Efficient Query Answering over Fuzzy EL-OWL Based on Crisp Datalog Rewritable Fuzzy $\mathcal{EL}^{++}$

Afef Bahri, Rafik Bouaziz and Faiez Gargouri

MIRACL Laboratory, University of Sfax, Tunisia
E-mail: afef.bahri@gmail.com, raf.bouaziz@fsegs.rnu.tn, faiez.gargouri@fsegs.rnu.tn

Abstract. OWL EL is an extension of the tractable $\mathcal{EL}^{++}$ description logic. Despite their inference capabilities over TBoxes, DL reasoners have a high ABox reasoning complexity which may constitute a serious limitation in the Semantic Web where we rely mainly on query answering (i.e. instance checking). The subsomption algorithm used in fuzzy $\mathcal{EL}^{++}$ reduce instance checking into concept satisfiability. To allow efficient instance checking and query answering over fuzzy EL-OWL, we propose in this paper two approaches for integrating fuzzy $\mathcal{EL}^{++}$ and crisp Datalog programs: homogenous and hybrid approaches. In the homogenous approach, we define crisp $\mathcal{EL}^{++}$ rules based on a crisp rewriting of fuzzy $\mathcal{EL}^{++}$. To preserve decidability, crisp $\mathcal{EL}^{++}$ rules are then written with Datalog safe ones. In the hybrid approach, fuzzy $\mathcal{EL}^{++}$ axioms and assertions are defined with EDB facts and Datalog rules are used to denote fuzzy $\mathcal{EL}^{++}$ instance checking deduction rules. That is, DL axioms and assertions are translated into Datalog EDB facts and Datalog rules are used to derive conclusions about them.

Keywords: Fuzzy $\mathcal{EL}^{++}$, Instance checking, Homogenous and hybrid approaches, Rule based fuzzy OWL reasoning, Datalog entailment rules

1. Introduction

The OWL is a knowledge representation standard proposed by the W3C Consortium. Its semantics is based on description logics and particularly on the expressive SROIQ(DL). This expressivity lead to high reasoning complexity and make reasoning algorithm impractical in real life applications. To solve this issue, three lightweight sublanguages of OWL have been introduced. We talk about OWL RL, OWL QL and OWL EL profiles. OWL RL is the rule based fragment of OWL, OWL QL is a query language and OWL EL is used for conceptual modelling. OWL EL is an extended version of the $\mathcal{EL}^{++}$ description logic. OWL EL supports, among others, local reflexivity and concept products. The integration of description logics with rules and logic programming has attracted the interest of many researches [10,12,11,8,13,9]. The rules languages which are subject of theses integration are the ones based on Horn clausal logics. The integration of the two paradigms should play an important role in the Semantic Web. In fact, despite their inference capabilities over complex TBoxes, DL reasoners have a high ABox reasoning complexity which may constitute a serious limitation in the Semantic Web where we rely mainly on query answering (i.e. instance checking). The subsomption algorithm used in fuzzy $\mathcal{EL}^{++}$ reduce instance checking into concept satisfiability. That is, to retrieve the instances of a given concept, we need to run the subsomption algorithm for each individual in the ABox. In Datalog systems, query answers are computed in one pass (i.e. bottom-up, top-down). Two principal integration approaches are used in the literature: the hybrid and the homogenous approaches. In the hybrid approach, the two paradigms are used
to represent a knowledge bases. That is, a knowledge base $\mathcal{KB}$ is defined as $\mathcal{KB} = \langle \mathcal{D} \mathcal{D}, \mathcal{P} \mathcal{L} \rangle$ where $\mathcal{CD}$ is defined with description logic and $\mathcal{PL}$ is defined with Logic Programming. In hybrid approach, the reasoning over DL ontologies is performed only by the DL reasoner. The rules are used to define constraints on the defined ontology and rule engine are just used for rule execution.

While, in hybrid approaches, the rules and ontologies are treated separately, in homogenous ones rules and ontology are combined in a new single logic language. In practice, the description logic is mapped into a rule based formalisms known as description logic programs. Such an integration approach may lead to undecidability of reasoning problems due the opposite assumption (Closed world and Open world assumption) of the two paradigms. Decidability may be obtained by restricting rules to DL-safe ones [12]. Homogenous approaches perform reasoning only by rules engine.

The problem of combination of rules and ontologies has equally been treated for fuzzy ontologies. The majority of the works realized on this subject propose homogenous integration approaches in which fuzzy DL programs are defined as a result of the integration of fuzzy DLs and fuzzy rule languages [15]. The reasoning should be performed by fuzzy inference engines. The fact that there is no common implementation of fuzzy rule engines, thses works have only focused on the theoretical aspects of the integration.

In this paper we focus on the problem of combination of the fuzzy $\mathcal{EL}^{++}$ with crisp rule language for tractable query answering over large scale ABoxes. We propose in this paper two integration approach of the two paradigms: in the first one we adopt a tight integration approach in the sense that a unified language and semantics are used to represent the two paradigms (homogenous like). The crisp Datalog rule language acts as a unified language and Datalog based system correspond to the unique framework in which various reasoning tasks are realized. In the second, a loose coupled integration approach (hybrid like), the two paradigms act separately but in a complementary way. That is, fuzzy DL reasoners are used to reason over the TBox of the ontology and crisp Datalog inference engines reason over the ABoxes for instance checking and query answering.

