A description logic approach for representing and reasoning on fuzzy object-oriented database models

Fu Zhang, Z.M. Ma,∗, Li Yan, Yu Wang

a College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
b School of Software, Northeastern University, Shenyang 110819, China

Received 2 October 2010; received in revised form 18 April 2011; accepted 24 April 2011
Available online 6 May 2011

Abstract

With the wide utilization of description logics (DLs), researches on applications of DLs have become the important research lines in the Semantic Web. In particular, DLs have been deeply studied in database modeling. This paper proposes a description logic approach to represent and reason on fuzzy object-oriented database (FOOD) models. After recalling some basic notions of FOOD models and a fuzzy DL called f-ALCIQ(D), we propose a kind of formal definition and semantics of FOOD models. On this basis, representation and reasoning of FOOD models with f-ALCIQ(D) is studied. We first translate a FOOD model and its corresponding database instance into an f-ALCIQ(D) knowledge base (i.e., TBox and ABox), and further implement a prototype translation tool called FOOD2DL. Then, based on the translated f-ALCIQ(D) knowledge bases, we investigate how to reason on FOOD models (e.g., consistency, subsumption, and redundancy) through the reasoning mechanism of f-ALCIQ(D). The formalization in f-ALCIQ(D) of FOOD models makes it possible to improve the ability of reasoning on database models by means of the inference services offered by fuzzy DLs.

Keywords: Fuzzy database; Fuzzy object-oriented database (FOOD) model; Fuzzy description logic; Representation; Reasoning

1. Introduction

Description logics (DLs, for short), which are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way, have been extensively applied to various fields such as the Semantic Web, medicine, and database [3]. Over the years, DLs have been studied in depth regarding the theoretical side, and the applied researches on DLs have increasingly attracted considerable attention. In particular, recent studies have proposed a number of ways in which DLs can be extremely useful in the development of database modeling.

Database modeling is the activity of specifying the structure of the data to be managed within an application [3]. Yeung [63] pointed out “Constructing a database system without a proper model is analogous to building a complex engineering structure without a blueprint”. However, the current-day database models (e.g., ER model and object-oriented database model) still suffer from some deficiencies such as inconsistency, inadequacy, and complexity,
which may result in a degradation of the quality of the design and/or increased development times and costs [4,9,20]. Hence, it is highly desirable to improve the ability of reasoning on database models, so that the reasoning tasks of database models (such as whether an object belongs to a class, or whether a class is the subclass of another class [3,4]) can be done automatically.

Among several ways to approach knowledge representation and reasoning, DLs are gaining a privileged place in recent years. Based on the high expressive power and effective reasoning services of DLs, it is not surprising that DLs are particularly adept at representing and reasoning on various database models. If database models can be translated into DL knowledge bases, the reasoning tasks of database models as mentioned above may be detected automatically through the reasoning mechanism of DLs instead of checking them by hand [4]. To this end, the relationships between DLs and several database models (e.g., ER model [3,20], temporal ER model [1], UML model [9], and object-oriented database model [2,20]) was investigated. The DL system FaCT has also been successfully used as a reasoning tool supporting the database modeling [3]. The detailed introduction about the database modeling with DLs can be found in Section 6.

However, the above researches were not sufficient for handling imprecise and uncertain information that is commonly found in many application domains. The representation of fuzzy information with fuzzy set theory has been addressed several decades ago by Zadeh [70]. Over the years, the fuzzy set theory has been extensively introduced into the Semantic Web, databases, and so on [33,46]. In order for DLs to handle imprecise and uncertain knowledge, lots of fuzzy DLs have been proposed, such as FALC [49], FALC(D) [53], f-ALCIQ [55], f-SHIN [54], f-SHOIN(D) [50], and f-SROIQ(D) [58] (see Section 6 in detail).

Also, many researches have already been made to represent and manipulate fuzzy data in various database models such as relational database models and object-oriented database models (see [37] in detail). In recent years, it has been found that the classical relational database model and its extension of fuzziness do not satisfy the need of handling complex objects with imprecision and uncertainty [32]. Fuzzy object-oriented database (FOOD) models are hereby developed [10,11,22,26,27,30,32,35,36,39,43,45,69]. The FOOD model, which can model imprecise and uncertain data and complex-valued attributes as well as complex relationships among objects, has been applied to the data and knowledge intensive applications such as CAD/CAM, expert system, and engineering designs. For a comprehensive review of what has been done in the development of FOOD models can found in Section 6.

With the wide utilization of FOOD models, how to reason on FOOD models more efficiently has increasingly received attention. However, current modeling tools cannot provide completely automated techniques for reasoning on FOOD models, and designers are responsible for checking the reasoning tasks of FOOD models (e.g., whether a fuzzy class is satisfiable, or whether a fuzzy class is the subclass of another fuzzy class) [26,30,36]. This may be a complex and time-consuming task by hand (observing that reasoning on object-oriented database models is already PSPACE-hard [20]). Therefore, being similar to the classical database models, if FOOD models can be translated into fuzzy DL knowledge bases, the reasoning problems of FOOD models above may be checked automatically through the reasoning mechanism of fuzzy DLs. This will facilitate the development of fuzzy database modeling and the realization of semantic interoperations between the existing database applications and the Semantic Web.

To this end, this paper proposes a DL approach for representing and reasoning on fuzzy object-oriented database (FOOD) models. Aiming at the characteristics and reasoning requirement of FOOD models, and considering the tradeoff between expressiveness and computation complexity of fuzzy DLs, a fuzzy DL called f-ALCIQ(D) (a sublanguage of the fuzzy DL f-SROIQ(D) [58,5]) is recalled first. On this basis, the paper makes the following main contributions:

**How to formalize FOOD models?** In Section 3, we propose a kind of formal definition and semantics of FOOD models, in order to establish the precise correspondences between FOOD models and the fuzzy DL. Based on the definitions of (fuzzy) object-oriented database models [3,20,68], the paper further considers both the structural and dynamic aspects of FOOD models, and also adds several new features to FOOD models, such as cardinality constraints on set-valued attributes, disjointness and covering constraints of classes, the inverse of attributes, and methods.

**How to represent FOOD models with f-ALCIQ(D)?** In Section 4, we propose a formal approach and a tool for representing FOOD models with f-ALCIQ(D), including: (i) translating a FOOD model into an f-ALCIQ(D) TBox at schema level; (ii) translating object instances w.r.t. the FOOD model into an f-ALCIQ(D) ABox at instance level; (iii) implementing a prototype tool called FOOD2DL, which can automatically translate FOOD models and object instances stored in databases into f-ALCIQ(D) knowledge bases (i.e., TBoxes and ABoxes).
• How to reason on FOOD models with f-ALCIQ(D)? The answer is (see Section 5): (i) giving formal definitions of reasoning tasks of FOOD models (e.g., consistency, subsumption, and redundancy); (ii) investigating how to reason on FOOD models with f-ALCIQ(D), i.e., how to reduce reasoning on FOOD models to reasoning on f-ALCIQ(D) knowledge bases, so that reasoning tasks of FOOD models may be checked through the reasoning mechanism of f-ALCIQ(D).

The remainder of this paper is organized as follows. Section 2 recalls some notions of FOOD models and the fuzzy DL f-ALCIQ(D). Section 3 proposes a formal definition and semantics of FOOD models. Section 4 investigates how to represent FOOD models with f-ALCIQ(D), and implements a prototype translation tool. Reasoning on FOOD models with f-ALCIQ(D) is studied in Section 5. Section 6 introduces related work. Section 7 shows conclusions and the future work.

2. Preliminaries

In this section, we recall some preliminaries on fuzzy object-oriented database models and the fuzzy description logic f-ALCIQ(D) [5,58].

2.1. The fuzzy object-oriented database model

Generally speaking, a fuzzy object-oriented database model is based on the notions of object, class, attribute, method, and inheritance, which are briefly introduced as follows:

• Object: An object is an entity of the real world, and it is characterized by a unique object identity (OID). Formally, objects that have at least one attribute whose value is fuzzy are fuzzy objects.
• Class: Objects with the same structure and behavior are grouped into a class. The domains of some attributes of a class may be fuzzy and a fuzzy class is formed, and an object may belong to the class with membership degree of [0,1].
• Inheritance: Classes can be organized into hierarchies of the kind superclass–subclass. In a subclass/superclass relationship, the member degree that any object belongs to the subclass must be less than or equal to the member degree that it belongs to the superclass.
• Attribute: Attributes form the data structure of an object, and their values define the inner state of the object. An attribute whose domain is fuzzy is called a fuzzy attribute.
• Method: A behavior is defined by a set of methods which are used to operate on the object state.
• Moreover, to model real-world situations more completely, it is often necessary to express more complex relations in fuzzy object-oriented database models, such as cardinality constraints on set-valued attributes, disjointness and covering constraints of classes, and the inverse of attributes (see Section 3 in detail).

In a fuzzy object-oriented database model, the fuzziness may occur at three different levels [22,30,35], i.e., the attribute level, the object/class level, and the subclass/superclass level.

(1) The first level is the attribute level. In a class of a fuzzy object-oriented database model, each attribute is associated with a type (called the domain of an attribute), and an attribute may be one of the following two cases:
• A non-fuzzy attribute, its domain may be a simple/basic type (such as integer, real and string) or a complex type (such as set type and object type).
• A fuzzy attribute, its domain is a fuzzy-type-based simple type or complex type. Only fuzzy attributes have fuzzy types and fuzzy attributes are explicitly indicated in a class definition (e.g., a fuzzy keyword FUZZY is appeared on the front of attributes indicating they are fuzzy attributes). A fuzzy domain of a fuzzy attribute may be a set of possibility distributions or a set of fuzzy linguistic terms (each fuzzy linguistic term is associated with a membership function over a basic type). For example, the domain of an attribute Age may be a crisp set of integers, a set of possibility distributions over the basic type integers, or a set of fuzzy linguistic terms over the basic type integers such as {very young, young, old, very old}.

(2) The second level is the object/class level, i.e., an object belongs to a class with a membership degree of [0, 1]. An additional attribute \( u \in [0, 1] \) is introduced into a class to represent the membership degree of an object to the class.
Fig. 1. A simplified fuzzy object-oriented database model (parts of classes and attributes only).

Fig. 2. A fuzzy database instance (i.e., a set of object instances) w.r.t. the model in Fig. 1 (several objects and the detailed information of the object o2 only).

(3) The third level is the subclass/superclass level. In a subclass/superclass relationship, the member degree that any object belongs to the subclass is less than or equal to the member degree that it belongs to the superclass. Therefore, fuzzy subclass/superclass relationships in a fuzzy object-oriented database model can be assessed by utilizing the inclusion degree of objects to the class [22,35].

In order to explain the fuzziness of three different levels above more directly, Fig. 1 shows a simplified fuzzy object-oriented database model (part of classes and attributes). Correspondingly, Fig. 2 gives a fuzzy database instance (i.e., a set of object instances), which includes only several objects and the detailed information of object o2. In Fig. 1: (i) $\triangle$ denotes the inheritance relationship between classes; (ii) the constraints disjointness and covering denote that two classes Young-Employee and Old-Employee are disjoint and the union of them completely covers Employee.

It should be noted that actually there are unsatisfiability and redundancy in the fuzzy object-oriented database model in Fig. 1 (see Section 5.1 in detail), and these problems will be detected by means of the reasoning mechanism of fuzzy description logics (see Section 5.2).

