Fuzzy Description Logic Reasoning using a Fixpoint Algorithm[⋆]

Uwe Keller[1] and Stijn Heymans[2]

[1] Semantic Technology Institute (STI) Innsbruck, University of Innsbruck, Austria.
uwe.keller@sti2.at
[2] Knowledge-based Systems Group, Institute of Information Systems, Vienna University of Technology, Austria.
heymans@kr.tuwien.ac.at

Abstract. We present $\mathbf{FixIt}(\mathbb{ALC})$, a novel procedure for deciding knowledge base (KB) satisfiability in the Fuzzy Description Logic (FDL) $\mathbb{ALC}$. $\mathbf{FixIt}(\mathbb{ALC})$ does not search for tree-structured models as in tableau-based proof procedures, but embodies a (greatest) fixpoint-computation of canonical models that are not necessarily tree-structured, based on a type-elimination process. Soundness, completeness and termination are proven and the runtime and space complexity are discussed. We give a precise characterization of the worst-case complexity of deciding KB satisfiability (as well as related terminological and assertional reasoning tasks) in $\mathbb{ALC}$ in the general case and show that our method yields a worst-case optimal decision procedure (under reasonable assumptions). To the best of our knowledge it is the first *fixpoint-based* decision procedure for FDLs, hence introducing a new class of inference procedures into FDL reasoning.

1 Introduction

Description Logics (DLs) [1] are a popular family of formally well-founded and decidable knowledge representation languages. DLs have a wide range of applications, e.g., they form the basis for Semantic Web (SW) ontology languages used such as OWL [11]. Fuzzy Description Logics (FDLs) [17] extend DLs to represent *vague* concepts and relations, and as such are very well suited to cover for representing and reasoning with uncertainty, a requirement that naturally arises in many practical applications of knowledge-based systems, in particular the SW; FDLs for instance fit very well to the problem of multimedia information retrieval [9]. Another feature that makes FDLs specifically interesting for the SW is a basic form of para-consistency, i.e. a statement and its negation are possible to hold at the same time (to a certain extent). This allows knowledge providers on the SW to disagree on the basic properties of data objects and their interrelation without causing the (uninformative) explosion of the deductive closure as in classical DLs.

So far, reasoning in Fuzzy DLs is mainly based on tableau-methods (e.g. [17, 16, 7, 15, 20, 3]). Further, [18] demonstrates how to use inference procedures for classical

⋆ This work has been partially funded by the European Commission under the LarKC project (FP7 - 215535). Stijn Heymans is supported by the Austrian Science Fund (FWF) under projects P20305 and P20840.

DLs to perform reasoning in (some) FDLs. Still, reasoning in FDLs is at least as hard as reasoning in classical (crisp) DLs. Even in DLs of modest expressivity (e.g. $\mathbb{ALC}$ [17, 18, 16] the fuzzy variant of the DL $\mathcal{ALC}$ [14]) the worst-case complexity of reasoning is significant (cf. Section 3) even in restricted cases [17]. Therefore, it is clear that there can not be a *single* inference method that works well on *all* problems.

Consequently, our goal is to enrich the range of available methods for reasoning with FDLs with a fundamentally different approach. In practical applications of DLs (and hence FDLs) a particularly important feature for representing domain models is the support of so-called *general terminologies* (see e.g. [16]), i.e., the possibility to capture (potentially recursive) interdependencies between complex concepts in a domain model. However, besides the tableau-based methods for DLs (e.g [16, 7, 20, 3]) there are at present no other FDL inference methods which can deal with general terminologies. We want to provide an alternative to tableau-based methods that can deal with general terminologies.

The main contributions of the paper are as follows:

– We present a novel procedure $\mathbf{FixIt}(\mathbb{ALC})$ (cf. Section 4.2) for deciding knowledge base (KB) satisfiability in the FDL $\mathbb{ALC}$ (cf. Section 2).
– We clarify the worst-case complexity of the reasoning task addressed by our algorithm and show formally that the problem is EXPTIME-complete. From this result, we can further establish EXPTIME-completeness for a range of related terminological and assertional reasoning tasks (cf. Section 3).
– We formally prove soundness, completeness and termination of the algorithm (cf. Section 4.2) and show that the runtime behavior of the proposed algorithm is worst-case optimal (cf. Section 4.3).
– $\mathbf{FixIt}(\mathbb{ALC})$ generalizes a type-elimination-based decision procedure [12] for the (classical) modal logic $\mathbf{K}$ (i.e. $\mathcal{KBDD}$ [10]) to the FDL $\mathbb{ALC}$. Additionally we integrate (fuzzy) ABoxes and general TBoxes which are not dealt with in $\mathcal{KBDD}$.
– To the best of our knowledge it is the first *fixpoint-based* decision procedure that has been proposed for FDL introducing a *new class of inference procedures* into FDL reasoning.
– Besides the tableau-based methods in [16, 7, 20, 3], it is the only approach to integrate general terminologies in FDL reasoning and the first non-tableau-based one that we are aware of. General terminologies are handled in a fundamentally different way than in standard tableau-based method such as [16, 7].

Our method is interesting especially regarding the last aspect since the handling of general terminologies in standard tableau-based methods (e.g. [16, 7]) is a *major* source of non-determinism (cf. Section 5) and thus computational inefficiency. In our case no non-deterministic choice is introduced by terminologies.

2 Preliminaries

We introduce $\mathbb{ALC}$ [17], the fuzzy variant of the Description Logic $\mathcal{ALC}$ [14] (the latter can be seen as a syntactic variant of the multi-modal logic $\mathbf{K}_{(\mathbf{m})}$ [13]). $\mathbb{ALC}$ provides the starting point for more expressive FDLs [19] that have been proposed to fuzzify major fragments of OWL [11].

Syntax. Concept expressions are constructed from a signature $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ with concept names $\mathbf{C}$, role names $\mathbf{R}$, and individual names $\mathbf{I}$. The set of concept expressions $\mathcal{C}(\Sigma)$ over Σ is defined as the smallest set of expressions that contains $\mathbf{C}$, $\top$ and is closed under the application of the concept constructors $C \sqcap D$ (intersection), $C \sqcup D$ (union), $\neg C$ (complement), and $\forall R.C$ (universal role restriction) for $R \in \mathbf{R}$ and $C, D \in \mathcal{C}(\Sigma)$. We allow expressions $\exists R.C$ for $C \in \mathcal{C}(\Sigma), R \in \mathbf{R}$ and $\bot$ and treat them as shortcuts for $\neg\forall R.\neg C$ and $\neg\top$ respectively. A TBox axiom (or general concept inclusion axiom (GCI)) is an expression of the form $C \sqsubseteq D$ s.t. $C, D \in \mathcal{C}(\Sigma)$. A terminology (or TBox) $\mathcal{T}$ is a finite set of TBox axioms. Syntactically, the vagueness of descriptions becomes explicit only when describing specific instances and their interrelations: a (fuzzy) ABox axiom is either a $\langle i : C \bowtie d\rangle$ or a $\langle R(i, i') \geq d\rangle$ s.t. $i, i' \in \mathbf{I}$, $d \in [0, 1]$, and $\bowtie \in \{\leq, \geq, =\}$. An ABox $\mathcal{A}$ is a finite set of ABox axioms. Finally, a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. Let $Ind_{\mathcal{A}} \subseteq \mathbf{I}$ denote the individual names that occur in $\mathcal{A}$. We denote the set of all concept expressions that occur as subexpressions in $\mathcal{K}$ by $\mathsf{sub}(\mathcal{K})$.

