Reasoning in $\mathcal{ALC}$ with Fuzzy Concrete Domains

Dorian Merz[3], Rafael Peñaloza[1,2⋆], and Anni-Yasmin Turhan[1⋆⋆]

[1] Institute for Theoretical Computer Science, TU Dresden
[2] Center for Advancing Electronics Dresden
[3] Department of Computer Science, University of Erlangen-Nuremberg, Erlangen

Abstract. In the context of Description Logics (DLs) concrete domains allow to model concepts and facts by the use of concrete values and predicates between them. For reasoning in the DL $\mathcal{ALC}$ with general TBoxes concrete domains may cause undecidability. Under certain restrictions of the concrete domains decidability can be regained. Typically, the concrete domain predicates are crisp, which is a limitation for some applications. In this paper we investigate crisp $\mathcal{ALC}$ in combination with fuzzy concrete domains for general TBoxes, devise conditions for decidability, and give a tableau-based reasoning algorithm.

1 Introduction

Concrete domains were introduced in [2] as an extension to DLs, which allows to model DL concepts based on objects that come from a specified, i.e. concrete, domain and by a set of predicates on that domain, which constrain the set of objects. For example, the natural numbers could be used as a concrete domain to model sizes, or regions together with the RCC relations can be used to model geo-spatial domains.

In order to allow for reasoning a concrete domain $\mathcal{D}$ needs to satisfy some conditions. A concrete domain is called *admissible*, if it contains a predicate for domain membership, the set of predicates is finite and closed under negation, and testing for finite conjunctions of predicates is decidable. In [2] these conditions and a tableaux-based reasoning algorithm for testing concept satisfiability w. r. t. terminologies were given. Concept satisfiability w. r. t. general TBoxes easily becomes undecidable for admissible concrete domains [6]. In [7] Lutz and Miličić give a condition for concrete domains under which decidability can be regained. Essentially, these condition of ω-admissibility ensures that a model for all constraints expressed in the DL knowledge base can be constructed from locally consistent parts.

In this paper we consider fuzzy concrete domains (CDs), where objects from the concrete domain can be related to one another to some degree. This allows for a more fine-grained modelling for vague information as, for instance, in situation recognition in context-aware systems or even to model fuzzy spatial relations for

⋆ Partially supported by DFG within the Cluster of Excellence 'cfAED'
⋆⋆ Partially supported by DFG in the CRC 912 HAEC

image recognition. The combination of DLs and fuzzy concrete domains has been investigated already in a number of settings [11, 3, 9, 8]. However, fuzzy DLs can easily turn out to be undecidable [4]. In our approach, we consider a crisp DL language, with a fuzzy concrete domain. Since our underlying DL is crisp, while the concrete domain is not, the fuzzy values from the fuzzy concrete domain need to be discretized at some point. A natural question is whether the fuzzy CD can be (easily) encoded in a crisp one. In principle this can be done, however, the approach in [7] uses relational networks to represent a set of constraints imposed on the concrete domain objects. The predicates used in relational networks are required to be jointly exhaustive and pairwise disjoint. In a fuzzy setting, where all tuples of concrete domain objects are related to each other via *all* predicates (possibly by degree zero), this is no longer a natural requirement. Moreover, a translation of the fuzzy constraints to crisp ones can lead to an exponential blow-up of the knowledge base as shown in Section 4.

This finding motivates our direct reasoning algorithm for $\mathcal{ALC}$ (with functional roles) and fuzzy concrete domains, since it allows for a more succinct representation of the TBox. To this end we transfer the notion of ω-admissibility to fuzzy concrete domains and give a tableaux-based reasoning procedure for concept satisfiability in the presence of general TBoxes for the new DL $\mathcal{ALC}(\mathbb{D})$ in Section 3. We show soundness, completeness and termination for our procedure. For the full detail of the proofs, we refer the reader to [10].

We give the definition of the basic notions of DLs, (fuzzy) concrete domains and the DL $\mathcal{ALC}(\mathbb{D})$ in Section 2. In Section 3 we devise a tableau algorithm for $\mathcal{ALC}(\mathbb{D})$ with ω-admissible concrete domains. Afterwards we investigate the translation-based approach to handle fuzzy concrete domains by crisp ones in Section 4. We end the paper with conclusions and considerations for future work.