From Figs. 1 and 2, the fuzziness of three different levels above can be found more directly:

(1) The fuzziness of attribute level: A fuzzy keyword Fuzzy is appeared on the front of attributes Age and Salary indicating they are fuzzy attributes, and their values are represented by a fuzzy set (e.g., a fuzzy linguistic term Young) and a possibility distribution (e.g., \{5000/0.7, 6000/0.8\}), respectively.

(2) The fuzziness of object/class level: An attribute $u \in [0, 1]$ is introduced into a class to represent the membership degree of an object to the class, e.g., $u=0.95$ denotes that the object o2 belongs to the class Young-Employee with degree 0.95.
(3) The fuzziness of subclass/superclass level: As mentioned above, fuzzy subclass/superclass relationships can be assessed by utilizing the inclusion degree of objects to the class, e.g., since Young-Employee is a subclass of Employee, as shown in Fig. 2, the member degree that any object o1 or o2 belongs to Young-Employee is less than or equal to the member degree that it belongs to Employee.

In the following section, a fuzzy DL called f-ALCIQ(D) will be recalled in order to represent the fuzziness of different levels in fuzzy object-oriented database models as mentioned above.

2.2. The fuzzy description logic f-ALCIQ(D)

In this section, the existing fuzzy DL called f-ALCIQ(D) (i.e., a sublanguage of the fuzzy DL f-SROIQ(D) in [58,5]) is recalled. In this paper, f-ALCIQ(D) is chosen to represent fuzzy object-oriented database models by considering the characteristics of the models (see Sections 2.1 and 3) and the tradeoff between expressiveness and computation complexity of fuzzy DLs.

In the following, the syntax, semantics, knowledge base, and reasoning problems for f-ALCIQ(D) are recalled briefly.

2.2.1. The syntax and semantics of f-ALCIQ(D)

Definitions 1 and 2 give the syntax and semantics of f-ALCIQ(D). To begin with, we will recall two important elements of f-ALCIQ(D): fuzzy concrete domains D and fuzzy modifiers m.

The concrete domains [53] allow representing data types such as strings and integers. In fuzzy DLs, however, concrete domains are fuzzy sets. A fuzzy concrete domain D (or simple fuzzy domain) is a pair (A_D, Φ_D), where A_D is an interpretation domain and Φ_D is a set of fuzzy domain predicates d with a predefined arity n and an interpretation d^D : A^n_D → [0, 1], which is an n-ary fuzzy relation over A_D. For example, Young may be a fuzzy domain predicate over integers denoting the degree of youness of a person’s age with the formula [53]

\[
\text{Young}(x) = \begin{cases} 
1 & \text{if } x \leq 10 \\
(30 - x)/20 & \text{if } 10 \leq x \leq 30 \\
0 & \text{if } x \geq 30 
\end{cases}
\]  

(1)

A fuzzy modifier m likes very or slightly applies to a fuzzy set to change its membership function. Formally, a modifier is a function f_m : [0, 1] → [0, 1]. For example, we may define very(x) = x^2, while define slightly(x) = \sqrt{x}.

Based on the two elements above, f-ALCIQ(D) could represent concepts like “The age of Chris is 20 with degree 0.5” with the assertion (2):

\[\langle \text{age(Chris, 20)} = 0.5 \rangle\]  

(2)

And “Garfield is a young person with degree at least 0.7” with the axiom and assertion

\[
\{ \text{YoungPerson} = \text{Person} \sqcap \exists \text{age.Young} \} \quad \{ \text{YoungPerson(Garfield)} \geq 0.7 \}
\]  

(3)

Furthermore, we may represent the concept “a person who has the very high salary” as

\[
\{ \text{HighSalaryPerson} = \text{Person} \sqcap \exists \text{salary.very(High)} \}
\]  

(4)

where very is a fuzzy modifier and High is a fuzzy domain predicate over the domain of salary.

Definition 1 (Syntax). As usual, f-ALCIQ(D) assumes three alphabets of symbols, for concepts N_C, roles N_R and individuals N_I. The f-ALCIQ(D) arbitrary concepts (denoted C or D) can be built inductively from atomic concepts (A), atomic abstract roles (P), top concept ⊤, bottom concept ⊥, arbitrary abstract roles (R), concrete roles (T), fuzzy modifiers (m) and fuzzy domain predicates (d) as

\[
C, D \rightarrow \top | \bot | A | \neg C | C \sqcap D | C \sqcup D | \forall R.C | \exists R.C | n.R.C | \leq n.R.C | \geq n.R.C | \forall T.d | \exists T.d | \geq n.T.d | \leq n.T.d | m(C)
\]

where R → P|P^− (concrete role names do not have inverses), n ∈ N. To avoid considering roles R^−− we introduce the function Inv such that Inv(R) := R^− if R ∈ N_R, while Inv(R) := P if R = P^−. Moreover, the following
Definition 2 (Semantics). The semantics of $f$-ALCIQ($D$) are provided by a fuzzy interpretation $FI$. A fuzzy interpretation $FI$ with respect to a fuzzy domain $D$ is a pair $FI = (D^{FI}, \bullet^{FI})$ consisting of a nonempty set $D^{FI}$ (called the interpretation domain), disjoint with $D$, and of a fuzzy interpretation function $\bullet^{FI}$ that mapping:

- A fuzzy abstract individual $a$ to an element in $A^{FI}$,
- A fuzzy concrete individual $v$ to an element in $A_D$,
- A fuzzy concept $C$ to a membership degree function $C^{FI} : A^{FI} \rightarrow [0, 1]$,
- A fuzzy abstract role $R$ to a membership degree function $R^{FI} : A^{FI} \times A^{FI} \rightarrow [0, 1]$,
- A fuzzy concrete role $T$ to a membership degree function $T^{FI} : A^{FI} \times A_D \rightarrow [0, 1]$,
- An $n$-ary fuzzy domain predicate $d$ to the fuzzy relation $d^{FI} : A_D^n \rightarrow [0, 1]$,
- A fuzzy modifier $m$ to the function $f_m : [0, 1] \rightarrow [0, 1]$,
- The mapping $\bullet^{FI}$ is extended to concepts and roles as follows ($a, b \in A^{FI}, v \in A_D$):

\[
\begin{align*}
\top^{FI}(a) &= 1, \quad \bot^{FI}(a) = 0, \quad (\neg C)^{FI}(a) = 1 - C^{FI}(a) \\
(C \cap D)^{FI}(a) &= \min[C^{FI}(a), D^{FI}(a)] \\
(C \cup D)^{FI}(a) &= \max[C^{FI}(a), D^{FI}(a)] \\
(\forall R.C)^{FI}(a) &= \inf_{b \in A^{FI}} \{\max[1 - R^{FI}(a, b), C^{FI}(b)]\} \\
(\exists R.C)^{FI}(a) &= \sup_{b \in A^{FI}} \{\min[R^{FI}(a, b), C^{FI}(b)]\} \\
(\geq nR.C)^{FI}(a) &= \sup_{b_1, \ldots, b_n \in A^{FI}} \{\min[R^{FI}(a, b_i), C^{FI}(b_i)]\} \\
(\leq nR.C)^{FI}(a) &= \inf_{b_1, \ldots, b_{n+1} \in A^{FI}} \{\max[1 - R^{FI}(a, b_i), C^{FI}(b_i)]\} \\
(\forall T.d)^{FI}(a) &= \inf_{v \in A_D} \{\max[1 - T^{FI}(a, v), d^{FI}(v)]\} \\
(\exists T.d)^{FI}(a) &= \sup_{v \in A_D} \{\min[T^{FI}(a, v), d^{FI}(v)]\} \\
(\geq nT.d)^{FI}(a) &= \sup_{v_1, \ldots, v_n \in A_D} \{\min[T^{FI}(a, v_i), d^{FI}(v_i)]\} \\
(\leq nT.d)^{FI}(a) &= \inf_{v_1, \ldots, v_{n+1} \in A_D} \{\max[1 - T^{FI}(a, v_i), d^{FI}(v_i)]\} \\
(m(C))^{FI}(a) &= f_m(C^{FI}(a)) \\
(R^-)^{FI}(a, b) &= R^{FI}(b, a)
\end{align*}
\]

2.2.2. The knowledge base and reasoning in $f$-ALCIQ($D$)

An $f$-ALCIQ($D$) knowledge base $(f$-ALCIQ($D$) $KB$ consists of a TBox $T$ and an ABox $A$. The TBox introduces the terminology, i.e., the vocabulary of an application domain, while the ABox contains assertions about named individuals in terms of this vocabulary.
Definition 3 (TBox). A TBox $T$ is a set of fuzzy axioms: inclusions $C \sqsubseteq D$ or equalities $C \equiv D$, where $C$, $D$ are $f$-ALCIQ($D$) concepts. The semantics are interpreted as follows:

- $C \sqsubseteq D$, iff $\forall a \in A^{\text{fl}}, C^{\text{fl}}(a^{\text{fl}}) \leq D^{\text{fl}}(a^{\text{fl}})$;
- $C \equiv D$, iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., $\forall a \in A^{\text{fl}}, C^{\text{fl}}(a^{\text{fl}}) = D^{\text{fl}}(a^{\text{fl}})$.

A fuzzy interpretation $FI$ satisfies an $f$-ALCIQ($D$) TBox $T$ iff it satisfies all fuzzy concept axioms in $T$; in this case, we say that $FI$ is a model of $T$. An $f$-ALCIQ($D$) TBox $T$ allows to general concept inclusions (GCIs, i.e., axioms $C \subseteq D$) [56].

Definition 4 (ABox). An ABox $A$ is a set of fuzzy assertions: concept assertions $\langle C(a)\rangle \geq k$, role assertions $\langle R(a, b)\rangle \geq k$, and individual (in)equality assertions $a \approx b$ and $a \not\approx b$, where $a, b \in A^{\text{fl}}$, $v \in A_{D}$, $\bowtie \in \{\geq, >, \leq, <\}$, $k \in [0, 1]$. The semantics are interpreted as follows:

- A fuzzy interpretation $FI$ satisfies assertion $\langle C(a)\rangle \geq k$, iff $C^{\text{fl}}(a^{\text{fl}}) \geq k$;
- A fuzzy interpretation $FI$ satisfies assertion $\langle R(a, b)\rangle \geq k$, iff $R^{\text{fl}}(a^{\text{fl}}, b^{\text{fl}}) \geq k$;
- A fuzzy interpretation $FI$ satisfies assertion $\langle T(a, v)\rangle \geq k$, iff $T^{\text{fl}}(a^{\text{fl}}, v^{\text{fl}}) \geq k$;
- A fuzzy interpretation $FI$ satisfies assertion $a \approx b$ or $a \not\approx b$, iff $a^{\text{fl}} = b^{\text{fl}}$ or $a^{\text{fl}} \not\approx b^{\text{fl}}$.

A fuzzy interpretation $FI$ satisfies an $f$-ALCIQ($D$) ABox $A$ iff it satisfies all fuzzy assertions in $A$; in this case, we say that $FI$ is a model of $A$.

Definition 5 (KB). An $f$-ALCIQ($D$) $KB\Sigma$ is a pair $\langle T, A \rangle$, a fuzzy interpretation $FI$ satisfies $\Sigma$ if $FI$ satisfies all terminological axioms and assertions in $\Sigma$; in this case, $FI$ is called a model of $\Sigma$.