As we want to work with crisp Datalog program, we need to represent fuzzy ontology axioms and assertions with crisp Datalog predicates. Two approaches are equally used in this paper. In the first approach, we begin by transforming fuzzy concepts and roles are mapped into crisp ones based on [4]. We define after that, crisp $\mathcal{EL}^{++}$ rules based on DL axioms and assertions. To preserve decidability, the obtained $\mathcal{EL}^{++}$ rules are then mapped into safe Datalog ones. In the second approach, DL constructors are defined with EDB facts and Datalog rules are used to denote fuzzy $\mathcal{EL}^{++}$ instance checking deduction rules. DL axioms and assertions are translated into Datalog EDB facts and Datalog rules are used to derive conclusions about them.

### 2. Fuzzy $\mathcal{EL}^{++}$ and crisp Datalog programs

$\mathcal{EL}^{++}$ is a tractable (polynomial-time decidable) description logic proposed in [2]. The semantic of fuzzy description logic is based on an interpretation $I = (\Delta^I, \bot^I)$. $\Delta^I$ is a nonempty set called the domain whereas $\bot^I$ is a function that associates to every concept $C$ a membership function $C^I : \Delta^I \to [0, 1]$; and to every role $R$ a membership function $R^I : \Delta^I \times \Delta^I \to [0, 1]$; and as for the crisp case, to every individual an element of $\Delta^I$ [14]. $C^I$ (resp. $R^I$) is thus interpreted as the membership degree function of fuzzy concept $C$ (respectively role $R$). $C^I(d)$ gives the degree of $d$ $(d \in \Delta^I)$ being an element of the fuzzy concept $C$ under interpretation $I$. A concrete domain is considered as a fuzzy set. A fuzzy domain is defined with a pair $(\Delta_D, \Phi_D)$ where $\Delta_D$ is an interpretation domain and $\Phi_D$ is a set of fuzzy predicates $d$ of arity $n$ and an interpretation $d^O : \Delta_D^n \to [0, 1]$. For example, $\text{Large}$ is a fuzzy predicate which measures the degree of largeness of the width of a given country $c$ and may be defined with a trapezoidal membership function as shown in figure 1. We may define the fuzzy concept $\text{GreatCountry}$ is a fuzzy concept and is defined as follows:

$$\text{GreatCountry} = \text{Country} \cap \exists \text{width. Large}$$

The fuzzy predicates used in this paper are defined as follows:

- Trapezoidal : $\text{trz}(u,v,w,d)$
- Triangular : $\text{tri}(u,v,w)$
- Left shoulder : $\text{ls}(u,v)$
- Right shoulder : $\text{rs}(u,v)$
The fuzzy \( \mathcal{EL}^{++} \) description logic is based on four disjunct finite sets: the set of fuzzy concept names \( N_C \), the set of fuzzy role names \( N_R \), the set of individual names \( N_I \) and the set of fuzzy predicates \( \Phi_D \). We give in Table 1 the syntax and the semantics of fuzzy \( \mathcal{EL}^{++} \).

<table>
<thead>
<tr>
<th>Syntax/DL</th>
<th>Semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>( \top^f = 0 )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot^f = 1 )</td>
</tr>
<tr>
<td>( C_1 \sqcap C_2 )</td>
<td>( (C_1 \sqcap C_2)^f(a) = \min(C_1^f(a), C_2^f(a)) )</td>
</tr>
<tr>
<td>( \exists r.C )</td>
<td>( (\exists r.C)^f(a) = \sup_{a, b} { \min(r^f(a, b), C^f(b)) } )</td>
</tr>
<tr>
<td>( C_1 \sqsubseteq C_2 )</td>
<td>( \alpha \leq \inf_{a, b} { \psi(C_1^f(a), C_2^f(a)) } )</td>
</tr>
</tbody>
</table>

Given a fuzzy concept and roles axioms \( \Phi_D \), we get in the table 1, column semantic are satisfied.

A fuzzy CBox is in normal form if:

1. all GCIs have one of the following forms:
   \[
   C_1 \sqsubseteq \alpha \quad D, \quad C_1 \sqsubseteq \exists \alpha \quad C_2, \\
   C_1 \sqcap C_2 \sqsubseteq \alpha \quad D, \quad \exists \alpha \quad C_1 \sqsubseteq \alpha \quad D 
   \]
   where, \( C_1, C_2 \in BC_C \) and \( D \in BC_C \cup \{ \bot \} \).
2. all RIs are of the form \( r \sqsubseteq s \) or \( r_1 \circ r_2 \sqsubseteq s \) (we use a crisp definition of RIs).

3. Homogenous approach for integrating fuzzy \( \mathcal{EL}^{++} \) and Datalog programs

We propose in this section an homogenous approach for integration fuzzy \( \mathcal{EL}^{++} \) description logic and crisp Datalog programs. Fuzzy concept and roles axioms and assertions are then represented used crisp Datalog rules. The algorithm that we adopt to transform an \( \mathcal{EL}^{++} \) fuzzy knowledge base into a Datalog program is defined as follows:

1. Normalization of fuzzy \( \mathcal{EL}^{++} \).
2. Mapping of normalized fuzzy \( \mathcal{EL}^{++} \) into crisp \( \mathcal{EL}^{++} \).
3. Definition of \( \mathcal{EL}^{++} \) rules.
4. Mapping \( \mathcal{EL}^{++} \) rules into Datalog rules.