Semantics. Semantically, vagueness is reflected in the use of fuzzy sets and relations when interpreting concepts and roles: an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ called the domain, and a function $\cdot^{\mathcal{I}}$ which maps each concept name $C \in \mathbf{C}$ to a fuzzy set $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$, each role name $R \in \mathbf{R}$ to a fuzzy relation $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$ and each individual name $i \in \mathbf{I}$ to an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to arbitrary concept expressions $C \in \mathcal{C}(\Sigma)$ as follows: 1. $(C \sqcap D)^{\mathcal{I}}(o) = min(C^{\mathcal{I}}(o), D^{\mathcal{I}}(o))$ 2. $(C \sqcup D)^{\mathcal{I}}(o) = max(C^{\mathcal{I}}(o), D^{\mathcal{I}}(o))$ 3. $(\neg C)^{\mathcal{I}}(o) = 1 - C^{\mathcal{I}}(o)$ 4. $(\forall R.C)^{\mathcal{I}}(o) = inf_{o' \in \Delta^{\mathcal{I}}}\{max(1 - R^{\mathcal{I}}(o, o'), C^{\mathcal{I}}(o'))\}$ 5. $\top^{\mathcal{I}}(o) = 1$ for all $o \in \Delta^{\mathcal{I}}, C, D \in \mathcal{C}(\Sigma)$, $R \in \mathbf{R}$. Note that, in contrast to classical DLs, it does not hold that $(C \sqcup \neg C)^{\mathcal{I}} = \top^{\mathcal{I}}$ for all interpretations $\mathcal{I}$, hence the need to add $\top$ (or $\bot$) to the language explicitly.

An interpretation $\mathcal{I}$ satisfies a TBox axiom $\alpha = C \sqsubseteq D$ iff for all $o \in \Delta^{\mathcal{I}}$ it holds that $C^{\mathcal{I}}(o) \leq D^{\mathcal{I}}(o)$, i.e. C is a fuzzy subset of D. $\mathcal{I}$ satisfies an ABox axiom $\alpha = \langle i : C \bowtie d\rangle$ iff $C^{\mathcal{I}}(i^{\mathcal{I}}) \bowtie d$. $\mathcal{I}$ satisfies an ABox axiom $\alpha = \langle R(i, i') \geq d\rangle$ iff $R^{\mathcal{I}}(i^{\mathcal{I}}, i'^{\mathcal{I}}) \geq d$. In all these cases, we write $\mathcal{I} \models \alpha$. $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ (or is a model of $\mathcal{T}$) iff $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T}$. $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ (or is a model of $\mathcal{A}$) iff $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{A}$. Finally, $\mathcal{I}$ satisfies a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (or is a model of $\mathcal{K}$) iff $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$.

Reasoning in $\mathbb{ALC}$. Given a fuzzy KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, fuzzy ABox axioms or GCIs α and concept expressions $C, D \in \mathcal{C}(\Sigma)$, we can analyze particular semantic characteristics and interdependencies: We say that $\mathcal{K}$ is *satisfiable* (or consistent) iff there is a model $\mathcal{I}$ for $\mathcal{K}$. $\mathcal{K}$ *entails* α (denoted as $\mathcal{K} \models \alpha$) iff all models $\mathcal{I}$ of $\mathcal{K}$ satisfy α. Concept C is subsumed by concept D (wrt. a KB $\mathcal{K}$) iff $\mathcal{K} \models C \sqsubseteq D$. Two concepts C and D are called *equivalent* (wrt. a KB $\mathcal{K}$) iff for any model $\mathcal{I}$ of $\mathcal{K}$ it holds that $C^{\mathcal{I}}(o) = D^{\mathcal{I}}(o)$ for all $o \in \Delta^{\mathcal{I}}$. Two concepts C and D are called *disjoint* (wrt. a KB $\mathcal{K}$) iff for any model $\mathcal{I}$ of $\mathcal{K}$ it holds that there does not exists an $o \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(o) > 0$ and $D^{\mathcal{I}}(o) > 0$. A concept C is called satisfiable (wrt. a KB $\mathcal{K}$) iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}(o) > 0$ for some $o \in \Delta^{\mathcal{I}}$. Further, one might want to compute the truth value bounds for a given ABox assertion α wrt. $\mathcal{K}$ to determine the possibility interval that is enforced for α by the background knowledge in $\mathcal{K}$: The *greatest lower bound* of α wrt. $\mathcal{K}$ is defined as $glb(\alpha, \mathcal{K}) := sup\{d \,|\, \mathcal{K} \models \langle \alpha \geq d \rangle\}$ and the *least upper bound* of α wrt. $\mathcal{K}$ is defined as $lub(\alpha, \mathcal{K}) := inf\{d \,|\, \mathcal{K} \models \langle \alpha \leq d \rangle\}$ (where $sup\, \emptyset = 0$ and $inf\, \emptyset = 1$). Computing $glb(\alpha, \mathcal{K})$ and $lub(\alpha, \mathcal{K})$ is usually called the *best truth value bounds* (BTVB) problem.

One of the most fundamental reasoning problems is to determine whether a given fuzzy KB $\mathcal{K}$ is satisfiable. A lot of other reasoning tasks (e.g., checking for concept satisfiability wrt. a TBox or the BTVB problem) can be reduced to KB satisfiability checking [17] and therefore solved by a respective decision procedure. For this reason, we consider KB satisfiability as the reasoning problem to be solved.

3 Complexity of Reasoning with Knowledge Bases

Deciding the satisfiability of KBs in $\mathbb{ALC}$ where the TBox $\mathcal{T}$ is restricted to axioms of the form $A \sqsubseteq C$ or $A \equiv C$ (for concept names $A \in \mathbf{C}$ and concept expressions $C \in \mathcal{C}(\Sigma)$) such that any concept name A occurs at most once on the left-hand side and the TBox does not contain any cyclic dependencies between concept names is known to be a PSPACE-complete problem [17]. For the general case of unrestricted terminologies (allowing arbitrary GCIs $C \sqsubseteq D$), we are not aware of any worst-case complexity characterization. We show that determining the satisfiability of a KB in the general case is EXPTIME-complete (which corresponds to the situation in the classical variant $\mathcal{ALC}$). Detailed proofs are omitted here but can be found in [6].