As in some fuzzy DLs (e.g., $f$-ALC [49], $f$-SHIN [54] and $f$-SROIQ($D$) [58,5]), the basic tasks we consider when reasoning over an $f$-ALCIQ($D$) $KB\Sigma = \langle T, A \rangle$ are as follows:

- $KB$ satisfiability: An $f$-ALCIQ($D$) $KB\Sigma$ is satisfiable (unsatisfiable) iff there is (not) a fuzzy interpretation $FI$ which satisfies all terminological axioms and assertions in $\Sigma$.
- Concept satisfiability: A fuzzy concept $C$ is satisfiable w.r.t. a TBox $T$ iff there is some model $FI$ of $T$ or which there is some $a \in A^{\text{fl}}$ such that $C^{\text{fl}}(a^{\text{fl}}) = n, n \in (0, 1]$.
- Concept subsumption: A fuzzy concept $C$ is subsumed by a fuzzy concept $D$ (written as $C \sqsubseteq D$) w.r.t. a TBox $T$ if for every model $FI$ of $T$ it holds that, $\forall a \in A^{\text{fl}}, C^{\text{fl}}(a^{\text{fl}}) \leq D^{\text{fl}}(a^{\text{fl}})$. Moreover, concept disjointness can be expressed by concept subsumption, i.e., two disjoint concepts $C$ and $D$ can be expressed as $C \sqcap D \subseteq \bot$.
- ABox consistency: An $f$-ALCIQ($D$) ABox $A$ is consistent w.r.t. a TBox $T$ if there is a model $FI$ of $T$ which is also a model of $A$.
- Entailment: An $f$-ALCIQ($D$) $KB\Sigma$ entails a fuzzy concept terminological axiom or a fuzzy assertion $\Psi$, written $\Sigma \models \Psi$, iff all models of $\Sigma$ satisfy $\Psi$.
- Retrieval: Given an $f$-ALCIQ($D$) $KB\Sigma$, a fuzzy concept $C$, and $n \in (0, 1]$, to retrieve all instances $a$ occurring in the ABox, such that $\Sigma$ entails $C(a) \geq n$, written as $\Sigma \models C(a) \geq n$.
- Greatest lower bound (glb): Given an assertion $\Psi$ and an $f$-ALCIQ($D$) $KB\Sigma$, the greatest lower bound of $\Psi$ w.r.t. $\Sigma$ is $\text{glb}(\Sigma, \Psi) = \sup\{n | \Sigma \models \Psi \geq n\}$, where $\sup \emptyset = 0$.

All of the reasoning problems above can be checked through the reasoning techniques for $f$-SROIQ($D$) (an $f$-ALCIQ($D$) extended language), which are not included here, refer to Refs. [58,7] in detail. Furthermore, the existing fuzzy reasoner DeLorean [8] also supports the fuzzy DL $f$-ALCIQ($D$) with general concept inclusions (GCIs).

Based on the fuzzy DL $f$-ALCIQ($D$), the later sections will investigate how to represent and reason on fuzzy object-oriented database models with $f$-ALCIQ($D$). Before that, we first formalize fuzzy object-oriented database models in the following section.

3. Formalization of fuzzy object-oriented database (FOOD) models

In this section, based on Section 2.1, we further add some new features to fuzzy object-oriented database models, and propose a kind of formal definition and semantics of fuzzy object-oriented database (FOOD) models, so as to establish the precise correspondences between FOOD models and the fuzzy DL $f$-ALCIQ($D$).
3.1. The formal definition of FOOD models

In the following, we propose a kind of formal definition of FOOD models. Here, based on the definitions of (fuzzy) object-oriented database models [3,20,68], we further consider both the structural and dynamic aspects of FOOD models, and also add some new features to FOOD models, such as cardinality constraints on set-valued attributes, disjointness and covering constraints of classes, the inverse of attributes, methods, and database instances.

**Definition 6.** A FOOD model is a tuple $FS = (FC_{FS}, FA_{FS}, FD_{FS})$ consisting of a set of classes, attributes and class declarations, where:

- $FC_{FS}$ is a set of fuzzy class names $FC$;
- $FA_{FS}$ is a set of fuzzy attribute names $FA$; each attribute $FA$ is associated with a domain $FB$ which may be a simple/complex type or a fuzzy-type-based simple/complex type (see Section 2.1); A fuzzy keyword $FUZZY$ is appeared on the front of attributes indicating these attributes are fuzzy attributes.
- $FD_{FS}$ is a set of fuzzy class declarations. For each fuzzy class $FC \in FC_{FS}$, $FD_{FS}$ contains exactly one such declaration:

  $\text{Class } FC \text{ is-a } FC_1, \ldots, FC_n \text{ type-is } FT,$

$FT \rightarrow FC|FB$

\[ \begin{array}{l}
\text{Union } FT_1, \ldots, FT_k \text{ (disjointness, covering) End } | \\
\text{Set-of } FT [(m_1, n_1), (m_2, n_2)] | \\
\text{Record } FA_1 : FT_1, \ldots, FA_k : FT_k, u : \text{Real End } | \\
\text{f}(P_1, \ldots, P_m) : R
\end{array} \]

where:

- (i) The $\text{type-is}$ part specifies the structure of the class FC through a type expression FT.
- (ii) The $\text{is-a}$ part, which is optional, denotes inheritance relationships between fuzzy classes.
- (iii) The $\text{Union ... End}$ part denotes a generalization relationship between a general class and several specific classes; the (disjointness, covering) part is optional, disjointness means that all the specific classes are mutually disjoint, and covering means that the union of specific classes completely covers the general class.
- (iv) $[(m_1, n_1), (m_2, n_2)]$ denotes the cardinality constraints of a set.
- (v) An additional attribute $u \in [0, 1]$ is introduced into a class to represent the membership degree of an object to the class.
- (vi) $f(P_1, \ldots, P_m) : R$ represents the method, where $f$ is the name of the method, $P_1, \ldots, P_m$ are types of $m$ parameters, and $R$ is the type of the result.

From Definition 6, it is shown that the fuzziness of different levels mentioned in Section 2.1 can be represented in the formal FOOD models. For example, the fuzziness of attribute level is represented by a fuzzy attribute FA associated with a fuzzy-type-based simple/complex type FB in the formal FOOD models, where a symbol “FUZZY” next to FA means that FA is a fuzzy attribute; the additional attribute $u \in [0, 1]$ is introduced into the formal FOOD models to represent the fuzziness of object/class level; and the $\text{is-a}$ part in the formal FOOD models can model the subclass/superclass relationship of fuzzy classes. The fuzziness of different levels in FOOD models can be found in Figs. 3 and 4 more directly.

Fig. 3 shows the FOOD model $FS_1$ modeling parts of the reality at a company from Fig. 1. The detailed instruction about Fig. 3 is as follows:

1. A fuzzy keyword $FUZZY$ is appeared on the front of attributes indicating these attributes may take fuzzy values.
2. The method $\text{IsDepartment( ):String}$ returns a possibility distribution value $\{\text{Department} / u_i\}$, which denotes that a Young-Employee works in the Department with degree $u_i \in [0, 1]$. The type of the parameter is $\text{null}$. 
Fig. 3. The FOOD model $FS_1$ modeling parts of the reality at a company from Fig. 1.

Fig. 4. A fragment of a fuzzy database instance w.r.t. the FOOD model $FS_1$ in Fig. 3 (including all the object instances in Fig. 2 and some additional information).

(3) $[(2, 6), (1, 1)]$ denotes that each Chief-Leader manages at least 2 and at most 6 Young-Employees, and each Young-Employee is managed by exactly one Chief-Leader.

Being the same as Fig. 1, there may be redundancy or other undesirable problems in the FOOD model $FS_1$, and these problems will be detected by means of the reasoning mechanism of $f$-ALCIQ($D$) (see Section 5).

Fig. 4 gives a fuzzy database instance (i.e., a set of object instances) w.r.t. the FOOD model $FS_1$ in Fig. 3, which includes all the object instances in Fig. 2 and further adds some detailed information of several objects:

(1) $u=0.85$ denotes that the object $o_3$ belongs to the class Chief-Leader with degree 0.85.
(2) $\beta = 0.7$ denotes that the object $o_3$ manages the object $o_1$ with degree 0.7.
(3) The attributes such as Age and Salary are fuzzy, e.g., the age of Chris is represented by a possibility distribution: 
{20/0.5, 28/0.1}, and the age of John is represented by a fuzzy linguistic term Young.
3.2. The semantics of FOOD models

The semantics of a FOOD model can be given by the fuzzy database state (i.e., the database instance), namely, by specifying which fuzzy database state/instance is consistent with the information structure of the FOOD model.

**Definition 7.** The semantics of a FOOD model $FS$ is given by a fuzzy database state $FJ$, which is defined by a fuzzy interpretation $FI_{FD} = (FV_{FJ}, \pi^{FJ}, \rho^{FJ}, \bullet^{FJ})$ consisting of:

1. A set $FV_{FJ} = FB_{FS} \cup FO^{FJ} \cup FV_{FR} \cup FV_{FS}$ of fuzzy values is inductively defined as follows:
   - $FB_{FS} = \bigcup_{i=1}^{n} FB_i$, where $FB_i$ is a crisp or fuzzy domain as mentioned in Section 2.1.
   - $FO^{FJ} = \{o_1/u_1, \ldots, o_n/u_n\}$, where $o_i$ is an object identifier denoting the real-world object, and each $o_i$ is associated with a membership degree $u_i$.
   - $FV_{FR}$ is a set of record values. A record value is denoted by $[FA_1 : FV_1, \ldots, FA_k : FV_k]$, where $FA_i$ is the attribute of the class, $FV_i \in FV_{FJ}$, $i \in \{1, \ldots, k\}$.
   - $FV_{FS}$ is a set of set-values. A set-value is denoted by $\{FV_1, \ldots, FV_k\}$, where $FV_i \in FV_{FJ}$, $i \in \{1, \ldots, k\}$.

2. A mapping $\pi^{FJ}$ assigns to each class $FC \in FC_{FS}$ a subset of $FO^{FJ}$.
3. A mapping $\rho^{FJ}$ assigns to each object in $FO^{FJ}$ a value of $FV_{FJ}$.
4. A function $\bullet^{FJ}$ maps each type expression $FT$ to a set $FT^{FJ}$ in such a way that:
   - If $FT$ is a class FC, then $FT^{FJ} = FC^{FJ} = \pi^{FJ}(FC)$.
   - If $FT$ is a union type $FT_1, \ldots, FT_k$ (disjointness, covering) End, then $FT^{FJ} = FT_1^{FJ} \cup \cdots \cup FT_k^{FJ}$ and $FT_{FJ}^i \cap FT_{FJ}^j = \emptyset$, where $i, j \in \{1, \ldots, k\}$, and $i \neq j$.
   - If $FT$ is a record type $FA_1 : FT_1, \ldots, FA_k : FT_k, u:Real$ End (resp. set type Set-of FT), then $FT^{FJ}$ is a set of record values $FV_{FR}$ (resp. set values $FV_{FS}$).

A fuzzy database state is considered *acceptable* if it satisfies all the constraints of a FOOD model. This is captured by the definition of legal fuzzy database state (see Definition 8).

