the ABox $\mathcal{A}$. To be more specific, for each individual a in the ABox (in this case, we have only one individual, *Tom*, in the ABox) and for each axiom of the form $\langle \top \sqsubseteq \neg C \sqcup D, \alpha\rangle\langle f_c, f_d\rangle$ in the TBox, we add an assertion $\langle a{:}\neg C \sqcup D, \alpha\rangle\langle f_c, f_d\rangle$ to the ABox. Hence, in this step, we add the following two assertions to the ABox:

$\{\langle Tom : ((\forall owns.\neg Porsche) \sqcup (Rich \sqcup CarFanatic)), [0.8,1]\rangle\langle min, max\rangle,$
$\langle Tom : (\neg Rich \sqcup Golfer), [0.7, 1]\rangle\langle -, max\rangle\}$

Now, we can initialize the extended ABox to be:

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

and the constraint set to be $\mathcal{C}_0 = \{\}$.

Note that, according to the clash triggers, there is no trivial contradiction in the knowledge base. So, once the pre-processing steps are over, we are ready to apply the completion rules to construct the model. For sake of brevity, we show only how to apply the Role Exists Restriction Rule to the first assertion.

According to the first assertion, $\langle Tom : \exists owns.Porsche, [0.9, 1]\rangle\langle min, -\rangle$, *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:

$\mathcal{A}_1^{\mathcal{E}} = \mathcal{A}_0^{\mathcal{E}} \cup \{\langle (Tom, p_1) : owns, x_{(Tom,\, p1):owns}\rangle\langle -, -\rangle,$
$\quad\quad \langle p_1 : Porsche, x_{p1:Porsche}\rangle\langle -, -\rangle\}$

where p_1 is a new individual

$\mathcal{C}_1 = \mathcal{C}_0 \cup \{(min\ (x_{(Tom,\, p1):owns}, x_{p1:Porsche}) = x_{Tom:\exists owns.Porsche})\}$

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

After applying the 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, a linear programming technique 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 first terminological axiom in the above knowledge base can be interpreted using simple probability as: $\langle \exists owns.Porsche \sqsubseteq (Rich \sqcup CarFanatic), [0.8, 1]\rangle\langle \times, ind\rangle$, which asserts that the probability that someone owns a 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 bases with more general probability theory, such as positive/negative correlation [9], ignorance [9], and conditional probability [4,8].

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 different underlying certainty formalism.

4 Conclusion and Future Works

We introduced a generic framework which allows us to incorporate various forms of uncertainty within DLs in a uniform way. In particular, we abstracted away the underlying notion of uncertainty (which could be fuzzy, probability, possibilistic, etc.), the way in which the constructors in 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. In addition, on the basis of the finite model property and disallowing terminological cycles, we can guarantee termination of the proposed reasoning procedure. We are working to establish this and the completeness of this procedure. As future work, we plan to further extend the generic framework to a more expressive fragment of DL (e.g., $\mathcal{SHOIN}$), and study optimization techniques for the extended framework.

Acknowledgements

This work is supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada, Genome Québec, and by ENCS, Concordia University, Montreal, Québec, Canada. We also thank the anonymous reviewers for their helpful comments.

References

1. Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF, eds (2003) The description logic handbook: theory, implementation, and applications, Cambridge University Press.
2. Bacchus F (1990) Representing and reasoning with probabilistic knowledge - a logical approach to probabilities, MIT Press.
3. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Scientific American 284(5).
4. Giugno R, Lukasiewicz T (2002) P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In: Proceedings of the European conference on logics in artificial intelligence, Cosenza, Italy, pp 86–97.
5. Gruber TR (1993) A translation approach to portable ontology specifications. Knowledge acquisition 5(2):199–220.

6. Haarslev V, Pai HI, Shiri N (2005) A generic framework for description logics with uncertainty. In: Proceedings of uncertainty reasoning for the semantic web, Galway, Ireland, pp 77–86.
7. Hölldobler S, Khang TD, Störr HP (2002) A fuzzy description logic with hedges as concept modifiers. In: Proceedings of the 3rd international conference on intelligent technologies, Science and Technics Publishing House, Hanoi, Vietnam, pp 25–34.
8. Koller D, Levy AY, Pfeffer A (1997) P-CLASSIC: A tractable probablistic description logic. In: Proceedings of the 14th national conference on artificial intelligence, AAAI Press, Providence, Rhode Island, pp 390–397.
9. Lakshmanan LVS, Sadri F (1994) Probabilistic deductive databases. In: Proceedings of workshop on design and implementation of parallel logic programming systems, MIT Press, Ithaca, NY, pp 254–268.
10. Lakshmanan LVS, Shiri N (2001a) Logic programming and deductive databases with uncertainty: A survey. Encyclopedia of computer science and technology, vol 45, Marcel Dekker, New York, pp 153–176.
11. Lakshmanan LVS, Shiri N (2001b) A parametric approach to deductive databases with uncertainty. IEEE transactions on knowledge and data engineering, 13(4):554–570.
12. Motro A, Smets P, eds. (1997) Uncertainty management in information systems - from needs to solutions, Springer-Verlag.
13. Parsons S (1996) Current approaches to handling imperfect information in data and knowledge bases. IEEE transactions on knowledge and data engineering, 8(3):353–372.
14. Ross TJ, Booker JM, Parkinson WJ, eds (2002) Fuzzy logic and probability applications: bridging the gap, SIAM.
15. Sánchez D, Tettamanzi, AGB (2004) Generalizing quantification in fuzzy description logics. In: Proceedings of the 8th Fuzzy Days, Springer-Verlag, Dortmund, Germany.
16. Straccia U (1998) A fuzzy description logic. In: Proceedings of the 15th national conference on artificial intelligence, AAAI Press, Menlo Park, CA, USA, pp 594–599.
17. Straccia U (2001) Reasoning within fuzzy description logics. Journal of artificial intelligence research 14:137–166.
18. Tresp C, Molitor R (1998) A description logic for vague knowledge. In: Proceedings of the 13th European conference on artificial intelligence, John Wiley and Sons, Brighton, UK, pp 361–365.
19. Zadeh LA (1965) Fuzzy sets. Information and control, 8:338–353.

20. Zadeh LA (1978) Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1(1):3–28.