EXPTIME-*Hardness.* We show EXPTIME-hardness by a polynomial-time (many-one) reduction of concept satisfiability wrt. general terminologies in the classical DL $\mathcal{ALC}$ which is known to be an EXPTIME-hard problem [1, Chapter 3]. We define the necessary reduction function as follows:

Definition 1 (Reduction). *Let* T *denote a finite set of GCIs and* $C \in \mathcal{C}(\Sigma)$ *be any concept expression. Then we define the reduction* $\pi(\mathrm{T}, C)$ *of the terminology* T *and the concept expression* C *as* $\pi(\mathrm{T}, C) := \mathcal{K}$ *where* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *is the* $\mathbb{ALC}$ *knowledge base consisting of the TBox* $\mathcal{T} := \mathrm{T} \cup \mathcal{T}^*$ *with* $\mathcal{T}^* := \{\top \sqsubseteq \neg D \sqcup D \,|\, D \in \mathsf{sub}(\mathrm{T} \cup \{C\})\}$ *and the ABox* $\mathcal{A} := \{\langle i : C \geq 1 \rangle\}$ *for some new individual name* i*.*

The intention of the additional TBox components $\mathcal{T}^*$ ensures that any model of $\mathcal{T}$ (and hence $\pi(\mathrm{T}, C)$) assigns only possibility degrees in $\{0, 1\}$ to any concept (sub-)expression occurring somewhere in T or C. In particular, any such model always assigns classical truth values to any concept name $A \in \mathbf{C}$ and any concept expression that is constructed from a role $R \in \mathbf{R}$ (but not necessarily to the roles $R \in \mathbf{R}$ themselves). Since for possibility degrees in $\{0, 1\}$ the (fuzzy) semantics of concept constructors in $\mathbb{ALC}$ coincides with the classical semantics in $\mathcal{ALC}$, we know that a fuzzy interpretation that satisfies $\mathcal{T}$ can be modified (easily) into a classical (i.e. crisp) model of T. By the same line of argumentation, $\mathcal{A}$ ensures that in any model of $\pi(\mathrm{T}, C)$ (and hence $\mathcal{T}$) the input concept C is $\mathcal{ALC}$-satisfiable. The implication in the other direction is immediate since any crisp interpretation is a model of the additional TBox components $\mathcal{T}^*$:

Proposition 1. *C is satisfiable wrt. T in $\mathcal{ALC}$ iff $\pi(\mathrm{T}, C)$ is satisfiable in $\mathbb{ALC}$.*

It is straightforward to see that the many-one reduction $\pi(\mathrm{T}, C)$ can be computed in linear time (wrt. the size of T and C) for each finite set T of GCIs and concept expressions $C \in \mathcal{C}(\Sigma)$. As an immediate consequence we get the following corollary:

Corollary 1 (EXPTIME-Hardness of KB Satisfiability). *The problem of deciding the satisfiability of KBs in $\mathbb{ALC}$ is* EXPTIME*-hard if GCIs are allowed.*

EXPTIME-*Membership.* KB satisfiability in $\mathbb{ALC}$ is in EXPTIME since [18] shows that checking KB satisfiability in $\mathbb{ALC}$ can be reduced (in polynomial time) to checking KB satisfiability in $\mathcal{ALC}$ which is known to be in EXPTIME, since KB satisfiability is in EXPTIME even for an extension of $\mathcal{ALC}$, i.e. the more expressive DL $\mathcal{SHIQ}$ [21, Corollary 6.30].

Theorem 1 (EXPTIME-Completeness of KB Satisfiability). *The problem of deciding the satisfiability of KBs in $\mathbb{ALC}$ is* EXPTIME*-complete if GCIs are allowed.*

Besides KB satisfiability, one can show that also the following reasoning problems are EXPTIME-complete: $\mathcal{K} \models C \equiv D$, $\mathcal{K} \models C \sqsubseteq D$, concept disjointness wrt. $\mathcal{K}$, concept satisfiability wrt. $\mathcal{K}$, $\mathcal{K} \models \langle o : C \geq n \rangle$, $\mathcal{K} \models \langle o : C \leq n \rangle$, and $\mathcal{K} \models \langle o : C = n \rangle$; we refer the reader for more details to [6].

4 A Decision Procedure based on Fuzzy Type Elimination

We present a decision procedure for KB satisfiability in $\mathbb{ALC}$ which does not rely on systematic search in the first place (as e.g. tableau-based methods), but instead constructs a canonical interpretation by means of a fixpoint construction. The so-constructed (canonical) interpretation (if non-empty) satisfies the TBox of a KB and allows to derive a model for the given knowledge base $\mathcal{K}$ iff $\mathcal{K}$ is satisfiable. In contrast to tableau-based procedures a canonical interpretation is in general *not* tree-shaped. Further, it can be shown that the number of iterations required to reach a fixpoint is *linear* in the modal depth of $\mathcal{K}$.

Preprocessing. Without loss of generality, we can restrict ourselves to *normalized* knowledge bases [16], i.e. knowledge bases which contain only fuzzy ABox assertions of the form $\langle \alpha \geq d \rangle$, by applying the following equivalent transformation fuzzy ABox axioms: $\langle i : C \leq d \rangle \rightsquigarrow \langle i : \neg C \geq 1 - d \rangle$ and $\langle i : C = d \rangle \rightsquigarrow \langle i : C \geq d \rangle, \langle i : \neg C \geq 1 - d \rangle$. Further, we can assume that all axioms in $\mathcal{K}$ are in box normal form (BNF) [10] (i.e. the only negative concept subexpressions are of the form $\neg \forall R.C$ or negated atomic concept names $\neg C$), by exhaustively applying the following equivalent transformation to concept expressions: $\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D$, $\neg(C \sqcup D) \rightsquigarrow \neg C \sqcap \neg D$, and $\neg\neg C \rightsquigarrow C$. These preprocessing steps can be performed altogether in linear time wrt. the size of the input KB.

4.1 Basic Notions and Intuition

Types. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ denote a normalized $\mathbb{ALC}$ knowledge base in BNF. The *closure* of a knowledge base $\mathsf{cl}(\mathcal{K})$ is defined as the smallest set of concept expressions such that for all $C \in \mathsf{sub}(\mathcal{K})$, if C is not of the form $\neg D$, then $\{C, \neg C\} \subseteq \mathsf{cl}(\mathcal{K})$. Further, let $\mathsf{PossDeg}(\mathcal{K})$ denote the set of all relevant possibility degrees that can be derived from $\mathcal{K}$, i.e. $\mathsf{PossDeg}(\mathcal{K}) = \{0, 0.5, 1\} \cup \{d | \langle \alpha \geq d \rangle \in \mathcal{A}\} \cup \{1 - d | \langle \alpha \geq d \rangle \in \mathcal{A}\}$. It has been shown in [17, 18] that if $\mathcal{K}$ is satisfiable, then there is as well a model of $\mathcal{K}$ which assigns possibility degrees in $\mathsf{PossDeg}(\mathcal{K})$ only. Hence, for our purposes we do not need to consider arbitrary possibility degrees $d \in [0, 1]$, but only the *finite* set $\mathsf{PossDeg}(\mathcal{K})$ that can be derived from $\mathcal{K}$.