**Definition 8.** A fuzzy database state $FJ$ is said to be legal w.r.t. a FOOD model $FS$, if for each class declaration: Class $FC$ is-a $FC_1, \ldots, FC_n$ type-is FT, it satisfies:

- $FC^{FJ}(o) \subseteq FC^{FJ}(o)$ for each object $o \in \pi^{FJ}(FC)$, $i \in \{1, \ldots, n\}$;
- $\rho^{FJ}(FC^{FJ}) \subseteq FT^{FJ}$.

Based on the formalization of FOOD models, the following sections investigate how to represent and reason on FOOD models with $f-ALCIQ(D)$.

4. Representation of FOOD models with $f-ALCIQ(D)$

This section proposes a formal approach and an automated tool to represent FOOD models with $f-ALCIQ(D)$, i.e., translate FOOD models into $f-ALCIQ(D)$ knowledge bases, including: (i) translating a FOOD model into an $f-ALCIQ(D)$ TBox at schema level (see Section 4.1); (ii) translating a fuzzy database instance (i.e., a set of object instances) w.r.t. the FOOD model into an $f-ALCIQ(D)$ ABox at instance level (see Section 4.2); (iii) developing a prototype translation tool named FOOD2DL, which can automatically extract the schema and instance information of a fuzzy object-oriented database and translate them into an $f-ALCIQ(D)$ knowledge base (see Section 4.3).

4.1. Translating FOOD model into $f-ALCIQ(D)$ TBox

Definition 9 gives a formal approach for translating a FOOD model into an $f-ALCIQ(D)$ TBox. Starting with the construction of *fuzzy concepts and fuzzy roles*, the approach induces a *set of fuzzy axioms* from the FOOD model.
In order to explicitly represent in f-ALCIQ(D) the type structure of classes in FOOD models, based on [3,20,59], we introduce in the f-ALCIQ(D) knowledge base concepts and roles with a specific meaning:

- The concepts AbstractClass, RecType, and SetType are used to denote fuzzy classes, record types, and set types, respectively.
- Furthermore, the concepts representing types RecType and SetType are assumed to be mutually disjoint, and disjoint from the concept representing classes AbstractClass. These constraints can be expressed by the following f-ALCIQ(D) axioms:

\[\text{RecType} \sqcap \text{SetType} \sqsubseteq \bot\quad \text{// this axiom asserts that two concepts RecType and SetType are disjoint, i.e., the fuzzy OWL axiom of the form DisjointClasses(RecType SetType)}\]

\[\text{SetType} \sqcap \text{AbstractClass} \sqsubseteq \bot\]

\[\text{RecType} \sqcap \text{AbstractClass} \sqsubseteq \bot\]

- The functional role value models the association between a fuzzy class FC and its type expression FT (see Definition 6). The role member is used to specify the type of elements of a set. These constraints can be expressed by the following f-ALCIQ(D) axioms:

\[\top \sqsubseteq 1\text{value}\quad \text{// this axiom asserts the intuitive definition Func(value), i.e., defines that the role value is a functional role.}\]

\[\exists\text{value} . \top \sqsubseteq \text{AbstractClass}\quad \text{// this axiom asserts the intuitive definition domain(value)=AbstractClass with a fuzzy DL f-ALCIQ(D) axiom form, i.e., defines that the domain of role value is AbstractClass.}\]

\[\exists\text{member} . \top \sqsubseteq \text{SetType}\quad \text{// this axiom asserts that a fuzzy class has and only has one type expression.}\]

\[\text{AbstractClass} \sqsubseteq= 1\text{value}\quad \text{// this axiom asserts the intuitive definition domain(member)=SetType with a fuzzy DL f-ALCIQ(D) axiom form, i.e., defines that the domain of role member is SetType.}\]

All of these f-ALCIQ(D) axioms above will be part of the f-ALCIQ(D) knowledge base we are going to define in Definition 9.

Definition 9 (Schema translation). Let \(FS = (FC_{FS}, FA_{FS}, FD_{FS})\) be a FOOD model. The f-ALCIQ(D) TBox \(\varphi(FS) = (FA, FP, FT)\) can be derived by a function \(\varphi\) as shown in Table 1.

Below we prove the correctness of the translation \(\varphi\) in Definition 9 based on [2,3,20,47,67], which can be sanctioned by establishing mappings between fuzzy database states (w.r.t. the FOOD model \(FS\)) and models of the translated f-ALCIQ(D) TBox \(\varphi(FS)\) (see Theorem 1).

Theorem 1. For every FOOD model \(FS = (FC_{FS}, FA_{FS}, FD_{FS})\), let \(FJ\) be a fuzzy database state w.r.t. \(FS\), and \(\varphi(FS)\) be the f-ALCIQ(D) TBox derived from \(FS\) by Definition 9. There are mappings: \(\alpha_{FS}\) is a mapping from \(FJ\) to a fuzzy interpretation of \(\varphi(FS)\), and \(\alpha_{FV}\) is a mapping from values of \(FJ\) to domain elements of the fuzzy interpretation of \(\varphi(FS)\); conversely, \(\beta_{FS}\) is a mapping from a fuzzy interpretation of \(\varphi(FS)\) to \(FJ\), and \(\beta_{FV}\) is a mapping from domain elements of the fuzzy interpretation of \(\varphi(FS)\) to values of \(FJ\), such that:

- For each legal fuzzy database state \(FJ\) for \(FS\), there is \(\alpha_{FS}(FJ)\) which is a model of \(\varphi(FS)\), and for each type expression \(FT\) of \(FS\) and each \(v \in FV_{FJ}\), \(v \in FT_{FJ}\) if and only if \(\alpha_{FV}(v) \in (\varphi(FT))_{\alpha_{FS}(FJ)}\);
- For each model \(FJ\) of \(\varphi(FS)\), there is \(\beta_{FS}(FJ)\) which is a legal fuzzy database state for \(FS\), and for each concept expression \(\varphi(FT)\) and each \(d \in A_{FJ}\), \(d \in (\varphi(FT))_{FJ}\) if and only if \(\beta_{FV}(d) \in FT_{\beta_{FS}(FJ)}\).

Proof. The following gives the proof of the first part of Theorem 1. Here, since Definition 9 aims at translating a FOOD model to an f-ALCIQ(D) TBox at terminological level, the following proof takes no account of the membership degrees which occur at three levels of fuzziness in a FOOD model (see Section 2.1). Firstly, let us define a fuzzy interpretation
Table 1
Translating rules from a FOOD model to an f-ALCQ(D) TBox.

<table>
<thead>
<tr>
<th>FOOD model</th>
<th>fALCQ(D) TBox</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( FS = (FC_{FS}, FA_{FS}, FD_{FS}) )</td>
<td>( \phi(FS) = (FA, FP, FT) )</td>
<td></td>
</tr>
<tr>
<td><strong>Fuzzy class FC(<em>{FS} ), fuzzy attribute FA(</em>{FS} )</strong></td>
<td>Atomic fuzzy concept set ( FA )</td>
<td>An atomic fuzzy concept ( \phi(FC) )</td>
</tr>
<tr>
<td>Each fuzzy class ( FC )</td>
<td>A fuzzy domain predicate ( \phi(R) )</td>
<td></td>
</tr>
<tr>
<td>Each result type ( R ) in method ( f() : R ) (the parameter of the method is null)</td>
<td></td>
<td>( m + 1 ) atomic fuzzy domain predicates ( \phi(P_1), \ldots, \phi(P_m), \phi(R) ), denoting the types of ( m ) parameters and a result, respectively</td>
</tr>
<tr>
<td>Each method ( f(P_1, \ldots, P_m) : R )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Each domain ( FB ) of fuzzy attributes ( FA )</td>
<td>A fuzzy domain predicate ( \phi(FB) )</td>
<td>Additional atomic fuzzy concepts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{AbstractClass, RecType, and SetType} )</td>
</tr>
<tr>
<td><strong>Fuzzy class declaration FD(_{FS} )</strong></td>
<td>Fuzzy axioms set ( FT )</td>
<td>The detailed introduction about the first seven axioms can be found before Table 1</td>
</tr>
<tr>
<td>Each class declaration in ( FD_{FS} ): Class FC is a ( FC_1, \ldots, FC_n ) type-is FT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( FT \rightarrow FC \mid FB )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Union ( FT_1, \ldots, FT_k ) ( (\text{disjointness, covering}) ) End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set-of ( FT), ( [(m_1, n_1), (m_2, n_2)] )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record ( FA_1 : FT_1, \ldots, FA_k : FT_k, u : \text{Real} ) End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( f(P_1, \ldots, P_n) : R )</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Each type expression ( FT )</strong></td>
<td>A fuzzy concept expression ( \phi(FT) )</td>
<td>The semantics of “disjoint concepts”, which can be found in Section 2.2, is given based on [59]</td>
</tr>
<tr>
<td>Each fuzzy class ( FC )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Each expression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Union ( FT_1, \ldots, FT_k ) ( (\text{disjointness, covering}) ) End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Each expression ( FA ) Set-of ( FT) ( [(m_1, n_1), (m_2, n_2)] )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: The table continues with similar entries for other axioms and rules.*
Each record type expression may include the following three cases:

**Record**

\[
\begin{align*}
&F_A : FT_1, \ldots, F_A : FT_k, u : \text{Real} \\
&f(j) : R \\
&f(P_1, \ldots, P_n) : R \\
\text{End}
\end{align*}
\]

The atomic fuzzy relations of set and it shows that

\[
\begin{align*}
\text{FS} & = (FC_{FS}, FA_{FS}, FD_{FS}) \\
\text{f-ALCIQ(D)} & \text{ TBox } \varphi(\text{FS}) = (FA, FP, FT) \\
\end{align*}
\]

\[
\begin{align*}
\text{Comments} & \text{ In the concept expression (iii), the part}_1 \\
&\text{states that each instance of } \varphi(FC)\text{, representing a tuple, correctly is connected to} \\
&\text{exactly one object for each of the roles }\varphi(R_{f_1}(P)), \varphi(R_1), \ldots, \varphi(R_m)\text{; the other two} \\
&\text{parts impose the correct typing of the parameters and of the return value.}
\end{align*}
\]

\(z_{FS}(FJ)\) of \(\varphi(\text{FS})\) as follows:

- As mentioned above, \(z_{FP}\) is a mapping from values \(FV_{FJ}\) of the fuzzy database state \(FJ\) w.r.t. \(\text{FS}\) to domain elements \(A^{z_{FS}(FJ)}\) of the fuzzy interpretation of \(\varphi(\text{FS})\), i.e., \(A^{z_{FS}(FJ)}\) is constituted by the set of elements \(z_{FP}(v)\) for each \(v \in FV_{FJ}\). Since each fuzzy object of \(\text{FS}\) is assigned a structured value, in order to explicitly represent in \(\text{f-ALCIQ(D)}\) the type structure of fuzzy classes, we denote with \(\text{FC}\) corresponding to fuzzy object identifiers, fuzzy record values, and fuzzy set values, respectively.
- The atomic fuzzy concepts of set \(FA\) in Definition 9 are interpreted as follows:

\[
\begin{align*}
\text{AbstractClass}^{z_{FS}(FJ)} & = \Delta_{fid}, \\
\text{RecType}^{z_{FS}(FJ)} & = \Delta_{free}, \\
\text{SetType}^{z_{FS}(FJ)} & = \Delta_{fset}; \\
(\varphi(FC))^{z_{FS}(FJ)} & = \{z_{FP}(v) | v \in \pi_{FJ}(FC), FC \in FC_{FS}\}.
\end{align*}
\]

- The atomic fuzzy relations of set \(FP\) in Definition 9 are interpreted as follows:

\[
\begin{align*}
\text{member}^{z_{FS}(FJ)} & = \{(d_1, d_2)|d_1 \in \Delta_{fset} \land z_{FP}^{-1}(d_1) = \{\ldots, z_{FP}^{-1}(d_2)\}, \ldots\}; \\
\text{value}^{z_{FS}(FJ)} & = \{(d_1, d_2)|(z_{FP}^{-1}(d_1), z_{FP}^{-1}(d_2)) \in \pi_{FJ}\}; \\
(\varphi(FA))^{z_{FS}(FJ)} & = \{(d_1, d_2)|d_1 \in \Delta_{free} \land z_{FP}^{-1}(d_1) = \{\ldots, FA : z_{FP}^{-1}(d_2)\}, \ldots\}, \text{where } FA \in FA_{FS}, z_{FP}^{-1}(d_1) \text{ is the} \\
&\text{inverse function (mapping) of } z_{FP}(d_1).
\end{align*}
\]

Now, based on the definition of \(z_{FS}(FJ)\) above, we prove the first part of Theorem 1 by considering each case of type expressions \(FT\) in Definition 9.