A fuzzy CBox is in normal form if:

1. all GCIs have one of the following forms:
   \[
   C_1 \sqsubseteq \alpha \quad D, \quad C_1 \sqsubseteq \exists \alpha \quad C_2, \\
   C_1 \sqcap C_2 \sqsubseteq \alpha \quad D, \quad \exists \alpha \quad C_1 \sqsubseteq \alpha \quad D 
   \]

We present in this section a crisp representation of fuzzy \( \mathcal{EL}^{++} \) based on the works of Bobillo and al. [3,5]. The principle of crisp representation of fuzzy DLs is based on the definition of new crisp concepts and roles as \( \alpha \)-cuts of fuzzy ones. The set of degrees of degrees to be considered for the definition of theses \( \alpha \)-cuts depends on the semantic used for the definition of logic operators \( \{ \sqcap, \sqcup, \circ, \Rightarrow \} \). That is, we obtain different sets of d.o.m if we transform a fuzzy DL based on Gödel or Zadeh [4] or on Lukasiewicz [6]. Equally, the number of degrees to be considered may influence the complexity of reasoning as the cardinality of the obtained crisp knowledge base depends on the number of d.o.m and by the way the number of obtained \( \alpha \)-cuts.

Given a fuzzy \( \mathcal{EL}^{++} \) knowledge base \( \mathcal{F} \mathcal{K} \), the principle of transformation of this \( \mathcal{F} \mathcal{K} \) into a crisp knowledge base \( \mathcal{K} \) is defined as follows:

- Definition of crisp concepts and roles as \( \alpha \)-cuts of fuzzy ones from \( \mathcal{F} \mathcal{K} \). Based on Zadeh semantics, the set of d.o.m \( N \) to be considered is defined as follows:
  \[
  \gamma = X^{FK} \cup \{ 1 - \alpha | \alpha < X^{FK} \} \\
  X^{FK} = \{ 0, 0.5, 1 \} \cup \{ \gamma | \gamma \in FK \} 
  \]
For each fuzzy concept \( C \in \mathcal{FC} \) and \( \alpha, \beta \in [0, 1] \) defined in \( \mathcal{N} \), two crisp concepts \( C_{>\alpha} \) and \( C_{>\beta} \) are defined in the crisp knowledge base \( \mathcal{K} \). These two concepts correspond to the set of individuals which belong respectively to \( C \) with a degree \( \geq \alpha > \beta \). The same is for fuzzy roles.

- In order to preserve reasoning, new subsumption axioms are defined between crisp concepts. Then, for each fuzzy concept \( C \) and for each \( \alpha \in [0, 1] \) and \( \beta \in [0, 1] \) defined in \( \mathcal{N} \) with \( \alpha > \beta \), the following axioms are defined in the knowledge base \( \mathcal{K} \):

\[
C_{>\alpha} \sqsubseteq C_{>\beta}
\]

Equally, for each \( \gamma \in \mathcal{N} \), we have:

\[
C_{>\gamma} \sqsubseteq C_{>\gamma}
\]

The same is for fuzzy roles.

- Crisp rewriting of fuzzy axioms as shown in Table 2.

<table>
<thead>
<tr>
<th>( \mathcal{EL}++ ) fluo</th>
<th>( \mathcal{EL}++ ) exacte</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ABox</strong></td>
<td></td>
</tr>
<tr>
<td>( (a : C ; a) )</td>
<td>( (a : C_{&gt;\alpha}) )</td>
</tr>
<tr>
<td>( ((a, b) : R_{&gt;\alpha}) )</td>
<td>( ((a, b) : R_{&gt;\alpha}) )</td>
</tr>
<tr>
<td><strong>CBox</strong></td>
<td></td>
</tr>
<tr>
<td>( C \sqsubseteq a ) E</td>
<td>( C_{&gt;\alpha} \sqsubseteq E_{&gt;\alpha} )</td>
</tr>
<tr>
<td>( C \sqcap D \sqsubseteq a ) E</td>
<td>( C_{&gt;\alpha} \sqcap D_{&gt;\alpha} \sqsubseteq E_{&gt;\alpha} )</td>
</tr>
<tr>
<td>( R \sqsubseteq a ) E</td>
<td>( R_{&gt;\alpha} \sqsubseteq a ) E</td>
</tr>
<tr>
<td>( {a_1/a_1, \ldots, a_n/a_n} )</td>
<td>( {a_1/a_1 \geq 1-a_1, \ldots, a_n/a_n } )</td>
</tr>
<tr>
<td>( p(f_1, \ldots, f_k) \sqsubseteq a ) C</td>
<td>( p_{&gt;\alpha}(f_1, \ldots, f_k) \sqsubseteq C_{&gt;\alpha} )</td>
</tr>
<tr>
<td>( r_1 \circ r_2 \sqsubseteq s )</td>
<td>( r_1 \circ r_2 \sqsubseteq s )</td>
</tr>
</tbody>
</table>

Table 2: Crisp rewriting fuzzy normalized \( \mathcal{EL}++ \) axioms.

**Theorem 3.1** Crisp representation of fuzzy \( \mathcal{EL}++ \) under Zadeh semantic is correct.

### 3.2. \( \mathcal{EL}++ \) rules definition

We present in this section the third step of fuzzy \( \mathcal{EL}++ \) Datalog rewriting algorithm. The \( \mathcal{EL}++ \) knowledge base \( \mathcal{K} \) is mapped into an \( \mathcal{EL}++ \) rule base \( \mathcal{RB} \). We propose in the following a definition for \( \mathcal{EL}++ \) rules.