2 Preliminaries

We give only a short introduction to the basic notions of DLs—for a more thorough presentation see [1]. Starting from countable and disjoint sets N_C of concept names and N_R of role names, *concept constructors* are used to build complex concepts. In the DL $\mathcal{ALC}$ complex concepts are formed using the concept constructors listed in Table 1.

The semantics of this logic is given by means of interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a pair consisting of an *interpretation domain* $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations on $\Delta^{\mathcal{I}}$. This function is extended to complex $\mathcal{ALC}$-concepts as shown in the last column of Table 1. As usual in DLs, we use $\bot$ to denote any contradictory concept (e.g. $A \sqcap \neg A$) and $\top$ to denote a tautology ($A \sqcup \neg A$).

Concepts are related to each other by *general concept inclusions* (GCIs), which are statements of the form $C \sqsubseteq D$. The interpretation $\mathcal{I}$ *satisfies* the GCI $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A finite set of GCIs is called a *TBox* $\mathcal{T}$. If a TBox $\mathcal{T}$ contains only GCIs, with concept names as left-hand sides, each concept name appears at most once on the left-hand side of a GCI and the concept names in

Table 1. Syntax and semantics of $\mathcal{ALC}$-concepts.

g

Constructor	**Syntax**	**Semantics**
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
negation	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{d \mid \exists e \in \Delta^{\mathcal{I}}.(d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$(\forall r.C)^{\mathcal{I}} = \{d \mid \forall e \in \Delta^{\mathcal{I}}.(d,e) \in r^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$

the left-hand sides of GCIs do neither directly nor indirectly refer to themselves, then the TBox $\mathcal{T}$ is called a *terminology*. An interpretation is a *model* of a TBox $\mathcal{T}$, if it satisfies each GCI in $\mathcal{T}$.

We consider here the reasoning task of testing satisfiability of concepts with respect to the TBox. Given the concept C and a TBox $\mathcal{T}$, C is *satisfiable* w. r. t. $\mathcal{T}$ iff $\mathcal{T}$ has a model $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

2.1 Concrete Domains

We extend the approach of Lutz and Miličić in [7] to the fuzzy setting and thus adopt their way of introducing concrete domains. They use constraint systems as concrete domains that have binary predicates which are interpreted as jointly exhaustive and pairwise disjoint (JEPD) relations. This does not limit the expressiveness of the concrete domain, since any concrete domain with a finite set of predicates can be translated into one with binary JEPD relations, e.g. see [10]. Before introducing constraint systems, we introduce the class of structures they describe.

Definition 1. *Let V be a countably infinite set of variables and Rel a finite set of binary relations. A Rel-*constraint *is a tuple of the form (t, R), where t is a pair over V and $R \in Rel$. A Rel-*network *N is a (possibly infinite) set of Rel-constraints. For a given Rel-network N, the set of its variables is denoted by V_N and the set of its relations by Rel_N. A Rel-network N is* in normal form*, if for all $x, y \in V_N$, there is exactly one constraint $((x, y), R) \in N$.*

Let τ be a mapping from variables to variables, then τ is extended to pairs by $\tau((v, w)) = (\tau(v), \tau(w))$, to constraints by $\tau((t, R)) = (\tau(t), R)$, and to Rel-networks by $\tau(N) = \{\tau(c) \mid c \in N\}$. A Rel-network N' in normal form is a model *of network N, if there is a total mapping $\tau : V_N \to V_{N'}$ such that $\tau(N) \subseteq N'$.*

Intuitively, a constraint system defines a set of Rel-networks that are satisfiable.

Definition 2. *A* constraint system *$\mathbf{D} = (V, Rel, \mathcal{M})$ is a tuple consisting of the sets of variables V, relations Rel and $\mathcal{M}$, a* set of models of *$\mathbf{D}$, which are complete Rel-networks. A Rel-network N is* satisfiable *in $\mathbf{D}$, if there is a model*

$M \in \mathcal{M}$ and a total mapping $\tau : V_N \to V_M$ from the variables of N to those of M, such that $\tau(N) \subseteq M$.

The notion of ω-admissible constraint systems was introduced in [7]. We refer the reader to this paper for the definition of this notion and only give its variant for the case of fuzzy concrete domains here.