**Case 1:** \(FT \rightarrow FC\). By Definition 9, FC is mapped into \(\varphi(FC)\).

**Proof”\(\Rightarrow\)”**: If \(v \in FT_{FI} = FC_{FI}\), by Definition 7, we have \(v \in \pi_{FI}(FC)\). Then, according to the definition of \(z_{FS}(FJ)\) above it follows \(z_{FP}(v) \in (\varphi(FC))^{z_{FS}(FJ)}\), and vice versa.

**Case 2:** \(FT \rightarrow \) Set-of \(FT’\), for simplicity, the cardinality constraint of a set is omitted. By Definition 9, FT is mapped into SetType \(\cap\text{member}(\varphi(FT’))\). We first give an induction assumption that \(v \in (FT’)_{FI}\) iff \(z_{FP}(v) \in (\varphi(FT’))^{z_{FS}(FJ)}\), and it shows that \(v \in (FT’)_{FI}\) iff \(z_{FP}(v) \in (\varphi(FT’))^{z_{FS}(FJ)}\).

**Proof”\(\Rightarrow\)”**: If \(v \in (FT’)_{FI}\), according to Definition 7, we have \(v = \{FV_1, \ldots, FV_k’\}\) and \(FV_i’ \in (FT’)_{FI}\) for \(i \in \{1, \ldots, k\}\). Then, by the induction assumption above it follows \(z_{FP}(FV_i’) \in (\varphi(FT’))^{z_{FS}(FJ)}\) for \(i \in \{1, \ldots, k\}\).
Furthermore, by the definition of \( x_{FS}(FJ) \) above, we have \( x_{FS}(v) \in SetType^{x_{FS}(FJ)} \) and \( (x_{FS}(v), x_{FS}(V_i^j)) \in member^{x_{FS}(FJ)}, i \in \{1, \ldots, k\}. \) Therefore, \( x_{FS}(v) \in (\phi(FT))^\{x_{FS}(FJ)\} \).

Proof “\( \leftarrow \)”: If \( d = x_{FS}(v) \in (\phi(FT))^\{x_{FS}(FJ)\} \), according to \( \phi(FT) = SetType \sqcap \forall member. \phi(FT') \), there is exactly one \( d_i \in A^{x_{FS}(FJ)} \) with \( (d, d_i) \in member^{x_{FS}(FJ)} \) and \( d_i \in (\phi(FT'))^\{x_{FS}(FJ)\}, i \in \{1, \ldots, k\}. \) Then, by the definition of \( x_{FS}(FJ) \) above, we have \( v = \{V'_1, \ldots, V'_k\} \) and \( V'_i = x_{FS}^{-1}(d_i) \) for \( i \in \{1, \ldots, k\}. \) Furthermore, by the induction assumption above it follows \( FV'_i \in (FT')^\{FJ\} \) for \( i \in \{1, \ldots, k\}. \) Therefore, \( v \in FT^\{FJ\} = (\text{Set-of } FT^\{FJ\}). \)

Case 3: \( FT \rightarrow \text{Record } FA_1: FT_1, \ldots, FA_k: FT_k \) End, note that \( u \in FA_1 \) and the method is included in the record type. By Definition 9, \( FT \) is mapped into \( \text{RecType} \sqcap \forall \phi(FA_1).\phi(FT_1) \sqcap 1 = \phi(FA_1) \sqcap \cdots \sqcap \forall \phi(FA_k).\phi(FT_k) \sqcap 1 = 1\phi(FA_k). \) We first give an induction assumption that \( v \in FT^\{FJ\} \) iff \( x_{FS}(v) \in (\phi(FT))^\{x_{FS}(FJ)\} \), for \( i \in \{1, \ldots, k\} \), and it shows that \( v \in FT^\{FJ\} \) iff \( x_{FS}(v) \in (\phi(FT))^\{x_{FS}(FJ)\} \).

Proof “\( \Rightarrow \)”: If \( v \in FT^\{FJ\} \), by Definition 7 it follows \( v = \{FA_1: FV_1, \ldots, FA_k: FV_k\} \) and \( FV_i \in FT^\{FJ\} \) for \( i \in \{1, \ldots, k\}. \) Furthermore, according to the induction assumption above, we have \( x_{FS}(FV_i) \in (\phi(FT_i))^\{x_{FS}(FJ)\} \) for \( i \in \{1, \ldots, k\}. \) Then, by the definition of \( x_{FS}(FJ) \) above we have \( x_{FS}(v) \in \text{RecType}^\{x_{FS}(FJ)\}, (x_{FS}(v), x_{FS}(FV_i)) \in (\phi(FA_i))^\{x_{FS}(FJ)\} \) for \( i \in \{1, \ldots, k\} \), and all fuzzy relations \( \phi(FA) \) corresponding to fuzzy attribute names FA are functional. Therefore, \( x_{FS}(v) \in (\phi(FT))^\{x_{FS}(FJ)\} \).

Proof “\( \leftarrow \)”: If \( d = x_{FS}(v) \in (\phi(FT))^\{x_{FS}(FJ)\} \), according to \( \phi(FT) = \text{RecType} \sqcap \forall \phi(FA_1).\phi(FT_1) \sqcap 1 = \phi(FA_1) \sqcap \cdots \sqcap \forall \phi(FA_k).\phi(FT_k) \sqcap 1 = 1\phi(FA_k). \) then there is exactly one \( d_i \in A^{x_{FS}(FJ)} \) with \( (d, d_i) \in (\phi(FT_i))^\{x_{FS}(FJ)\} \) and \( d_i \in (\phi(FT_i))^\{x_{FS}(FJ)\}, i \in \{1, \ldots, k\}. \) Furthermore, by the definition of \( x_{FS}(FJ) \) above, we have \( v = [FA_1: FV_1, \ldots, FA_k: FV_k] \) with \( FV_i = x_{FS}^{-1}(d_i) \) for \( i \in \{1, \ldots, k\}. \) Then, by the induction assumption above, we have \( FV_i \in FT^\{FJ\} \) for \( i \in \{1, \ldots, k\}. \) Therefore, \( v \in FT^\{FJ\} = (\text{Record } FA_1: FT_1, \ldots, FA_k: FT_k \text{ End})^\{FJ\}. \)

Case 4: The case for \( FT \rightarrow \text{Union } FT_1, \ldots, FT_k \) (disjointness, covering) End can be treated analogously.

The two parts of Theorem 1 are a mutually inverse process. The proof of the second part of Theorem 1, which can be treated analogously according to the first part above, is omitted here. \( \square \)

In the following, we further illustrate the translation procedure in Definition 9 with Example 1.

**Example 1.** By applying Definition 9 to the FOOD model \( FS_1 \) in Fig. 3, we can obtain the corresponding \( f-ALCIQ(D) \) TBox \( \phi(FS_1) = (FA, FP, FT) \) in Fig. 5.

4.2. Translating database instance into \( f-ALCIQ(D) \) ABox

In this section, we investigate how to translate a fuzzy database instance (i.e., a set of object instances) w.r.t. the FOOD model into an \( f-ALCIQ(D) \) ABox. First, let us briefly introduce the representation forms of instances in \( f-ALCIQ(D) \) and FOOD models.

In \( f-ALCIQ(D) \), one specifies a descriptive first of affairs of an application domain in terms of assertional. The assertions used in an \( f-ALCIQ(D) \) ABox contain concept assertions \( (C(a) \approx d_k) \), role assertions \( (R(a,b) \approx d_k) \) and \( (T(a,v) \approx d_k) \), and individual (in)equality assertions \( a \approx b \) and \( a \not\approx b \), where \( a, b \in A^\{FI\} \), \( v \in A_D, \therefore \approx \in \{\geq, >, \leq, <\}, k \in [0, 1] \) (see Definition 4).

A fuzzy object-oriented database, which describes the real world by means of objects, values, and their mutual relationships, can be considered as a finite set of assertions \([14,36,47,68]\). The assertion formalisms of a fuzzy database instance (i.e., a set of object instances) with respect to a FOOD model include:

(1) The assertion of the form \( FO: FC; u \), which denotes that a fuzzy object \( FO \) is an instance of a fuzzy class \( FC \) with membership degree of \( u \in [0, 1] \).

(2) The assertion of the form \( FO: [FA_1: FV_1: n_1, \ldots, FA_k: FV_k: n_k] \), which denotes the fuzzy structured value associated with \( FO \), where \( FV_i \in FV^E \) (see Definition 7), \( FA_i \) denotes the attribute of \( FO, n_i \in [0, 1] \) denotes the membership degree, and \( i \in \{1 \ldots k\} \). Here, since the value of an attribute \( FA_i \) may be a possibility distribution, for simplicity, \( FV_i : n_i \) only denotes one element of the possibility distribution.

(3) Since the fuzzy subclass/superclass relationship in a FOOD model can be assessed by utilizing the inclusion degree of objects to the class \([26,30,35]\), the assertion formalism of fuzzy subclass/superclass relationship can be re-expressed by the above assertion formalism of objects to the class.