We can then introduce the notion of a *type*, which allows to represent individuals of an interpretation in a syntactic way:

Definition 2 (Fuzzy $\mathcal{K}$-Type). *A* fuzzy $\mathcal{K}$-type τ *is a maximal subset of* $\mathsf{cl}(\mathcal{K}) \times \mathsf{PossDeg}(\mathcal{K})$ *such that the following conditions are satisfied: 1. if $\langle C, d \rangle \in \tau$ and $\langle C, d' \rangle \in \tau$ then $d = d'$ 2. if $C = \neg C'$ then $\langle C, d \rangle \in \tau$ iff $\langle C', 1 - d \rangle \in \tau$ 3. if $C = C' \sqcap C''$ then $\langle C, d \rangle \in \tau$ iff $\langle C', d' \rangle \in \tau$ and $\langle C'', d'' \rangle \in \tau$ and $d = min(d', d'')$ 4. if $C = C' \sqcup C''$ then $\langle C, d \rangle \in \tau$ iff $\langle C', d' \rangle \in \tau$ and $\langle C'', d'' \rangle \in \tau$ and $d = max(d', d'')$ 5. for all $C \sqsubseteq C' \in \mathcal{T}$: if $\langle C, d \rangle \in \tau$ and $\langle C', d' \rangle \in \tau$ then $d \leq d'$ 6. if $C = \top$ then $\langle C, 1 \rangle \in \tau$.*

Since $\mathsf{cl}(\mathcal{K})$ and $\mathsf{PossDeg}(\mathcal{K})$ are both finite sets, there are at most $2^{|\mathsf{cl}(D)| \cdot |\mathsf{PossDeg}(\mathcal{K})|}$ different $\mathcal{K}$-types. Each type τ can be seen as an individual and syntactically represents *all* (fuzzy) properties that can be observed about that individual: $\langle C, d \rangle \in \tau$ indicates that the individual τ belongs to C with the possibility degree d. Hence, the set of all $\mathcal{K}$-types (or simply types) provides enough vocabulary to let us describe all kinds of interpretations for $\mathcal{K}$ simply by fixing how to interconnect individuals (and therefore types).

Canonical Model. It turns out that it is possible to connect types in a fixed (or canonical) way, such that the interconnection defined is consistent with *almost* all properties specified syntactically in the type. The interconnections can be derived from the types themselves:

For a set of types T we can define for each role R a *canonical accessibility relation* $\Delta_R : T \times T \to \mathsf{PossDeg}(\mathcal{K})$ that "maximally" interconnects types $\tau, \tau' \in T$ with possibility degree $d \in \mathsf{PossDeg}(\mathcal{K})$: let $\delta(d, d') := 1$ if $d \leq d'$ and $\delta(d, d') := 1 - d$ if $d > d'$. Then, we can define Δ_R by

$$\Delta_R(\tau, \tau') := min\{\delta(d, d') | \langle \forall R.C, d \rangle \in \tau, \langle C, d' \rangle \in \tau'\}$$

if $\forall R.C \in \mathsf{cl}(\mathcal{K})$ for some $C \in \mathbf{C}$, and $\Delta_R(\tau, \tau') := 1$ otherwise.

This way, we can construct a canonical interpretation $\mathcal{I}_T$ for any given set of types T using the canonical interconnection of types by Δ_R as follows: $\mathcal{I}_T = (T, \cdot^{\mathcal{I}_T})$ with (i) for any concept name C in $\mathcal{K}$ and any $\tau \in T$ we set $C^{\mathcal{I}_T}(\tau) = d$ if $\langle C, d \rangle \in \tau$, and (ii) $R^{\mathcal{I}_T}(\tau, \tau') = \Delta_R(\tau, \tau')$ for any role R in $\mathcal{K}$ and any $\tau, \tau' \in T$. Please note, that by our definition of $\mathcal{K}$-types, $\mathcal{I}_T$ is well-defined for any concept name or role name. However, our definition deliberately leaves open the interpretation of individuals. We therefore define in fact a class of canonical interpretations, each of which fixes a specific way of how to interpret the individuals in a KB $\mathcal{K}$.

The canonical interconnection in $\mathcal{I}_T$ is chosen in such a way that all assignments of possibility degrees to concepts of the form $C = \forall R.C \in \tau$ are lower bounds for the possibility degrees that are in fact assigned by a canonical interpretation $\mathcal{I}_T$. Hence, such a canonical interpretation is *almost* immediately a (canonical) model for the terminology $\mathcal{T}$, i.e. it satisfies that

$$C^{\mathcal{I}_T}(\tau) = d \text{ iff } \langle C, d\rangle \in \tau \qquad (*)$$

for *almost* all $C \in \mathsf{cl}(\mathcal{K})$ and therefore $\mathcal{I}_T \models C \sqsubseteq C'$ for all $C \sqsubseteq C' \in \mathcal{T}$ by clause (5) in our definition of $\mathcal{K}$-types. That $(*)$ is satisfied by $\mathcal{I}_T$ is straightforward for the cases of concept names C, or complex concepts of the form $C = C' \sqcap C''$, $C = C' \sqcup C''$, $C = \neg C'$ and the $C^{\mathcal{I}_T}(\tau) \geq d$ case for $C = \forall R.C$ by our definition of types and the definition of Δ_R. The only cases where $(*)$ can be violated by $\mathcal{I}_T$ is for types τ containing universally role restricted concepts $\forall R.C$ that are assigned a possibility degree which is *too small* (wrt. the R-successor types τ' in $\mathcal{I}_T$) to properly reflect the semantics of $\forall R.C$ in $\mathbb{ALC}$, i.e. to coincide with the *greatest* lower bound of the set

$$\{max(1 - R^{\mathcal{I}_T}(\tau, \tau'), C^{\mathcal{I}_T}(\tau')) \,|\, \tau' \in T\}$$

Types τ in which the possibility degree assigned d to $\forall R.C$ is too small to be consistent with the semantics of $\mathbb{ALC}$ are called *bad types*. Bad types $\tau \in T$ can be detected easily, since they satisfy that there exist $R \in \mathbf{R}, C \in \mathcal{C}(\Sigma), d \in \mathsf{PossDeg}(\mathcal{K})$ s.t. $\langle \forall R.C, d\rangle \in \tau$ and for all $\tau' \in T$: if $\langle C, d'\rangle \in \tau'$ then $max(1 - \Delta_R(\tau, \tau'), d') > d$.

This suggests the following simple algorithm (which uses a *fuzzy type elimination* process at its core): in order to compute a maximal interpretation that satisfies all terminological axioms, we start off with the maximal set of types (i.e all $\mathcal{K}$-types) and iteratively fix all problems that prevent $(*)$ from being satisfied by removing bad types. This way, we must eventually reach a fixpoint after finitely many steps. If the resulting set of types is non-empty, we know that $(*)$ must hold (since all problems have been fixed) and therefore we can be certain that the corresponding canonical interpretation satisfies $\mathcal{T}$ (and covers all other possible models of $\mathcal{T}$ at the same time). Hence, we eventually need to check if all ABox axioms are satisfied by the canonical interpretation. If this is the case, we have found a model for $\mathcal{K}$, otherwise, we know that there can not be any interpretation that satisfies both $\mathcal{T}$ and $\mathcal{A}$ at the same time. In other words, $\mathcal{K}$ is not satisfiable.