**Definition 3.1** \( \mathcal{EL}++ \) rules. Given a description logic \( DL \) with a set of concept expressions \( \mathcal{C} \), a set of roles expressions \( \mathcal{R} \), a set of individual names \( \mathcal{I}_1 \), a finite set of variables \( V \) and a set of concrete predicates \( \mathcal{P}^D \) and a set of predicate symbols \( \mathcal{P}_P \). We have, \( \mathcal{C} \cup \mathcal{R} \cup \mathcal{P}_P \subseteq \mathcal{N}_P \). A term in an element of \( V \cup \mathcal{N}_1 \cup \Delta^D \). We consider two terms \( t \) and \( u \) defined in \( V \cup \mathcal{N}_1 \). \( n \) terms defined in \( \Delta^D \) a concept \( C \in \mathcal{C} \) and a role \( r \in \mathcal{R} \) and a concrete predicate of arity \( n \) \( p \in \mathcal{P}_P \). An \( \mathcal{EL}++ \) atom has the forms \( C(t), R(t, u) \) et \( p(t_1, \ldots, t_n) \). A non \( \mathcal{EL}++ \) atom is defined with predicate symbols \( \notin \mathcal{C} \cup \mathcal{R} \cup \mathcal{P}_P \). An \( \mathcal{EL}++ \) rule has the following form:

\[
H \leftarrow B
\]

where \( B \) and \( H \) are conjunction of \( \mathcal{EL}++ \) atoms.

<table>
<thead>
<tr>
<th>( \mathcal{EL}++ ) fluo</th>
<th>( \mathcal{EL}++ ) exacte</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ABox</strong></td>
<td></td>
</tr>
<tr>
<td>( (a : C_{&gt;\alpha}) )</td>
<td>( C_{&gt;\alpha}(a) )</td>
</tr>
<tr>
<td>( ((a, b) : R_{&gt;\alpha}) )</td>
<td>( R_{&gt;\alpha}(a, b) )</td>
</tr>
<tr>
<td><strong>CBox</strong></td>
<td></td>
</tr>
<tr>
<td>( C \sqsubseteq a ) E</td>
<td>( C_{&gt;\alpha} \sqsubseteq E_{&gt;\alpha} )</td>
</tr>
<tr>
<td>( C \sqcap D \sqsubseteq a ) E</td>
<td>( C_{&gt;\alpha} \sqcap D_{&gt;\alpha} \sqsubseteq E_{&gt;\alpha} )</td>
</tr>
<tr>
<td>( R \sqsubseteq a ) E</td>
<td>( R_{&gt;\alpha} \sqsubseteq a ) E</td>
</tr>
<tr>
<td>( {a_1/a_1, \ldots, a_n/a_n} )</td>
<td>( {a_1/a_1 \geq 1-a_1, \ldots, a_n/a_n } )</td>
</tr>
<tr>
<td>( p(f_1, \ldots, f_k) \sqsubseteq a ) C</td>
<td>( p_{&gt;\alpha}(f_1, \ldots, f_k) \sqsubseteq C_{&gt;\alpha} )</td>
</tr>
<tr>
<td>( r_1 \circ r_2 \sqsubseteq s )</td>
<td>( r_1 \circ r_2 \sqsubseteq s )</td>
</tr>
</tbody>
</table>

Table 3: Mapping \( \mathcal{EL}++ \) into \( \mathcal{EL}++ \) rules.

We present in Table 3, the mapping of an \( \mathcal{EL}++ \) knowledge base \( \mathcal{K} \) into an \( \mathcal{EL}++ \) rule base \( \mathcal{RB} \). We can see \( \mathcal{K} \) are \( \mathcal{RB} \) semantically equivalent and my that is induce the same facts. As we can see, each \( \mathcal{EL}++ \) assertion is transformed into a fact. Except the one defined with nominals, each \( \mathcal{EL}++ \) axiom is transformed into one rule. The axiom defined with nominals is transformed into \( n \) facts where \( n \) corresponds to the number of nominals used in the axiom. We may then deduce the following proposition.

**Proposition 3.1** Given an \( \mathcal{EL}++ \) knowledge base \( \mathcal{KB} \), the algorithm maps \( \mathcal{KB} \) into an \( \mathcal{EL}++ \) rule base \( \mathcal{RB} \) in linear time and is correct. Given, a concept or role assertion \( \tau \) and \( \rho(\tau) \) the fact obtained as a result of the mapping of \( \tau \), we have: \( \mathcal{KB} \models \tau \iff \mathcal{RB} \models \rho(\tau) \).

### 3.3. Mapping \( \mathcal{EL}++ \) rules into safe Datalog rules

To define the syntax of Datalog we need some basic concepts. A term is defined as follows:

- Every constant is a term.
– For every n-ary predicate \( p \) and vector \((t_1, \ldots, t_n)\) of terms, \( p(t_1, \ldots, t_n) \) is a term.
– A constant term contains no variables.

An atom has the form \( p(t_1, \ldots, t_n) \) where the \( t_i \)'s are terms and \( p \) is a predicate symbol of arity \( n \). An atom is ground if no variables occur on it.

**Definition 3.2** A Datalog program \( P \) is a finite set of rules (or function-free Horn clauses) of the form:

\[
a \leftarrow b_1, \ldots, b_n
\]

where \( a \) and \( b_i \) are positive atoms. The left-hand side is called the head of the rule and the right-hand side is called the body of the rule. When the body of the rule is empty, the rule is called a fact.