On this basis, Definition 10 gives a translation approach from fuzzy database instances to \( f-ALCIQ(D) \) ABoxes.
The \( f\text{-}ALCIQ(D) \) TBox \( \varphi(FS_1) \) derived from the FOOD model \( FS_1 \) in Fig. 3: \( FA = \{ \text{AbstractClass, RecType, SetType, Chief-Leader, Leader, Old-Employee, Young-employee, Employee, String, Integer, Real} \} \)
\( FP = \{ \text{value, member, u, Name, Age, Salary, IsDepartment, Number, Manage} \} \)
\( FT = \{ \text{SetType} \ni \text{AbstractClass} \perp \)
\( \text{RecType} \ni \text{AbstractClass} \perp \)
\( \text{RecType} \ni \text{SetType} \ni \perp \)
\( \top \ni \leq 1 \text{value} \)
\( \exists \text{value}. \top \ni \text{AbstractClass} \)
\( \text{AbstractClass} \ni \text{= 1 value} \)
\( \exists \text{member}. \top \ni \text{SetType} \)
\( \text{Employee} \ni \text{AbstractClass} \ni \forall \text{value}.(\text{Young-Employee} u \text{Old-Employee}) \)
\( \text{Young-Employee} \ni \text{Old-Employee} \ni \perp \)
\( \text{Young-Employee} \ni \text{Employee} \ni \perp \)
\( \text{Young-Employee} \ni \text{Old-Employee} \ni \perp \)
\( \text{Young-Employee} \ni \text{Employee} \ni \perp \)
\( \text{Chief-Leader} \ni \text{AbstractClass} \ni \text{Leader} \ni \forall \text{value}.(\text{RecType} \ni \forall \text{Number}. \text{String} \ni =1 \text{Name} u \ni \text{Age}. \text{Real} \ni =1 \text{Age} u \ni \text{Salary}. \text{Integer} \ni =1 \text{Salary} u \ni =1 \text{Manager}. \text{Chief-Leader} \ni \forall \text{IsDepartment}. \text{String} \ni \leq 1 \text{IsDepartment}. \top) \)
\( \text{Chief-Leader} \ni \text{AbstractClass} \ni \text{Leader} \ni \forall \text{value}.(\text{RecType} \ni \forall \text{Number}. \text{String} \ni =1 \text{Name} u \ni \text{Age}. \text{Real} \ni =1 \text{Age} u \ni \text{Manage}. (\text{SetType} \ni \forall \text{member}. \text{Young-Employee} \ni \geq 2 \text{member} \ni \leq 6 \text{member}) ) \)

Fig. 5. The \( f\text{-}ALCIQ(D) \) TBox \( \varphi(FS_1) \) derived from the FOOD model \( FS_1 \) in Fig. 3.

**Definition 10** (Instance translation). Given a fuzzy database instance (i.e., a set of object instances) w.r.t. a FOOD model \( FS \), the corresponding \( f\text{-}ALCIQ(D) \) ABox (w.r.t. the \( f\text{-}ALCIQ(D) \) TBox \( \varphi(FS) \) in Definition 9) can be derived as the following rules:

1. Each fuzzy object symbol \( FO \) is mapped into an \( f\text{-}ALCIQ(D) \) abstract individual \( \varphi(FO) \).
2. Each fuzzy class symbol \( FC \in FC_{FS} \) and each fuzzy attribute symbol \( FA \in FA_{FS} \) are mapped into a fuzzy concept \( \varphi(FC) \) and a fuzzy role \( \varphi(FA) \), respectively. Here, to improve readability, \( \varphi(FC) \) stands for all the fuzzy concepts in Table 1, and \( \varphi(FA) \) stands for all the fuzzy abstract and concrete roles in Table 1 (see Table 1 in detail).
3. Each fuzzy assertion \( FO: FC: u \in FS \) is mapped into an \( f\text{-}ALCIQ(D) \) assertion \( \langle \varphi(FC)(\varphi(FO)) \ni \leq u \rangle \), where \( \ni \leq \in \{ \geq, \leq \} \).
4. Each fuzzy assertion \( FO : [FA_1 : FV_1 : n_1, \ldots, FA_k : FV_k : n_k] \) is mapped into the \( f\text{-}ALCIQ(D) \) assertions \( \langle \varphi(FA_1)(\varphi(FO), FV_j) \ni \leq n_j \rangle \), where \( FV_j \in FV_{FS} \) (see Definition 7), which is mapped into an abstract individual or a concrete individual, \( i \in \{ 1, \ldots, k \} \), \( \ni \leq \in \{ \geq, \leq \} \).

In the following, we further illustrate the translation procedure in Definition 10 with Example 2.

**Example 2.** Given the fuzzy database instance in Fig. 4 (w.r.t. the FOOD model \( FS_1 \) in Fig. 3), Fig. 6 shows the translated \( f\text{-}ALCIQ(D) \) ABox (w.r.t. the \( f\text{-}ALCIQ(D) \) TBox \( \varphi(FS_1) \) in Fig. 5). Here, when the membership degree \( u \) or \( n_i \) in Definition 10 is equal to 1.0, the [\( \ni \leq u \)] or [\( \ni \leq n_i \)] part is omitted in a fuzzy assertion.

### 4.3. Prototype translation tool

Following the proposed approaches in Sections 4.1 and 4.2, we developed a prototype translation tool called \( FOOD2DL \), which can automatically extract FOOD models and object instances stored in databases and translate them into \( f\text{-}ALCIQ(D) \) knowledge bases.
Given the fuzzy database instance in Fig. 4, the corresponding f-ALCIQ(D) ABox is derived as follows:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young-Employee</td>
<td>Salary</td>
<td>Salary (O2, 6000)</td>
<td>0.8</td>
</tr>
<tr>
<td>Young-Employee</td>
<td>Name</td>
<td>Name (O2, Chris)</td>
<td>0.95</td>
</tr>
<tr>
<td>Age (O2, 20)</td>
<td>Chief-Leader</td>
<td>Chief-Leader (O2)</td>
<td>0.85</td>
</tr>
<tr>
<td>Age (O2, 20)</td>
<td>Number</td>
<td>Number (O2, 03)</td>
<td>0.7</td>
</tr>
<tr>
<td>Salary (O2, 3000)</td>
<td>Manage</td>
<td>Manage (O2, O1)</td>
<td>0.7</td>
</tr>
<tr>
<td>IsDepartment</td>
<td>Old-Employee</td>
<td>Old-Employee (O2)</td>
<td>0.6</td>
</tr>
<tr>
<td>Young-Employee</td>
<td>Employee</td>
<td>Employee (O2)</td>
<td>0.9</td>
</tr>
<tr>
<td>Name (O2, John)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age (O2, Young)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. The f-ALCIQ(D) ABox derived from the fuzzy database instance in Fig. 4.

The fuzzy object-oriented databases (including the schema information FOOD models and the object instances) are stored in the object-oriented database DB4O [24] by using the techniques presented in [10,30,43,45]. DB4O, which is an open source object database developed in Java, is extensively used. DB4O enables Java and .NET developers to store and retrieve any application object with only one line of code, eliminating the need to predefine or maintain a separate, rigid data model. In the following, we briefly introduce the implementation of FOOD2DL.

The implementation of FOOD2DL is based on Java 2 JDK 1.5 platform. FOOD2DL includes three main modules: connecting and parsing module, translation module, and output module. The connecting and parsing module uses the regular expression to parse the FOOD model file and stores the parsed results as Java ArrayList classes; translation module translates the FOOD model and data instances into the f-ALCIQ(D) knowledge bases based on the proposed approaches in Sections 4.1 and 4.2; output module produces the resulting f-ALCIQ(D) knowledge base which is saved as a text file and displayed on the tool screen.

We have carried out lots of case studies using FOOD2DL. Here we give an example to show that the proposed approach is feasible and the implemented tool is efficient. Fig. 7 shows the screen snapshot of FOOD2DL, which displays the
translations from the FOOD model (Fig. 3) and the corresponding database instance (Fig. 4) to the \( f\text{-ALCIQ}(D) \) TBox (Fig. 5) and the \( f\text{-ALCIQ}(D) \) ABox (Fig. 6).

So far, on the basis of the proposed approach and the implemented tool, FOOD models can be represented by \( f\text{-ALCIQ}(D) \) knowledge bases. Therefore, it is possible for us to take advantage of the associated reasoning techniques of \( f\text{-ALCIQ}(D) \) to reason on the FOOD models. Based on the translated \( f\text{-ALCIQ}(D) \) knowledge bases from FOOD models, the following section will investigate how to reason on FOOD models with \( f\text{-ALCIQ}(D) \).

5. Reasoning on FOOD models with \( f\text{-ALCIQ}(D) \)

This section investigates how to reason on FOOD models based on the translated \( f\text{-ALCIQ}(D) \) knowledge bases, including: (i) we introduce several familiar reasoning tasks considered in FOOD models and give their formal definitions; (ii) we investigate how to reason on FOOD models by \( f\text{-ALCIQ}(D) \), i.e., how to reduce reasoning on FOOD models to reasoning on \( f\text{-ALCIQ}(D) \) knowledge bases. Further, reasoning on \( f\text{-ALCIQ}(D) \) knowledge bases can be automatically done by means of the existing fuzzy reasoner DeLorean [8], which supports \( f\text{-ALCIQ}(D) \) with general concept inclusions (GCIs).

5.1. Reasoning problems of FOOD models

In the following we first give a brief example, which is helpful to understand the reasoning tasks of FOOD models and can show that it is possible and meaningful to represent and reason on FOOD models with fuzzy DLs. Then, we give formal definitions of reasoning problems of FOOD models.

Example 3. We take the FOOD model in Fig. 1 and its corresponding object instances in Fig. 2 for example to introduce the reasoning tasks of FOOD models. From Figs. 1 and 2, there are unsatisfiability and redundancy in the FOOD model since the following reasons:

(a) Since \( Young\text{-Employee} \) is a subclass of \( Old\text{-Employee} \), i.e., for any object, the membership degree that it belongs to the subclass \( Young\text{-Employee} \) is less than or equal to the membership degree that it belongs to the superclass \( Old\text{-Employee} \), we have \( \{o_1/ \geq 0.9, o_2/ \geq 0.95, o_1'/0.6, o_2'/0.8\} \in Old\text{-Employee} \).

(b) Since \( Young\text{-Employee} \) and \( Old\text{-Employee} \) are disjoint, i.e., there is no instance which belongs to the two classes, we have \( \{o_1/0, o_2/0\} \in Old\text{-Employee} \).

(c) From (a) and (b), it is shown that the two instance sets of \( Old\text{-Employee} \) are conflictive. Therefore, we know that the instance set of \( Young\text{-Employee} \) is an empty set, because the empty set is the only set that can be at the same time disjoint from and contained in the class \( Old\text{-Employee} \). That is, \( Young\text{-Employee} \) is unsatisfiable since it is an empty class.

(d) The class \( Young\text{-Employee} \) is an empty class, and \( Employee \) is the union of \( Young\text{-Employee} \) and \( Old\text{-Employee} \). Therefore, \( Employee \) is equivalent to \( Old\text{-Employee} \), i.e., there is redundancy in the FOOD model.

(e) The above cases may result in that other undesirable problems occur in the complete FOOD model in Fig. 3.

All the reasoning problems mentioned in Example 3 may occur in the FOOD modeling activities, and the burden of checking these problems is left to the designers. Therefore, it would be highly desirable to improve the ability of reasoning on database models. Based on the previous translation work, in the following we make an attempt to resolve the problem by means of the fuzzy DL \( f\text{-ALCIQ}(D) \).

Similar to the classical/fuzzy database models (such as ER models [20,25,67], UML models [9,38], and object-oriented database models [3,20,47]), the familiar reasoning problems considered in FOOD models include consistency, satisfiability, subsumption, and redundancy. The following gives their formal definitions.

Definition 11 (Consistency of fuzzy object-oriented databases). A fuzzy object-oriented database contains its schema information (i.e., a FOOD model) and its instance information (i.e., a set of object instances). A fuzzy object-oriented database is consistent, if the set of object instances satisfies all the constraints of the FOOD model.

In particular, when no instance exists in the fuzzy object-oriented database, the consistency problem of the database above is reduced to the satisfiability problem of the FOOD model in Definition 12.
Definition 12 (Satisfiability of FOOD models). A FOOD model is satisfiable, if it admits at least one fuzzy database instance (i.e., a set of object instances).

If a FOOD model is not satisfiable, the classes altogether are contradictory, i.e., it does not allow that any class can be populated without violating any of the requirements imposed by the FOOD model. This may be due to a design error or over-constraining.