4.2 Algorithm

The type elimination process sketched above can be formalized as shown in Algorithm 1. Note that the emptiness test for the fixpoint T is covered implicitly: if the fixpoint T is empty, then the test in the if-statement fails trivially.

The termination, soundness, and completeness of our algorithm can be proven formally (cf. [6] for full proofs):

```
procedure satisfiable($\mathcal{K}$): boolean
T := $\{\tau | \tau$ is a $\mathcal{K}$-type $\}$;
repeat
    $T'$ := $T$;
    $T$ := $T' \setminus badtypes(T')$;
until $T = T'$ ;
if there exists a total function $\pi : Ind_{\mathcal{A}} \rightarrow T$ s.t. $\langle C, d' \rangle \in \pi(o)$ and $d \leq d'$ for each
$\langle o : C \geq d \rangle \in \mathcal{A}$, and $\Delta_R(\pi(o), \pi(o')) \geq d$ for each $\langle R(o, o') \geq d \rangle \in \mathcal{A}$ then
    return true;
end
return false;

function $badtypes(T) : 2^T$
return $\{\tau \in T | \langle \forall R.C, d \rangle \in \tau$ and for all $\tau' \in T$: if $\langle C, d' \rangle \in \tau'$ then
$max(1 - \Delta_R(\tau, \tau'), d') > d\}$;
```

Algorithm 1: The Type Elimination-based Decision Procedure $\mathbf{FixIt}(\mathbb{ALC})$

Theorem 2 (Termination). *For any $\mathbb{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ the algorithm $\mathbf{FixIt}(\mathbb{ALC})$ terminates after finitely many steps with either* true *or* false *as return value.*

The following lemma is a key element of the soundness and completeness proof and shows that by successively removing bad types we can indeed ensure that types encode possibility degree assignments to concepts that coincide with the canonical interpretation, and that any such canonical interpretation is a model of the $\mathcal{T}$.

Let T be the set of types that is computed as the fixpoint in the algorithm $\mathbf{FixIt}(\mathbb{ALC})$, i.e. $badtypes(T) = \emptyset$ and let $\mathcal{I}_T$= $(T, \cdot^{\mathcal{I}_T})$ be a canonical interpretation for T as defined above.

Lemma 1. *For each $\mathcal{K}$-type τ, concept $C \in \mathsf{cl}(\mathcal{K})$ and $d \in \mathsf{PossDeg}(\mathcal{K})$ it holds that $C^{\mathcal{I}_T}(\tau) = d$ iff $\langle C, d \rangle \in \tau$. Further, $\mathcal{I}_T \models \mathcal{T}$.*

Theorem 3 (Soundness). *If $\mathbf{FixIt}(\mathbb{ALC})$ returns true for a $\mathbb{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, then $\mathcal{K}$ is satisfiable.*

A second key element for the completeness proof is the following lemma that shows that our canonical way of interconnecting types (in the fixpoint set) is maximal or the strongest possible one in the following sense: the interconnection R of individuals o, o' defined by any model $\mathcal{I}$ of $\mathcal{K}$ is covered by the canonical interconnection Δ_R of the respective types $\tau(o), \tau(o')$ representing o, o' in $\mathcal{I}$.

Lemma 2. *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be any model of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. For each individual $o \in \Delta^{\mathcal{I}}$ we define its corresponding type $\tau(o) := \{\langle C, d \rangle \in \mathsf{cl}(\mathcal{K}) \times \mathsf{PossDeg}(\mathcal{K}) | C^{\mathcal{I}}(o) = d\}$. Then, $\Delta_R(\tau(o), \tau(o')) \geq R^{\mathcal{I}}(o, o')$ for all $o, o' \in \Delta^{\mathcal{I}}$.*

Theorem 4 (Completeness). *If an $\mathbb{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is satisfiable, then $\mathbf{FixIt}(\mathbb{ALC})$ returns true for $\mathcal{K}$.*

This leads to the main result, which is an immediate consequence of Theorems 3, 4, and 2:

Corollary 2. *The algorithm* $\mathbf{FixIt}(\mathbb{ALC})$ *is a sound and complete decision procedure for knowledge base satisfiability in* $\mathbb{ALC}$*.*

4.3 Runtime and Space Requirements

We now analyze the runtime and space requirements of our algorithm based on a naive implementation model to derive upper bounds. The considered implementation works directly on types and explicit representations of sets of types.

In the following we assume that the number p of different possibility degrees that can occur in any KB $\mathcal{K}$ is fixed. This assumption seems realistic and does not restrict applications of FDLs in practice, i.e., we can assume a limited (numerical) resolution of sensors and algorithms (or humans) assigning possibility degrees to individual observations. Further, we consider the computation of the basic functions $min, max, 1-d$ and comparisons $d \leq d'$ as atomic operations with unit costs.

Representation. According to Def. 2, a $\mathcal{K}$-type τ is a function $\tau : \mathsf{cl}(\mathcal{K}) \rightarrow \mathsf{PossDeg}(\mathcal{K})$ that satisfies a particular consistency property. Fix some total order $\langle C_1, C_2, \ldots, C_k\rangle$ on the subset $\mathsf{cl}_+(\mathcal{K})$ of $\mathsf{cl}(\mathcal{K})$ that consists of all positive concept expression $C_i \in \mathsf{cl}(\mathcal{K})$. Then, we can represent a type τ by a sequence of pairs

$$\tau = \langle d_1, \overline{d_1}\rangle, \langle d_2, \overline{d_2}\rangle, \ldots, \langle d_k, \overline{d_k}\rangle$$

with $k = |\mathsf{cl}_+(\mathcal{K})| \in O(|\mathcal{K}|)$, where d_i represents the possibility degree assigned to a positive concept subexpression $C_i \in \mathsf{cl}(\mathcal{K})$ and $\overline{d_i}$ represents the possibility degree assigned to a negative concept subexpression $\neg C_i \in \mathsf{cl}(\mathcal{K})$. There are only finitely many relevant possibility degrees. Assuming a binary encoding $\langle\cdot\rangle_{01} : \mathsf{PossDeg}(\mathcal{K}) \rightarrow \{0,1\}^p$ of possibility degrees, each such sequence requires at most $2 \cdot k \cdot log_2(p) \in O(k)$ bits. Clearly, not each such sequence represents a $\mathcal{K}$-type. However, for any sequence checking the consistency requirements from Def. 2 can be done time $O(k \cdot |\mathcal{T}|)$ and space $O(1)$: property (1) is satisfied already by our encoding, properties (2)-(3) and (6) can each be decided in $O(k)$, property (5) requires $O(k \cdot |\mathcal{T}|)$ computation steps. In any case, we need only $O(1)$ additional space for the computation.

The algorithm can be separated into three different phases: the initialization step, the fixpoint computation, and the final test. We analyze all of these steps one-by-one.