2.2 Fuzzy Concrete Domains

While in the classical notion of concrete domains a predicate for elements holds completely or not at all, fuzzy concrete domains can express that a predicate holds for elements *to some extent*, i.e., with a membership degree from the real unit interval. The requirement to allow for a tuple of variables to be related exclusively via a single relation is not well-defined for fuzzy concrete domains, since variables are always related via all relations of the same arity—possibly only by degree 0. For that reason, we drop the requirement of JEPD relations in the fuzzy setting.

It is not hard to show that fuzzy relations of arbitrary arity can be represented by binary ones, see [10]. Thus relations of higher arity can be handled by our approach, but for the ease of presentation, we only use binary relations here. To allow for a general notion of fuzzy constraints, we use membership degree *sets* defined over a domain $\underline{\mathbf{1}} \subseteq [0, 1]$. We consider a class of *membership degree sets* such that $\underline{\mathbf{1}}$ is a membership degree set and for every two membership degree sets σ, σ', (i) σ has a finite representation, and (ii) $\sigma \cap \sigma'$ and $\underline{\mathbf{1}} \setminus \sigma$ are membership degree sets, too.

Definition 3. *Let V and Rel be as before and $\underline{\mathbf{1}} \subseteq [0, 1]$. A* fuzzy Rel-constraint *is a triple (t, R, σ) with $t \in V^2$, $R \in Rel$, and $\sigma \subseteq \underline{\mathbf{1}}$. A* fuzzy Rel-network $\mathbb{N}$ *is a set of fuzzy Rel-constraints. For $\mathbb{N}$ the set of its variables is indicated by $V_{\mathbb{N}}$ and the one for its relations by $Rel_{\mathbb{N}}$.*

A fuzzy constraint system $\mathbb{D} = (V, Rel, \underline{\mathbf{1}}, \mathfrak{M})$ *consists of the sets of variables V, of relations Rel, and of models $\mathfrak{M}$, a* set of fuzzy Rel-networks.

Intuitively, a fuzzy concrete domain represents a set of Rel-networks that are satisfiable in a fuzzy constraint system $\mathbb{D}$.

Definition 4. *Let $\mathbb{N}$ be a fuzzy Rel-network. An* interpretation *of $\mathbb{N}$ is a function $\mathbb{I} \colon V^2 \times Rel \to \underline{\mathbf{1}}$ that maps pairs of variables and relations to a fuzzy degree. An interpretation $\mathbb{I}$ satisfies a constraint (t, R, σ) if $\mathbb{I}(t, R) \in \sigma$. If $\mathbb{I}$ satisfies all constraints in a fuzzy Rel-network $\mathbb{N}$, then $\mathbb{I}$ satisfies $\mathbb{N}$.*

$\mathbb{N}$ *is* satisfiable *in a fuzzy constraint system $\mathbb{D} = (V, Rel, \underline{\mathbf{1}}, \mathfrak{M})$, if there exists a model $\mathbb{M} \in \mathfrak{M}$, a mapping $\tau : V_{\mathbb{N}} \to V_{\mathbb{M}}$ and an interpretation that satisfies $\mathbb{M}$ and $\tau(\mathbb{N})$.*

The idea is that the interpretation $\mathbb{I}$ assigns to each relation R a membership degree function $\mu_R : V^2 \to \underline{\mathbf{1}}$ such that $\mu_R(t) = d$, if $\mathbb{I}(t, R) = d$. In case the elements in t are not related via R, the membership degree assigned is 0.

A fuzzy *Rel*-network is in *normal form*, if it contains exactly one fuzzy constraint for each pair of variables and relation $R \in \mathit{Rel}$. It is shown in [10] that every fuzzy *Rel*-network $\mathbb{N}$ can be transformed into a normalized one that is satisfied by the same interpretations. Essentially, the two constraints (t, R, σ) and (t, R, σ') can be equivalently replaced by the constraint $(t, R, \sigma \cap \sigma')$, which is well defined, since the class of membership degree sets is closed under intersection.

A fuzzy *Rel*-network $\mathbb{N}$ contains a *Rel-clash*, if for a relation $R \in \mathit{Rel}$ and a tuple t there is a subset of *Rel*-constraints $\{(t, R, \sigma_i) \mid i \in I\} \subseteq \mathbb{N}$, such that $\bigcap_{i \in I} \sigma_i = \emptyset$, with an arbitrary index set I. In other words, this fuzzy *Rel*-network contains a clash iff after transforming it into normal form, it contains a constraint of the form $(t, R, \emptyset)$. Otherwise it is *clash-free*.