Definition 13 (Satisfiability of fuzzy classes). A fuzzy class $FC$ in a FOOD model $FS$ is satisfiable, if there is at least one fuzzy database state/instance $FJ$ of $FS$ such that $FC^{FJ} \neq \emptyset$ (see Definition 7), i.e., $FS$ admits a fuzzy database instance in which $FC$ has a non-empty set of objects.

An unsatisfiable fuzzy class weakens the understandability of a FOOD model, since the fuzzy class stands for an empty class, and thus, at the very least, it is inappropriately named. In this case, designers should modify or delete the fuzzy class to increase understandability.

Definition 14 (Subsumption of fuzzy classes). Let $FS$ be a FOOD model, and $FC_1$, $FC_2$ be two fuzzy classes in $FS$. For each fuzzy database state/instance $FJ$ of $FS$, if $FC_1^{FJ}$ is a subset of $FC_2^{FJ}$ (see Definition 7), then $FC_1$ is a subclass of $FC_2$.

In short, a fuzzy class $FC_1$ is a subclass of another fuzzy class $FC_2$ if, for any object, the membership degree of it to $FC_1$ is less than or equal to the membership degree of it to $FC_2$. Class subsumption is the basis for a classification of all the classes in a FOOD model.

As mentioned in Example 3, a FOOD model is redundant if there is an empty class or two equivalent fuzzy classes. Therefore, before giving the definition of redundancy of FOOD models, the concept of equivalence of fuzzy classes should be introduced first.

As we have known, in the classical object-oriented database, two classes are equivalent if they denote the same set of object instances. In a FOOD model, however, an object may belong to a class with membership degree of $[0, 1]$, and thus two fuzzy classes may have same objects with same/different membership degrees. In this case, the definition of equivalent classes should be extended under fuzzy environment as will be shown in Definition 15.

Definition 15 (Equivalence of fuzzy classes). A fuzzy class $FC_1$ is equivalent to another fuzzy class $FC_2$, if at least one of the following conditions is satisfied: (1) two classes have same objects with same membership degrees; (2) two classes do not have subclass/superclass relationship but they have same objects with different membership degrees.

Note that, when two fuzzy classes have same objects with same membership degrees, the two fuzzy classes are strictly equivalent. In this case, one of the fuzzy classes can be removed and replaced by another. Now let us focus on how to handle the case that two classes have same objects with different membership degrees. Let $FC_1$ and $FC_2$ be two fuzzy classes, and assume that there is an object $o$ belonging to the two fuzzy classes with different membership degrees $u_{FC_1}(o) \in (0, 1]$ and $u_{FC_2}(o) \in (0, 1]$. At this moment, which one in $FC_1$ and $FC_2$ is the class of object $o$ depends on the following case: if $u_{FC_1}(o) > u_{FC_2}(o)$, then $FC_1$ is considered as the class of object $o$ and we say $FC_1$ fuzzily include $FC_2$, else $FC_2$ is considered as the class of object $o$ and we say $FC_2$ fuzzily include $FC_1$. Correspondingly, the class which is included by another class is removed, and the class which includes the removed class is retained.

Based on Definition 15, the following gives the formal definition of redundancy of FOOD models.

Definition 16 (Redundancy of FOOD models). A FOOD model is redundant, if there is some class standing for an empty class, i.e., it is inappropriately named; or there are two equivalent fuzzy classes.

Removing redundancy can reduce the complexity of a FOOD model. The reasoning problems mentioned above may occur in the FOOD modeling activities, the following section will investigate how to reason on these problems of FOOD models by means of the fuzzy $DL_f$-ALC$\mathcal{IQ}(D)$.
5.2. Reasoning on FOOD models with f-ALCIQ(D)

The following theorems allow us to reduce reasoning on FOOD models to reasoning on f-ALCIQ(D) knowledge bases, so that the reasoning problems of FOOD models may be checked through the reasoning mechanism of f-ALCIQ(D).

**Theorem 2** (Consistency of fuzzy object-oriented databases). Given a fuzzy object-oriented database (including a FOOD model and a fuzzy database instance), and \( \Sigma = (\text{TBox}, \text{ABox}) \) is the translated f-ALCIQ(D) KB according to the proposed approach in Section 4. The fuzzy object-oriented database is consistent iff \( \Sigma \) is satisfiable, i.e., the ABox is consistent w.r.t. the TBox.

**Theorem 3** (Satisfiability of FOOD models). Given a FOOD model FS, and \( \varphi(FS) \) is the translated f-ALCIQ(D) TBox. FS is satisfiable iff \( \varphi(FS) \) is satisfiable, i.e., there is a fuzzy interpretation FI which is a model of \( \varphi(FS) \).

Note that, when no instance exists in a fuzzy object-oriented database (i.e., the translated f-ALCIQ(D) KB contains only the TBox), the consistency problem of the fuzzy object-oriented database (Theorem 2) is reduced to the satisfiability problem of the FOOD model (Theorem 3). Moreover, Theorems 2 and 3 are the straightforward consequences of Definitions 9–12, and thus their proofs are omitted here.

**Theorem 4** (Satisfiability of fuzzy classes). Given a FOOD model FS, \( \varphi(FS) \) is the translated f-ALCIQ(D) TBox, and FC is a fuzzy class in FS. FC is satisfiable iff: \( \varphi(FS) \not\models \varphi(FC) \subseteq \bot \).

**Proof.** “\( \Rightarrow \)” : If FC is consistent, then there is a legal fuzzy database state FJ with \( FC^FJ \not\equiv \emptyset \), i.e., \( \exists v. v \in FC^FJ \) and \( v \in FC^C \). By part 1 of Theorem 1, \( \varphi_{FS}(FJ) \) is a model of \( \varphi(FS) \) with \( \varphi_{FS}(v) \in (\varphi(FC))^{FS(FJ)} \), i.e., \( (\varphi(FC))^{FS(FJ)} \not\equiv \emptyset \). That is, \( \varphi(FS) \not\models \varphi(FC) \subseteq \bot \).

“\( \Leftarrow \)” : If \( \varphi(FS) \not\models \varphi(FC) \subseteq \bot \), i.e., FC is consistent in \( \varphi(FS) \), then there is a fuzzy interpretation FI of \( \varphi(FS) \) with \( (\varphi(FC))^FI \not\equiv \emptyset \), i.e., \( \exists d. d \in A^F \) and \( d \in (\varphi(FC))^FI \). By part 2 of Theorem 1, \( \beta_{FS}(FI) \) is a legal fuzzy database state for FS with \( \beta_{FS}(d) \in FC_{\beta_{FS}(FI)} \), i.e., \( FC_{\beta_{FS}(FI)} \not\equiv \emptyset \). That is, FC is consistent.

**Theorem 5** (Subsumption of fuzzy classes). Given a FOOD model FS, \( \varphi(FS) \) is the translated f-ALCIQ(D) TBox, and FC1, FC2 are two fuzzy classes in FS. FC1 is a subclass of FC2 iff: \( \varphi(FS) \models \varphi(FC_1) \subseteq \varphi(FC_2) \).

**Proof.** “\( \Rightarrow \)” : Let \( \varphi(FS) \not\models \varphi(FC_1) \subseteq \varphi(FC_2) \), i.e., \( \varphi(FC_1) \cap \neg \varphi(FC_2) \) is consistent in \( \varphi(FS) \), then there is a fuzzy interpretation FI of \( \varphi(FS) \) with \( (\varphi(FC_1))^{FI} \not\equiv \emptyset \), i.e., \( \exists d. d \in A^F \) such that \( d \in (\varphi(FC_1))^FI \) and \( d \not\in (\varphi(FC_2))^FI \). By part 2 of Theorem 1, \( \beta_{FS}(FI) \) is a legal fuzzy database state for FS with \( \beta_{FS}(d) \in FC_{\beta_{FS}(FI)} \) and \( \beta_{FS}(d) \not\in FC_{\beta_{FS}(FI)} \), i.e., FC1 is not the subclass of FC2, and thus there is a contradiction. That is, \( \varphi(FS) \not\models \varphi(FC_1) \subseteq \varphi(FC_2) \).

“\( \Leftarrow \)” : If FC1 is not the subclass of FC2, i.e., there is a legal fuzzy database state FJ for FS and \( v \in FC_{\bot} \) and \( v \not\in FC^FJ \). By part 1 of Theorem 1, \( \varphi_{FS}(FJ) \) is a model of \( \varphi(FS) \) with \( \varphi_{FS}(v) \in (\varphi(FC_1))^{FS(FJ)} \) and \( \varphi_{FS}(v) \not\in (\varphi(FC_2))^{FS(FJ)} \), i.e., \( \varphi(FS) \not\models \varphi(FC_1) \subseteq \varphi(FC_2) \), and thus there is a contradiction. That is, FC1 is the subclass of FC2.

Before giving Theorem 6, we first introduce a notion of having the same individuals for two different f-ALCIQ(D)-concepts. As mentioned in Section 2.2.2, given an f-ALCIQ(D) knowledge base, a fuzzy concept \( \varphi(FC) \) and \( n' \) \in \( (0, 1] \), the retrieval reasoning service of f-ALCIQ(D) can return all the individuals \( d \), which satisfy the condition \( \varphi(FC)(d) \geq n' \). Therefore, given two f-ALCIQ(D)-concepts \( \varphi(FC_1) \) and \( \varphi(FC_2) \), two sets of individuals respectively belonging to \( \varphi(FC_1) \) and \( \varphi(FC_2) \), written as Retrieval(\( \varphi(FC_1) \), \( n \)) and Retrieval(\( \varphi(FC_2) \), \( n \)), can be created by the retrieval reasoning service above. Here, the degree of truth \( n \) \in \( (0, 1] \) may be a positive infinite small value or be specified by the designers in the practical FOOD modeling activities. Furthermore, we say that two concepts \( \varphi(FC_1) \) and \( \varphi(FC_2) \) have the same individuals if the following conditions hold:

(a) For every \( x \) in Retrieval(\( \varphi(FC_1) \), \( n \)), one of the following holds: (i) \( x \) in Retrieval(\( \varphi(FC_2) \), \( n \)), or (ii) exists \( y \) in Retrieval(\( \varphi(FC_2) \), \( n \)) such that \( x \approx y \).
(b) For every \( x \) in \( \text{Retrieval}(\varphi(FC_2), n) \), one of the following holds: (i) \( x \) in \( \text{Retrieval}(\varphi(FC_1), n) \), or (ii) exists \( y \) in \( \text{Retrieval}(\varphi(FC_1), n) \) such that \( x \approx y \).

Based on the discussion above, the following gives a theorem of checking the redundancy of a FOOD model.

**Theorem 6 (Redundancy).** Given a FOOD model \( FS, \varphi(FS) \) is the translated \( f\text{-ALCIQ}(D) \) TBox, and \( FC, FC_1, FC_2 \) are fuzzy classes in \( FS \). \( FS \) is redundant if and only if at least one of the following conditions is satisfied: (i) \( \varphi(FC_1)\equiv\perp \); (ii) \( \varphi(FC_1)\equiv\varphi(FC_2) \), i.e., \( \varphi(FC_1)\equiv\varphi(FC_2) \) and \( \varphi(FC_1)\equiv\varphi(FC_2) \); (iii) \( \varphi(FC_1)\not\equiv\varphi(FC_2) \), \( \varphi(FC_1)\not\equiv\varphi(FC_2) \) and \( \varphi(FC_1)\) and \( \varphi(FC_2) \) contain same individuals.