Initialization Phase. To compute the set of all $\mathcal{K}$-types, we generate any sequence $\langle d_1, \overline{d_1}\rangle, \langle d_2, \overline{d_2}\rangle, \ldots, \langle d_k, \overline{d_k}\rangle$ s.t. $d_i, \overline{d_k} \in \mathsf{PossDeg}(\mathcal{K})$. For each sequence this requires at most $2k$ steps. Testing if such a generated sequence satisfies the consistency requirements, takes at most $O(k \cdot |\mathcal{T}|)$ steps and only constant additional space. Any sequence that satisfies the consistency requirements is stored in a linked list. During the initialization phase, we need to check p^{2k} sequence, which requires $O(p^{2k} \cdot k \cdot |\mathcal{T}|)$ computation steps and $O(p^{2k} \cdot k)$ space.

Fixpoint Computation. The current set of types T in the fixpoint computation is represented as linked list from which elements are removed during this phase. Clearly, the function $\delta(d, d')$ can be computed in constant time. The computation of $\Delta_R(\tau, \tau')$ therefore can be performed in $O(k^2)$ computation steps and constant additional space for any two given types τ, τ'. Given a type $\tau \in T$, we can then determine if τ is a bad type (wrt. T) in $O(k \cdot (|T| \cdot k^2)) = O(k^3 \cdot |T|)$ basic computation steps using constant space only. This test has to be performed for all $|T|$ types in T in order to compute T'. Removing a bad type from the current list of types T requires $O(|T|)$ time and constant additional space. To find out if we reached a fixpoint after an iteration, we can simply use a boolean variable, which is set to $false$ at the beginning of each loop and set to $true$ if a type is removed from the list. This requires at most $O(|T|)$ additional assignments and one boolean comparison during each iteration and constant additional space. Hence, each iteration of the loop (to compute T') requires $O((k^3 + 1) \cdot |T|^2 + |T|)$ computation steps and constant additional space.

Let t denote the size of the initial set of types T. Since we need at most t iterations to reach a fixpoint, we can compute the fixpoint in at most $O((k^3 + 1) \cdot t^3 + t)$ steps using $O(t)$ additional memory units. Since $t \leq p^{2k}$, we can derive $O((k^3 + 1) \cdot p^{6k} + p^{2k})$ as an upper bound on the number of computation steps performed for the fixpoint computation and $O(p^{2k})$ as an upper bound for the additionally required space.

Final Test. We can represent a total total mapping $\pi : Ind_{\mathcal{A}} \rightarrow T$ as a mapping from $Ind_{\mathcal{A}}$ to the index of an element in the list T. This requires, $O(i \cdot log_2(t))$ additional memory units with $i = |Ind_{\mathcal{A}}|$ and lookups for values of π can be performed in $O(t)$ time using a hashing function and a subsequent iteration over the list representing T. Given such a mapping π, we can therefore determine if the required property in the if-statement in Algorithm 1 is satisfied by π in at most $O(a \cdot (t \cdot k^2))$ steps (where $a = |\mathcal{A}|$) using $O(1)$ additional memory only. We can generate any such mapping $\pi : Ind_{\mathcal{A}} \rightarrow T$ in at most $O(i \cdot log_2(t))$ steps. Further, there are at most t^i different such mapping, which we can generate and test one-by-one. Hence, we can implement that final test using at most $O(t^i \cdot (i \cdot log_2(t) + a \cdot (t \cdot k^2))) = O(p^{2ki} \cdot (i \cdot k \cdot log_2(p) + a \cdot (p^{2k} \cdot k^2)))$ steps and $O(i \cdot k \cdot log_2(p))$ additional memory units.

Overall runtime and space bounds. For an upper bound on the overall execution of the algorithm, we can sum up the runtime and space bounds for the three different phases above: In regard of runtime, the algorithm requires no more than $O(p^{2k} \cdot k \cdot |\mathcal{T}| + (k^3 + 1) \cdot p^{6k} + p^{2k} + (p^{2ki} \cdot (i \cdot k \cdot log_2(p) + a \cdot (p^{2k} \cdot k^2))))$ computation steps. In regard of the space, the algorithm requires no more than $O(p^{2k} \cdot k + p^{2k} + i \cdot k \cdot log_2(p) + |\mathcal{K}|)$ space. Since $k, i \in O(|\mathcal{K}|)$ and since we assume that p is a constant (and hence independent from a size of a KB $\mathcal{K}$), the algorithm requires at most exponentially many computations steps and an exponential amount of memory in regard of the size of the input KB $\mathcal{K}$.

This means that (under the realistic assumptions) our algorithm can be implemented in a way that is *worst-case optimal* in regard of the runtime of the algorithm, since by Theorem 1 the problem of determining the satisfiability of a KB in $\mathbb{ALC}$ requires exponential time (wrt. the size of the input KB) in the worst case. Further, note that applying a standard tableau-based decision procedure (e.g. [1]) to (crisp) equisatisfiable

$\mathcal{ALC}$-reduction of a general $\mathbb{ALC}$ KB usually yields only a NExpTime-upper-bound on the runtime of such an (indirect) decision procedure. Therefore, indirect reasoning approaches by reduction to classical tableau-based methods usually can only give *substantially worse* runtime guarantees.

Consequently, the major obstacles when using the algorithm in practice are (i) the exponential space requirement and (ii) the necessity to consider all types in $\tau \in T$ during each loop one-by-one. These problems are mainly based on the fact that we use *explicit* representations of set of types. A potential solution to these problems is known from the area of Symbolic Model Checking [8] and has already been successfully applied for the implementation of the $\mathcal{KBDD}$ procedure in [10]: the use *implicit* (or declarative) representations of sets of types by means of formulae and to implement set-theoretic operations by formula modifications and (un)satisfiability tests. In particular, set-theoretic operations on sets can be implemented as set-at-a-time operations instead of element-at-a-time operations which can speed up computation significantly.

5 Discussion: Reasoning with Terminologies in Fuzzy Tableau-based Methods

Tableau-methods can be used to determine the satisfiability of KBs. In order to detect the satisfiability status of a given KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, they construct tree-structured labeled graphs (i.e. completion graphs). The nodes in a completion graph represent individuals in an interpretation $\mathcal{I}$ for $\mathcal{K}$ and the edges capture the interrelation of individuals via roles in $\mathcal{K}$. Node labels capture (bounds on) the degree to which the individual represented by the node is a member of a concept, edge labels specify (bounds on) the degree of interrelation of the connected individuals wrt. roles. The completion graph is initialized to represent the ABox $\mathcal{A}$. Then, it is stepwise extended by analyzing concept labels of nodes and interrelations. An expansion can create new labels for nodes and edges and insert new nodes and edges into the graph. The extension process follows a set of so-called completion rules which are exhaustively applied. During the completion process non-deterministic choices can appear: there might be more than one way to extend the graph further, but we can not tell which of these extensions will eventually lead to a successful construction of a model for $\mathcal{K}$ (if there is one). In order to stay complete, all possible choices (for any node and any node label) need to be considered as long as no model has been constructed yet. The latter can be determined by checking labels for elementary contradictions (in regard of the specified bounds on membership degrees). If no completion rules are applicable anymore and there are no labels to nodes and edges which contain an elementary contradiction (i.e. we have a fully-expanded and clash-free completion graph), then we found in fact a (witness for a) model for $\mathcal{K}$. If no such graph can be constructed at all, $\mathcal{K}$ is unsatisfiable.