Given a Datalog program \( P \), the predicates that appear only in the body of the program’s rules are called edb (extensional database) predicates, while those that appear in the head of a given rule are called idb (intensional database) predicates.

We present in this section the last step of the algorithm which consists on the mapping of \( \mathcal{EL}^+\) rules into Datalog rules. As a description logic, the assumption world of \( \mathcal{EL}^+\) logic is made on Open World Assumption opposed to the Closed World Assumption of Datalog programs. In a Datalog program, all the instances of a concept or a role are defined in the assertional level, while an \( \mathcal{EL}^+\) knowledge base may induce new “unknown” instances not explicitly defined in the assertion level. In logic programs, the definition of a model with an infinity of elements may lead to a problem of calculability and complexity. In order to resolve this problem, we need to defined then \( \mathcal{EL}^+\) safe rules.

**Definition 3.3** A rule \( r \) is safe if all the variables that occur in the head of the rule occur at least in one atom of the body of the rule.

**Definition 3.4** A rule \( r \) is \( \mathcal{EL}^+\) safe if all its variables occur in at least one non-\( \mathcal{EL}^+\) predicate in the body of the rule.

The description logic \( \mathcal{EL}^+\) supports the definition of concrete domains. We recall that, concrete domains are used in fuzzy \( \mathcal{EL}^+\) to define membership functions. In Datalog, concrete domains are defined with what we call Datalog built-ins. We define in the following two restrictions made on Datalog concrete predicate:

1. Concrete predicates should only appear in the body of a rule.

2. All the variables of a rule have to appear in at least one abstract predicate.

**Definition 3.5** An \( \mathcal{EL}^+\) rule \( r \) is a safe Datalog rule iff:

– All the predicates defined on \( r \) have one of the following forms:

\[
C(t), r(t, u), p(t_1, \ldots, t_n), O(t)
\]

\( C \in \mathcal{C}, r \in \mathcal{R}, p \in \mathcal{P}^D, O \in \text{EDB} \) and \( t, u \in N \cup V \cup N_I \) and \( t_i \in \Delta^R \).

– All the variables that occur in the head of \( r \) occur in at least one atom in the body of the rule.

– All the variables that occur in the head of a \( r \) should occur in at least one non-\( \mathcal{EL}^+\) atom in the body of the rule.

– Concrete predicates occur only in the body of the rule.

– All the variables of a rule have to occur in at least one abstract predicate.

Concrete predicate used in \( \mathcal{EL}^+\) rules are based on fuzzy concrete predicates used to define membership functions. The fuzzy concrete predicates are transformed into crisp ones based on the same principle adopted for the definition of fuzzy concepts and roles. Crisp concrete predicates are defined as \( \alpha \)-cuts of fuzzy concrete predicates. We propose in this section an approach to represent fuzzy concrete predicates (membership functions) and their \( \alpha \)-cuts with Datalog built-in. The idea is to define membership functions corresponding to theses concrete predicates with Datalog rules. We present in table 4, the principle of this approach applied to fuzzy concrete predicates corresponding to trapezoidal and triangular membership functions.

**Table 4**

<table>
<thead>
<tr>
<th>Trapezoidal</th>
<th>T : a, b, c, d</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(t, 0) )</td>
<td>( t \leq a )</td>
</tr>
<tr>
<td>( T(t, 0) )</td>
<td>( t \geq d )</td>
</tr>
<tr>
<td>( T(t, 1) )</td>
<td>( t &gt; c, t &lt; d )</td>
</tr>
<tr>
<td>( T(t, v) )</td>
<td>( t &gt; a, t &lt; b, t = v \ast (b - a) + a )</td>
</tr>
<tr>
<td>( T(t, v) )</td>
<td>( t &gt; c, t &lt; d, t = v \ast (d - c) + d )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Triangular</th>
<th>A : a, m, b</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A(t, 0) )</td>
<td>( t \leq a )</td>
</tr>
<tr>
<td>( A(t, 0) )</td>
<td>( t \geq b )</td>
</tr>
<tr>
<td>( A(t, v) )</td>
<td>( t &gt; a, t \leq m, t = v \ast (m - a) + a )</td>
</tr>
<tr>
<td>( A(t, v) )</td>
<td>( m &lt; t, t &lt; b, t = v \ast (b - m) + b )</td>
</tr>
</tbody>
</table>

Rewriting fuzzy concrete predicates with Datalog.
Example 1 We consider the two fuzzy concrete predicates Young and Adult defined with trapezoidal membership functions: \(T : 15, 20, 30, 35\) for Young concrete predicate and \(T : 20, 25, 55, 60\) for Adult concrete predicate. The definition of these fuzzy concrete predicate with Datalog is realized as follows:

\[
\begin{align*}
\text{Young}(t, 0) & \leftarrow t \leq 15 \\
\text{Young}(t, 1) & \leftarrow 20, t < 35 \\
\text{Young}(t, v) & \leftarrow 15, t < 20, t = v \cdot 5 + 15 \\
\text{Young}(t, v) & \leftarrow 30, t < 35, t = v \cdot 5 + 35 \\
\text{Adult}(t, 0) & \leftarrow t \leq 20 \\
\text{Adult}(t, 1) & \leftarrow t > 25, t < 55 \\
\text{Adult}(t, v) & \leftarrow 20, t < 25, t = v \cdot 5 + 15 \\
\text{Adult}(t, v) & \leftarrow 55, t < 60, t = v \cdot 5 + 60
\end{align*}
\]