Note that, since two concepts are called equivalent in fuzzy DLs when they have same individuals with same membership degrees (see Definition 3), the equivalence problem of fuzzy classes in the case (1) of Definition 15 can be reasoned by checking whether \( \varphi(FS)\equiv\varphi(FC_1) \equiv \varphi(FC_2) \), i.e., \( \varphi(FS)\equiv\varphi(FC_1) \equiv \varphi(FC_2) \) and \( \varphi(FS)\equiv\varphi(FC_1) \equiv \varphi(FC_2) \). However, for the case (2) of Definition 15, i.e., when two classes do not have subclass/superclass relationship but they have same objects with different membership degrees, it cannot be handled directly by checking whether \( \varphi(FS)\equiv\varphi(FC_1) \equiv \varphi(FC_2) \). However, for the case (2) of Definition 15, i.e., when two classes do not have subclass/superclass relationship but they have same objects with different membership degrees, it cannot be handled directly by checking whether \( \varphi(FS)\equiv\varphi(FC_1) \equiv \varphi(FC_2) \) and \( \varphi(FS)\equiv\varphi(FC_1) \equiv \varphi(FC_2) \). Furthermore, whether two classes contain same objects can also be checked by detecting whether two concepts \( \varphi(FC_1) \) and \( \varphi(FC_2) \) have the same individuals as mentioned above. Moreover, Theorem 6 is the straightforward consequences of Definitions 15 and 16, and the proof is hereby omitted here.

In the following, Fig. 8 shows how the reasoning problems of FOOD models (such as unsatisfiability, redundancy, and other undesirable problems mentioned in Example 3 in Section 5.1) can be checked by reasoning on \( f\text{-ALCIQ}(D) \) knowledge bases. The reasoning on \( f\text{-ALCIQ}(D) \) knowledge bases can be done by means of the fuzzy reasoner DeLorean [8]. The reasoning results show that the fuzzy concept Chief-Leader is satisfiable, the fuzzy concept Young-Employee is unsatisfiable since it has an empty set of individuals, \( KB\equiv\text{Employee} \equiv \text{Old-Employee} \), and so on. All of these are

![Fig. 8. The overall description of representing and reasoning on the FOOD model with \( f\text{-ALCIQ}(D) \).](image_url)
consistent with the results mentioned in Example 3. Notice that, in Fig. 8, we consider the complete FOOD model in Fig. 3 and its database instance in Fig. 4, which contain all information of the simplified FOOD model and object instances in Example 3.

Until now, it is possible to represent and reason on FOOD models with the fuzzy DL.

(i) Based on Section 4, a fuzzy object-oriented database (including a FOOD model and object instances) can be automatically translated into an $f$-ALCIQ($D$) knowledge base.

(ii) On the basis of the translated $f$-ALCIQ($D$) knowledge base and Section 5, reasoning on the FOOD model can be reduced to reasoning on the $f$-ALCIQ($D$) knowledge base, so that reasoning tasks of the FOOD model may be checked by means of the existing reasoning techniques for $f$-ALCIQ($D$) [58,7].

(iii) Reasoning in $f$-ALCIQ($D$) is a decidable procedure (it is known that reasoning in $f$-SROIQ($D$), an $f$-ALCIQ($D$) extended language, is a decidable procedure [58,7]), and thus it is shown that reasoning on FOOD models is decidable. Furthermore, the existing fuzzy DL reasoner DeLorean [8] supports the fuzzy DL $f$-ALCIQ($D$) with general concept inclusions (GCIs). Therefore, reasoning on FOOD models may be done automatically.

6. Related work

Regarding how to represent and reason on (fuzzy) database models by means of (fuzzy) DLs, the following approaches are related to our work according to their focuses: classical database modeling with DLs, fuzzy object-oriented database models, fuzzy DLs, and correspondences between fuzzy database models and fuzzy DLs.

Currently, there are several researches on classical database modeling with DLs. Borgida [4] introduced the applications of DLs in data management. Bresciani [13] pointed out that DLs can be useful for improving some database applications such as data archaeology, query optimization, and database reasoning. In detail, Bonifati [12] studied how to extract useful properties from database schemas by applying DLs. A formal framework based on DLs was proposed in [28] for checking the consistency of structured knowledge. Berardi [9] studied how to reason on UML class diagrams using DLs. Calvanese [20] established the relationships between several data models (including ER model and object-oriented data model) and DLs at schema level. The correspondences between the extended ER models (Temporal ER model and EER model) and DLs were established in [1,25]. Moreover, Artale [2] introduced how to describe database objects in a concept language environment. Beneventano [14] investigated how to apply DLs for semantic query optimization in object-oriented database systems. Roger [47] investigated how to bring together DLs and database in an object-oriented model. In addition, the relationships between DLs and relational databases were studied in [19,29,42].

However, the above researches were not sufficient for handling imprecise and uncertain information that commonly exists in many real-world applications. In particular, the fuzzy set theory [70], which has been identified as a successful technique for modeling the imprecise and uncertain data, is extensively introduced into databases, information systems, the Semantic Web, and so on [33,37,46,51].

In order to model imprecise and uncertain information in object-oriented databases, Zicari in [69] first introduced incomplete information, namely, null values, where incomplete schema and incomplete objects can be distinguished. From then on, the incorporation of imprecise and uncertain information in object-oriented databases has increasingly received attention. A fuzzy object-oriented database model was defined in [11] based on the extension of a graphs-based object model. Based on similarity relationship, uncertainty management issues in the object-oriented database model were discussed in [26]. Based on possibility theory, vagueness and uncertainty were represented in class hierarchies in [22]. In more detail, also based on possibility distribution theory, Ma in [35] introduced fuzzy object-oriented database models, some major notions such as objects, classes, objects–classes relationships and subclass/superclass relationships were extended under fuzzy information environment. Moreover, other fuzzy extensions of object-oriented databases were developed. In [39,40], fuzzy types were added into fuzzy object-oriented databases to manage vague structures. The fuzzy relationships and fuzzy behavior in fuzzy object-oriented database models were discussed in [16,27]. Several intelligent fuzzy object-oriented database architectures were proposed in [30,44,45]. The other efforts on how to model fuzziness and uncertainty in object-oriented database models were done in [34,41,61]. The fuzzy and probabilistic object bases [17,43], fuzzy deductive object-oriented databases [64], and fuzzy object-relational databases [18] were also developed.
In addition, an object-oriented database modeling technique was proposed based on the level-2 fuzzy sets in [23], where the authors also discussed how the object Data Management Group (ODMG) data model can be generalized to handle fuzzy data in a more advantageous way. Also, the other efforts have been paid on the establishment of consistent framework for a fuzzy object-oriented database model based on the standard for the ODMG object data model [15]. More recently, how to manage fuzziness on conventional object-oriented platforms was introduced in [10]. For a comprehensive review of fuzzy object-oriented database models, refer to [31,36,37,46,65].

In order that DLs can deal with imprecise and uncertain knowledge, there have been many efforts in the past to extend DLs with fuzzy set theory. The initial idea combining fuzzy logic and DLs was presented by Yen in [66]. A later approach was presented by Tresp and Molitor [60]. Typically, a fuzzy extension of ALC language called FALC was proposed and the reasoning algorithm based on tableaux calculus was provided in [49]. From then on, the approaches towards more expressive fuzzy DLs such as ALCQ⁺ [48] and f-ALCIQ [55] were presented. Furthermore, by extending FALC with transitive role axioms (S), inverse roles (I), role hierarchies (H) and number restrictions (N), a fuzzy DL named f-SHIN and the complete reasoning algorithms for f-SHIN were provided in [54]. A fuzzy extension of the corresponding DL of the ontology description language OWL DL, called f-SHOIN(D), was proposed in [50]. A more expressive fuzzy DL called f-SROIQ(D) was also presented in [58], and was further studied in [5]. Moreover, some fuzzy DL reasoners have been implemented, such as FuzzyDL [6], FiRE [57], and DeLorean [8] (see Refs. [6,7,62]). For a comprehensive review of fuzzy DLs, refer to [33,51,54].

Although there have been several kinds of fuzzy extensions of database models and DLs, less research has been done in establishing correspondences between fuzzy database models and fuzzy DLs. A fuzzy DL FDLR was proposed in [67] to represent and reason on fuzzy ER models. Furthermore, how to represent and reason on fuzzy UML models based on the FDLR was also investigated in [38], where the focus was on the structural aspects of fuzzy UML models. Also an approach for constructing fuzzy OWL DL ontologies from fuzzy object-oriented database (FOOD) models was presented in [68]. The current paper differs from [68] in three major aspects: (i) the goals of two papers are different, the work in [68] aimed at constructing fuzzy ontologies from FOOD models to support fuzzy ontology development, and this paper is to represent and reason on FOOD models by means of the high expressive power and effective reasoning services of fuzzy DLs; (ii) some features of FOOD models (e.g., the behavioral aspects of objects, the disjointness and covering constraints of classes, etc.) are considered in this paper but not in [68], and how to reason on FOOD models was also missed in [68]; (iii) this paper gives the semantic interpretation of FOOD models, which was missed in [68]. To our best knowledge, so far, there is no report on representing and reasoning on fuzzy object-oriented database models with fuzzy DLs.

7. Conclusion and future work

We have proposed a description logic (DL) approach for representing and reasoning on fuzzy object-oriented database (FOOD) models. The FOOD models were further investigated, and a formal definition and semantics of FOOD models were proposed. To represent and reason on FOOD models, a fuzzy DL called f-ALCIQ(D) was recalled. On this basis, we proposed an approach that can translate a FOOD model and its corresponding data instances into an f-ALCIQ(D) knowledge base (i.e., TBox and ABox), and a prototype translation tool FOOD2DL was implemented. Furthermore, based on the translated f-ALCIQ(D) knowledge bases, we gave formal definitions of reasoning tasks of FOOD models and investigated how to reason on FOOD models through the reasoning mechanism of f-ALCIQ(D).

In summary, the fuzzy DL f-ALCIQ(D) can provide an adequate expressive power to account for the essential features of FOOD models, and such a logic language has sound, complete and decidable reasoning procedures. The formalization in f-ALCIQ(D) of FOOD models can be considered as the basic steps towards developing intelligent systems that provide computer-aided support to improve some database applications. All of these will also facilitate the realization of semantic interoperations between the existing fuzzy database applications and the Semantic Web.

The future directions including: (i) we intend to investigate how DLs can be used for improving other database applications such as query processing and data mining. (ii) We intend to study relationships between classical/fuzzy DLs and other database models. (iii) We realize that the fuzzy object-oriented databases discussed in this paper do not cover all kinds of fuzzy extensions of object-oriented databases. Some fuzzy object-oriented databases (e.g., fuzzy object-oriented databases based on similarity relationship [26] and fuzzy deductive object-oriented databases [64]) may be important for applications, and we will consider them in the near future work.
Acknowledgements

The authors thank the anonymous referees for their valuable comments and suggestions, which improved the technical content and the presentation of the paper. Many thanks are due to Ruizhe Ma for reading our paper and offering editorial advice. The work is supported by National Natural Science Foundation of China (61073139 and 60873010) and Fundamental Research Funds for the Central Universities (N090504005 and N090604012), and by Program for New Century Excellent Talents in University (NCET-05-0288).

References

[23] db4o, Open Source Object Database [http://www.db4o.com].