Example. Consider the ABox $\mathcal{A} = \{\langle i : \forall R.\neg B \sqcup \exists R.(B \sqcap C) \leq 0.4\rangle, \langle i : \forall R.C \geq 0.7\rangle\}$. The completion process starts with the completion graph

$$L_0 = \{\langle \forall R.\neg B \sqcup \exists R.(B \sqcap C) \leq 0.4\rangle, \langle \forall R.C \geq 0.7\rangle\}$$

Since i can be a member of $\forall R.\neg B \sqcup \exists R.(B \sqcap C)$ only up to degree 0.4, we can derive upper bounds on the possibility degree for i wrt. $\forall R.\neg B$ and $\exists R.(B \sqcap C)$ directly from the semantics of the concept constructor $\sqcup$ and update the label by two new constraints

$$\begin{tikzcd} \overset{L_1 = L_0 \cup \{\langle \forall R.\neg B \leq 0.4\rangle, \langle \exists R.(B \sqcap C) \leq 0.4\rangle\}}{\underset{\mathrm{i}}{\bullet}} \end{tikzcd}$$

To satisfy $\langle \forall R.\neg B \leq 0.4\rangle$, we introduce a new R-successor node j, a constraint on the degree of R-interrelation between i and j, and a constraint on the degree of membership of j in $\neg B$. Both constraints follow immediately from the semantics of the universal role restriction in $\mathbb{ALC}$:

$$\begin{tikzcd} \overset{L_1}{\underset{\mathrm{i}}{\bullet}} \arrow[r, "\{\langle R \geq 0.6\rangle\}" description] & \overset{L'_0 = \{\langle \neg B \leq 0.4\rangle\}}{\underset{\mathrm{j}}{\bullet}} \end{tikzcd}$$

To satisfy the constraint $\langle \forall R.C \geq 0.7\rangle \in L_1$ we add $\langle C \geq 0.7\rangle$ to L'_0. Further, in $\mathbb{ALC}$ $\langle \neg B \leq 0.4\rangle \in L'_0$ can be rewritten as $\langle B \geq 0.6\rangle$

$$\begin{tikzcd} \overset{L_1}{\underset{\mathrm{i}}{\bullet}} \arrow[r, "\{\langle R \geq 0.6\rangle\}" description] & \overset{L'_1 = \{\langle B \geq 0.6\rangle, \langle C \geq 0.7\rangle\}}{\underset{\mathrm{j}}{\bullet}} \end{tikzcd}$$

Finally, the only constraint that we did not consider yet is $\langle \exists R.(B \sqcap C) \leq 0.4\rangle \in L_0$. To ensure the satisfaction of this concept expression, we need to add $\langle B \sqcap C \leq 0.4\rangle$ (since the edge constraint $\langle R \geq 0.6\rangle$ and the upper bound $\langle R \leq 0.4\rangle$ are inconsistent with each other):

$$\begin{tikzcd} \overset{L_1}{\underset{\mathrm{i}}{\bullet}} \arrow[r, "\{\langle R \geq 0.6\rangle\}" description] & \overset{L'_2 = L'_1 \cup \{\langle B \sqcap C \leq 0.4\rangle\}}{\underset{\mathrm{j}}{\bullet}} \end{tikzcd}$$

Considering the semantics of the concept intersection in $\mathbb{ALC}$, $\langle B \sqcap C \leq 0.4\rangle$ is satisfied iff $\langle B \leq 0.4\rangle$ *or* $\langle B \leq 0.4\rangle$ are satisfied. Hence, we face a non-deterministic choice point in the completion process. Since all possible extensions lead to an unsatisfiable constraint systems node j in the completion graph, we can conclude that the given ABox $\mathcal{A}$ is unsatisfiable.

Tableau-based implementations deal with (or) non-deterministic choice points by backtracking. Since for any choice, an unsatisfiable node might be detected only after a lot of node label and graph extension steps, backtracking can become a very costly operation.

Integration of GCIs. In order to integrate GCIs into the completion graph extension process sketched above, standard tableau-based methods (e.g. [16], and similarly [7]) exploit the following observation: An interpretation $\mathcal{I}$ satisfies the GCI $C \sqsubseteq D$ (i.e. $\mathcal{I} \models C \sqsubseteq D$) iff for all (relevant) possibility degrees $n \in \mathsf{PossDeg}(\mathcal{K})$ and all individuals $i \in \Delta^{\mathcal{I}}$ either $\langle i : C < n\rangle$ or $\langle i : D \geq n\rangle$ holds.

To reflect this property in the tableau completion process, a non-deterministic rule for the extension of node labels is added to the inference system: for each node i in the completion graph, each GCI $C \sqsubseteq D$ in the TBox, and each (relevant) possibility degree n, we either insert the constraint $\langle i : C < n\rangle$ or $\langle i : D \geq n\rangle$ into the current node label.

It is obvious that using this new inference rule GCIs essentially become the *major source of non-determinism* for tableau-based inference procedures: for *each* node i appearing (*somewhen*) in the completion graph, we have a distinct alternative for *each* relevant possibility degree $n \in \mathsf{PossDeg}(\mathcal{K})$ and *each* GCI $C \sqsubseteq D \in \mathcal{T}$. Hence, for each node i on a path of length l in the completion graph, we get up to $|\mathsf{PossDeg}(\mathcal{K})| \cdot |\mathcal{T}|$ different alternatives. This makes up to $l^{|\mathsf{PossDeg}(\mathcal{K})| \cdot |\mathcal{T}|}$ distinct cases that need to be considered during the completion process in the worst-case. It is further known [1] that when supporting GCIs, the maximal path length in the completion graph can be limited only by an upper bound that is (itself) exponential in the size of the input ABox $\mathcal{A}$ (using a technique called *blocking*). This gives a doubly-exponential runtime for tableau-based methods in the worst case, such that without clever implementation techniques and good heuristics for guiding the backtracking search, a tableau-based reasoner integration GCIs as described above might not be usable even in rather small practical scenarios.

In this paper, we present a method that does not suffer from this problem, i.e. no non-deterministic choice-points are introduced.

6 Related Work

Our method $\mathbf{FixIt}(\mathbb{ALC})$ generalizes the principle (i.e. a type elimination process) underlying the top-down variant of the $\mathcal{KBDD}$ procedure proposed in [10] for the modal logic $\mathbf{K}$ to the (more expressive) FDL $\mathbb{ALC}$. Further, our method integrates (fuzzy) ABoxes and TBoxes in the inference process both of which are *not* dealt with in $\mathcal{KBDD}$.