The \(\alpha\)-cuts concrete predicates may be defined as follows:

\[
\text{Young}_{\geq \alpha}(t) \leftarrow \text{Young}(t, v), v \geq \alpha
\]

Based on this principle, the \(\alpha\)-cuts of a fuzzy concrete predicate \(p\) may be defined with Datalog rules as follows:

\[
\begin{align*}
p_{\geq \alpha}(t) & \leftarrow p(t, v), v \geq \alpha \\
p_{> \alpha}(t) & \leftarrow p(t, v), v > \alpha
\end{align*}
\]

Except the rule \(p_{\geq \alpha}(f_1, ..., f_k)(t) \leftarrow C_{>1-\alpha}(t)\), the \(\mathcal{EL}^{++}\) rules of table 3 are safe Datalog rules. In fact, the rule \(p_{\geq \alpha}(f_1, ..., f_k)(t) \leftarrow C_{>1-\alpha}(t)\) is not a safe Datalog rule as its head contain a concrete predicate. This rule corresponds to the fuzzy axiom \(C \subseteq p\) which is transformed into a crisp axiom \(C_{>1-\alpha} \subseteq p_{\geq \alpha}(f_1, ..., f_k)\). This subsumption may be written as follows [?]:

\[
C_{>1-\alpha} \sqcap \neg p_{\geq \alpha}(f_1, ..., f_k) \subseteq \bot
\]

As \(\neg p_{\geq \alpha}(f_1, ..., f_k) = p_{\leq \alpha}(f_1, ..., f_k) (\mu_\neg p(f_1, ..., f_k)(t) = 1 - \mu p(f_1, ..., f_k)(t))\), we obtain after that:

\[
C_{>1-\alpha} \sqcap p_{\leq \alpha}(f_1, ..., f_k) \subseteq \bot
\]

We may rewrite this crisp inclusion with a safe Datalog rule as follows:

\[
\bot \leftarrow C_{>1-\alpha}(t), p_{\leq \alpha}(f_1, ..., f_k)(t)
\]

Example 2 Let us consider the fuzzy inclusion \(\text{Young}_{\geq 0.6}(t)\) and \(\text{Adult}_{\geq 0.6}(t)\) which is transformed into a crisp axiom

\[
\bot \leftarrow \text{Adult}_{<0.6}(t), \text{Young}_{\geq 0.4}(t)
\]

This rule is transformed into a safe Datalog rule as follow:

\[
\bot \leftarrow \text{Adult}_{<0.6}(t), \text{Young}_{\geq 0.4}(t)
\]

\(\text{Adult}_{\geq 0.6}(t) \equiv \text{Young}_{\geq 0.4}(t)\)

\(\text{Adult}_{\geq 0.6}(t)\) and \(\text{Young}_{\geq 0.4}(t)\) are defined based on example 1 as follows:

\[
\begin{align*}
\text{Adult}_{<0.6}(t) & \leftarrow \text{Adult}(t, v), v < 0.6 \\
\text{Young}_{\geq 0.4}(t) & \leftarrow \text{Young}(t, v), v > 0.4
\end{align*}
\]

Given an \(\mathcal{EL}^{++}\) rule knowledge base \(\mathcal{RB}\), the rewriting of \(\mathcal{RB}\) into a safe Datalog program \(\mathcal{P}(\mathcal{RB})\) is defined as follows:

1. \(\mathcal{EL}^{++}\) rules of the form \(p_{\geq \alpha}(f_1, ..., f_k)(t) \leftarrow C_{>1-\alpha}(t)\) are mapped into rules of the form:

\[
\bot \leftarrow C_{>1-\alpha}(t), p_{\leq \alpha}(f_1, ..., f_k)(t)
\]

2. Define fuzzy concrete predicates with Datalog built-in.

3. Define rules for each \(\alpha\)-cut concrete predicate.

4. For each variable \(t\) defined in a rule, add a predicate \(O(t)\) in the head of the rule (safe \(\mathcal{EL}^{++}\) rule see definition 3.4).

5. Add a fact \(O(a)\) for each individual \(a \in N_\ell\).

Proposition 3.2 The transformation of an \(\mathcal{EL}^{++}\) rule base \(\mathcal{RB}\) into a Datalog program \(\mathcal{P}(\mathcal{RB})\) is correct.

3.4. Discussion of the homogenous approach and implementation issues

In this section, we presented an homogenous approach for integrating the fuzzy \(\mathcal{EL}^{++}\) description logic and Datalog programs. The particularity of this approach consists on the fact that the integration is realized between fuzzy and crisp based logics. In fact, many implementation exist of crisp Datalog systems which is not the case of fuzzy ones. The problem that we found in this approach is the number of degrees to be considered to define the \(\alpha\)-cuts of fuzzy concepts and roles. In fact, this number influence the complexity of reasoning as it depends on the size of the knowledge base. As it is shown in [4], while the size of a crisp TBox is larger then the size of the corresponding fuzzy TBox, an ABox has the same number of axioms as the original fuzzy ABox. Concerning reasoning, this approach need one rule based reasoner to realize inferences on TBox and on ABox. Our rule based representation of fuzzy \(\mathcal{EL}^{++}\) may be enriched with OWL-EL.
constructors such as local reflexivity or concept products. In fact, we may use the ELP rule language which is a tractable fragment of the OWL2 supporting the majority of OWL-EL constructors [11]. The authors of [11] propose a polynomial reduction of ELP knowledge bases to a specific kind of Datalog programs that can be evaluated in polynomial time.

4. Hybrid approach for integrating fuzzy $\mathcal{EL}^{++}$ and crisp Datalog programs

We present in this section an hybrid approach to represent fuzzy $\mathcal{EL}^{++}$ knowledge base using crisp Datalog programs. In this approach, DL axioms and assertions are translated into Datalog EDB facts and Datalog rules are used to derive conclusions about them. We present in table 5 the translation of fuzzy DL constructors into Datalog facts. We may classify these facts into three classes:

- Terminological predicates: subClassOf, conjunctionOf, extRest, extRest*, subPropertyOf, propertyChain. Theses facts correspond to fuzzy axioms defined with fuzzy abstract roles and concepts.
- Concrete predicates: these predicates correspond to fuzzy concrete predicates used to define membership functions (trapezoidal, triangular, left shoulder, right shoulder).
- Assertional predicates: these predicates correspond to fuzzy assertions in description logics. We have a particular case in which fuzzy axiom \( \{\alpha_1/a_1, \ldots, \alpha_n/a_n\} \sqsubseteq_c D \) is translated into a set of fuzzy assertional predicates. Equally, an assertional predicate may correspond to a concrete predicate if the fuzzy concept for which the assertion is defined correspond to a fuzzy concrete predicate (i.e. the fuzzy assertion \( \langle a:Young, \alpha \rangle \) where Young is a fuzzy concrete predicate is translated into the fact Young\((a, \alpha)\)).

Based on our classification of fuzzy $\mathcal{EL}^{++}$ Datalog predicates, we define three classes of entailment rules: T-entailment rules, C-entailment rules and A-entailment rules.

4.1. T-entailment rules

T-entailment rules contain only fuzzy terminological predicates in their conclusion. Theses rules correspond to completion rules used in fuzzy $\mathcal{EL}^{++}$. We present in table 6 T-entailment rules defined with Datalog. We define in the following the predicates symbols:

- $SC$: SubClassOf predicate.
- $SP$: SubPropertyOf predicate.
- $ER$ and $ER^*$: Existential restriction.
- $Conj(C_1, C_2, D, \alpha)$: ConjunctionOf predicate.
- $PC(r, s, t)$: PropertyChain predicate.

### T-entailment rules

<table>
<thead>
<tr>
<th>Fuzzy DL</th>
<th>Crisp datalog facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r \in N_R$</td>
<td>role(r)</td>
</tr>
<tr>
<td>$C \in N_C$</td>
<td>class(C)</td>
</tr>
<tr>
<td>$a \in N_I$</td>
<td>nominal(a)</td>
</tr>
<tr>
<td>( \langle a, C, \alpha \rangle )</td>
<td>type(a,C,\alpha)</td>
</tr>
<tr>
<td>( \langle a, b : r, \alpha \rangle )</td>
<td>triplet(a,b,r,\alpha)</td>
</tr>
<tr>
<td>$C \sqsubseteq_a D$</td>
<td>subClassOf(C,D,\alpha)</td>
</tr>
<tr>
<td>$C_1 \sqcap C_2 \sqsubseteq_a D$</td>
<td>conjunctionOf(C_1,C_2,D,\alpha)</td>
</tr>
<tr>
<td>$C_1 \sqsubseteq_a \exists r.D$</td>
<td>extRest(r,C_1,D,\alpha)</td>
</tr>
<tr>
<td>$\exists r.C_1 \sqsubseteq_a D$</td>
<td>extRest^*(C_1,r,C_2,\alpha)</td>
</tr>
<tr>
<td>${\alpha_1/a_1, \ldots, \alpha_n/a_n} \sqsubseteq_a D$</td>
<td>$\langle Da_n,\min(\alpha_1,\ldots,\alpha_n) \rangle$ for $i=1 \ldots n$</td>
</tr>
<tr>
<td>$r \sqsubseteq_a s$</td>
<td>subPropertyOf(r,s,\alpha)</td>
</tr>
<tr>
<td>$r \circ s \sqsubseteq t$</td>
<td>propertyChain(s,t,\alpha)</td>
</tr>
</tbody>
</table>

| $C, C_1, C_2 \in BC$, $D \in BC \cup \{\bot\}$, $r, s, t \in N_R$, $a, b, \alpha, \alpha_i \in N_I$ and $\alpha, \alpha_i \in [0, 1]$ |
| Table 5 |

Translation of fuzzy $\mathcal{EL}^{++}$ constructors into Datalog facts

4.2. A-entailment rules

A-entailment rules rules contain only fuzzy assertional predicates in their conclusion and are used for instance checking. We define in table 7 A-entailment rules defined with Datalog. The predicates symbols used in table 7 are defined as follows:

- $type(a, C, \alpha)$: a fuzzy concept assertion. $a$ is an instance of the fuzzy concept $C$ with a degree $\alpha$. 

| $C, C_1, C_2 \in BC$, $D \in BC \cup \{\bot\}$, $r, s, t \in N_R$ and $\alpha, \beta, \gamma \in [0, 1]$ |
| Table 6 |

A-entailment Datalog rules
– triplet \((a, r, b, \alpha)\): a fuzzy role assertion. \(a\) and \(b\) are related with \(r\) with a degree \(\alpha\).