Inference Algorithms for FDLs and Reasoning with GCIs. So far, reasoning in Fuzzy DLs has been mostly based on tableau-methods (e.g., [17, 16, 7, 15]). Most of these methods do not support reasoning with general terminologies as it is possible with $\mathbf{FixIt}(\mathbb{ALC})$. The first method ever to integrate GCIs into FDL reasoning is [16]. A very similar approach is presented in [7] for the fuzzy variant of a more expressive DL, namely $\mathcal{SHI}$. Very recently, [20] proposed a novel and elegant method for reasoning with GCIs (under a more general semantics than here) which is inspired by earlier works on tableau-based reasoning in multi-valued logics. The method combines a tableau-construction procedure with a Mixed Integer Linear Programming (MILP) solver that serves as an oracle to the FDL tableau procedure. To the best of our knowledge there is no other approach to deal with GCIs in FDLs available at present. $\mathbf{FixIt}(\mathbb{ALC})$ therefore represents an interesting enrichment of inference calculi toolbox for FDLs, since no non-determinism is introduced by considering GCIs. A similar effect is achieved in [20] by the substantial modification of a standard tableau-based method and an extension with an MILP oracle: the tableau-expansion process does not become non-deterministic by introducing GCIs. However, depending on the solution techniques applied inside the MILP solver, non-determinism might simply be shifted from the tableau-construction process into the MILP oracle. In such cases, respective computational inefficiencies would then simply be hidden in then MILP oracle, but not actually resolved. A very similar approach that is not fixed to a specific semantics is presented in [3].

Further, [18] demonstrates how to use inference procedures for *classical* DLs to perform reasoning in (some) FDLs. This allows to use algorithms that have been developed for classical DLs in FDL reasoning (for some FDLs) in an indirect way. Please note that the $\mathcal{KBDD}$ procedure can not be used in such an indirect way to perform $\mathbb{ALC}$ reasoning, since both TBoxes and ABoxes are not supported.

[4, 5] consider a fuzzy version of $\mathcal{ALC}$ using arbitrary continuous t-norms (and the corresponding residuated implications) to define the semantics of the concept constructors and proposes a method for deciding $\emptyset \models \langle C \sqsubseteq D \geq 1 \rangle$ and the satisfiability of $\langle C \sqsubseteq D \geq 1 \rangle$ by mapping to a (decidable) propositional fuzzy logic. The generated propositional problems can be exponentially bigger than the FDL input problem. Although, the semantics considered in [4, 5] is more general than here (but differs for universal role restrictions), the proposed decision procedures cover more limited reasoning tasks, i.e. no background knowledge $\mathcal{K}$ is considered.

7 Conclusions and Future Work

We presented a novel procedure $\mathbf{FixIt}(\mathbb{ALC})$ for deciding knowledge base (KB) satisfiability in the FDL $\mathbb{ALC}$, introducing a *new class* of inference procedures into FDL reasoning. Besides the tableau-based methods [16, 7, 20, 3], it is the only (and the first non-tableau-based) approach to integrate general terminologies in FDL reasoning that we are aware of.

Additionally, we clarified the worst-case complexity of the reasoning problem that is solved by the algorithm and showed that deciding the satisfiability of a KB in $\mathbb{ALC}$ is an EXPTIME-complete problem. A discussion of a (straigthforward) implementation of the algorithm based on explicit representations of types shows that our algorithm can be implemented in a way that is worst-case optimal wrt. its runtime.

The main research questions that we want to address next are as follows: we will study means of implicit representation of sets of fuzzy types known from Symbolic Model Checking [8], in particular their implementation by means of Ordered Binary Decision Diagrams (OBDDs) [2] similar to [10], therefore addressing the main obstacle to apply the procedure in practice. A major question concerning optimization is clearly how to implement the final test of the algorithm efficiently, e.g. by heuristic search using the information in the ABox effectively to find the required mapping. The integration of optimizations such as full vs. lean representations or particle vs. types as discussed in [10] should be straightforward. We want to evaluate the effectiveness of the method by an implementation and comparison to tableau-based systems for FDLs. Moreover, we believe that it is interesting to study a bottom-up variant of $\mathcal{KBDD}$ in the context of FDLs too, and to check if the integration of ABoxes can be done more efficiently in such a variant. Finally, we would like to see to what extend the method can cover other semantics for FDLs (e.g. other t-norms) and extended constructs, such as fuzzy modifiers and concrete domains.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge

University Press, 2003.

2. R. E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.
3. V. Haarslev, H. Pai, and N. Shiri. Uncertainty Reasoning for Ontologies with General TBoxes in Description Logic. In P. C. G. Costa, C. D'Amato, N. Fanizzi, K. B. Laskey, K. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, editors, *Uncertainty Reasoning for the Semantic Web I*, LNAI. Springer, 2008.
4. P. Hájek. Making fuzzy description logic more general. *Fuzzy Sets and Systems*, 154(1):1–15, 2005.
5. P. Hájek. *What does Mathematical Fuzzy Logic offer to Description Logic?*, chapter in Fuzzy Logic and the Semantic Web. Capturing Intelligence. Elsevier, 2006.
6. U. Keller and S. Heymans. On Fixpoint-based Decision Procedures for Fuzzy Description Logics I. Technical Report STI TR 2008-08-07, Semantic Technology Institute (STI), University of Innsbruck, August 2008. Available for download at: http://www.uwekeller.net/publications.html.
7. Y. Li, B. Xu, J. Lu, and D. Kang. Discrete Tableau Algorithms for $\mathcal{FSHI}$. In *Proceedings of the International Workshop on Description Logics (DL)*, 2006.
8. K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.
9. C. Meghini, F. Sebastiani, and U. Straccia. A model of multimedia information retrieval. *Journal of the ACM*, 48(5):909–970, 2001.
10. G. Pan, U. Sattler, and M. Y. Vardi. BDD-based decision procedures for the modal logic K. *Journal of Applied Non-Classical Logics*, 16(1-2):169–208, 2006.
11. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Candidate Recommendation 18 August 2003, W3C, 2003.
12. V. R. Pratt. A Near-Optimal Method for Reasoning about Action. *J. Comput. Syst. Sci.*, 20(2):231–254, 1980.
13. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *In Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI 1991)*, pages 466–471, 1991.
14. M. Schmidt-Schauß and G. Smolka. Attributive Concept Descriptions with Complements. *Artif. Intell.*, 48(1):1–26, 1991.
15. G. Stoilos, G. B. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks. Reasoning with very expressive fuzzy description logics. *J. Artif. Intell. Res. (JAIR)*, 30:273–320, 2007.
16. G. Stoilos, U. Straccia, G. Stamou, and J. Pan. General Concept Inclusions in Fuzzy Description Logics. In *Proceedings of the 17th Eureopean Conference on Artificial Intelligence (ECAI-06)*, pages 457–461. IOS Press, 2006.
17. U. Straccia. Reasoning within Fuzzy Description Logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.
18. U. Straccia. Transforming Fuzzy Description Logics into Classical Description Logics. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04)*, number 3229 in Lecture Notes in Computer Science, pages 385–399, Lisbon, Portugal, 2004. Springer Verlag.
19. U. Straccia. A Fuzzy Description Logic for the Semantic Web. In E. Sanchez, editor, *Fuzzy Logic and the Semantic Web*, Capturing Intelligence, chapter 4, pages 73–90. Elsevier, 2006.
20. U. Straccia and F. Bobillo. Mixed integer programming, general concept inclusions and fuzzy description logics. *Mathware & Soft Computing*, 14(3):247–259, 2007.
21. S. Tobies. *Complexity results and practical algorithms for logics in Knowledge Representation*. Phd thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.