### A-entailment rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{type}(a,\alpha) \rightarrow \text{nominal}(a))</td>
<td>Type (\alpha) is nominal (\alpha)</td>
</tr>
<tr>
<td>(\text{type}(a,\alpha) \rightarrow \text{SC}(C,D,\beta), \alpha \leq \beta)</td>
<td>Type (a) satisfies (\text{SC}(C,D,\beta))</td>
</tr>
<tr>
<td>(\text{type}(a,\alpha) \rightarrow \text{SC}(C,D,\beta))</td>
<td>Type (a) satisfies (\text{SC}(C,D,\beta))</td>
</tr>
<tr>
<td>(\text{type}(a,\alpha) \rightarrow \text{type}(a,\beta), \text{type}(b,\beta), \alpha \leq \beta)</td>
<td>Type (a) satisfies (\text{type}(a,\beta),\text{type}(b,\beta))</td>
</tr>
<tr>
<td>(\text{type}(a,\alpha) \rightarrow \text{type}(a,\beta), \text{nominal}(b), \text{type}(b,\beta), \alpha \leq \beta)</td>
<td>Type (a) satisfies (\text{type}(a,\beta),\text{nominal}(b),\text{type}(b,\beta))</td>
</tr>
<tr>
<td>(\text{type}(a,\alpha) \rightarrow \text{type}(a,\beta), \text{nominal}(b), \text{type}(b,\gamma), \beta \leq \alpha &lt; \gamma)</td>
<td>Type (a) satisfies (\text{type}(a,\beta),\text{nominal}(b),\text{type}(b,\gamma))</td>
</tr>
<tr>
<td>(\text{triplet}(a,s,b,\alpha) \rightarrow \text{triplet}(a,s,b,\beta))</td>
<td>Triplet (a) satisfies (\text{triplet}(a,s,b,\beta))</td>
</tr>
<tr>
<td>(\text{triplet}(a,s,b,\alpha) \rightarrow \text{triplet}(a,s,b,\beta), \text{nominal}(c), \text{triplet}(c,r,b,\gamma) &lt; \alpha &lt; \beta &lt; \gamma)</td>
<td>Triplet (a) satisfies (\text{triplet}(a,s,b,\beta),\text{nominal}(c),\text{triplet}(c,r,b,\gamma))</td>
</tr>
</tbody>
</table>

### C-entailment rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \rightarrow B)</td>
<td>A entails B</td>
</tr>
<tr>
<td>(\text{fuzzy} \mathcal{EL}^{++})</td>
<td>Fuzzy (\mathcal{EL}^{++})</td>
</tr>
<tr>
<td>(\text{C-entailment rules})</td>
<td>C-entailment rules</td>
</tr>
</tbody>
</table>

### 4.3 Discussion of the hybrid approach and implementation issues

We presented in this section an hybrid approach for integrating the fuzzy \(\mathcal{EL}^{++}\) description logic and crisp Datalog programs. In this approach, rule based systems are only used to reason on the ABox. We thus need fuzzy DL reasoner to reason on the TBox level. After translating a fuzzy \(\mathcal{EL}^{++}\) knowledge base to a Datalog facts, we may compute all the consequences of theses facts in a bottom-up fashion. Nevertheless, adopting the same principal for TBox reasoning (i.e., Classification) may lead to exponential complexity. Class subsumption, for example may be reduced to instance checking. That is \(C \subseteq D\) iff for each assumption \(D(a)\) we need to run instance checking for \(C(a)\). The use of separate reasoning system for the TBox and the ABox levels may require an interfacing mechanism in order to maintain the coherence of the knowledge base. The problem is more challenging here as we need to do that between fuzzy description logic reasoning system (i.e. fuzzyDL [7]) and crisp rule inference engine.

### 5. Conclusion and future works

We propose in this paper two approaches for integrating fuzzy \(\mathcal{EL}^{++}\) and crisp Datalog programs. This integration is realized to allow efficient instance checking and query answering over fuzzy EL-OWL. In the first approach, we propose an algorithm to realize the mapping of fuzzy \(\mathcal{EL}^{++}\) knowledge base to a crisp Datalog program. We first realize a crisp rewriting of fuzzy \(\mathcal{EL}^{++}\). We define after that \(\mathcal{EL}^{++}\) rules which are then rewritten using safe Datalog programs. In the second approach, fuzzy \(\mathcal{EL}^{++}\) axioms and assertions are defined with EDB facts and Datalog rules are used to denote fuzzy \(\mathcal{EL}^{++}\) instance checking deduction rules. That is, DL axioms and assertions are translated into Datalog EDB facts and Datalog rules are used to derive conclusions about them.

The size of the knowledge base in the homogenous approach depends on the number of degrees used to define the \(\alpha\)-cut of fuzzy concepts and roles. This number may lead to a crisp TBox larger then the size of the corresponding fuzzy TBox. Nevertheless, a crisp ABox has the same number of axioms as the original fuzzy ABox. The advantage of this approach consists on the fact that we use one reasoner to realize inferences on the TBox and the ABox levels. In the hybrid approach, the number of rules does not depends on the number of used membership degrees and crisp ABox still have the same size the original fuzzy one. The issue of this approach is the necessity of two reasoner systems for the TBox and the ABox levels. An interfacing mechanism is needed between the two reasoner in order to preserve the coherence of the hole knowledge base. The homogenous approach may be implemented using existent Datalog based system such as KAON. Actually, we worked on the development of an inference engine based on the hybrid approach using FLORA-2 and XSB. We project to use this inference engine for the implementation of an extended version our fuzzy semantic annotation query language (FSAQL) proposed in [1].
References