It is well-known that extending $\mathcal{ALC}$ with concrete domains leads to undecidability of reasoning w.r.t. TBoxes. To regain decidability of reasoning in the presence of TBoxes, conditions need to be imposed on the concrete domain or on a constraint system, respectively. In the crisp case, the concrete domain is required to be ω-admissible by Lutz and Miličić in [7]. We transfer this condition now to the case of fuzzy constraint systems.

Definition 5. *Given a fuzzy constraint system* $\mathbb{D} = (V, \mathit{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$. $\mathbb{D}$ *has the*

- patchwork property *if for two finite, satisfiable fuzzy Rel-networks* $\mathbb{N}_1$ *and* $\mathbb{N}_2$ *holds: if* $\mathbb{N}_1 \cup \mathbb{N}_2$ *is clash-free, then* $\mathbb{N}_1 \cup \mathbb{N}_2$ *is satisfiable in* $\mathbb{D}$.
- compactness property *if it holds that any infinite fuzzy Rel-network* $\mathbb{N}$ *in normal form is satisfiable iff for all finite* $U \subseteq V$ *the fuzzy Rel-network* $\mathbb{N}_U = \{((x, y), R, \sigma) \in \mathbb{N} \mid x, y \in U\}$ *is satisfiable.*

$\mathbb{D}$ *is* ω-admissible *if (1) satisfiability of finite fuzzy Rel-networks in* $\mathbb{D}$ *is decidable, (2)* $\mathbb{D}$ *has the patchwork property, and (3)* $\mathbb{D}$ *has the compactness property.*

The condition of ω-admissibility ensures decidability of reasoning when combining $\mathcal{ALC}$ and fuzzy constraint systems.

2.3 A DL with Fuzzy Concrete Domains: $\mathcal{ALC}(\mathbb{D})$

To define the DL $\mathcal{ALC}(\mathbb{D})$ we need to introduce *features*, which are functional roles. Let N_{aF} be an infinite countable set of *abstract feature names* and N_{cF} be an infinite countable set of *concrete feature names* and $N_{aF} \cap N_{cF} = \emptyset$. A *feature path* P is either a concrete feature f or a pair of an abstract and a concrete feature: $P = a\ f$ with $a \in N_{aF}$ and $f \in N_{cF}$.

Definition 6. *Let* $\mathbb{D} = (V, \mathit{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ *be a fuzzy constraint system,* r *a role in* $N_R \cup N_{aF}$, $R \in \mathit{Rel}$, *and* $\sigma \subseteq \underline{\mathbf{1}}$. *Complex* $\mathcal{ALC}(\mathbb{D})$*-concepts are formed using the concept constructors of* $\mathcal{ALC}$ *listed in Table 1, where in existential or value restrictions abstract features can be used instead of roles. Additionally,* $\mathcal{ALC}(\mathbb{D})$ *allows for* fuzzy constraint restrictions, *which are expressions of the form* $\exists(P_1, P_2, R, \sigma)$ *or* $\forall(P_1, P_2, R, \sigma)$, *where* $R \in \mathit{Rel}$, *and* $\sigma \subseteq \underline{\mathbf{1}}$ *and* P_i *are feature paths.*

For the semantics of $\mathcal{ALC}(\mathbb{D})$-concepts, we need to extend the notion of an interpretation to fuzzy constraint restrictions and thus accommodate *Rel*-networks.

Definition 7. *An* interpretation *is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mathbb{N}_{\mathcal{I}})$ consisting of a domain $\Delta^{\mathcal{I}}$, a mapping $\cdot^{\mathcal{I}}$, and a fuzzy Rel-network in normal form $\mathbb{N}_{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ maps names from $N_C \cup N_R$ as for $\mathcal{ALC}$; abstract features $a \in N_{aF}$ are interpreted as partial functions over $\Delta^{\mathcal{I}}$, and concrete features $f \in N_{cF}$ are partial functions from $\Delta^{\mathcal{I}}$ to $\mathbb{N}_{\mathcal{I}}$. The interpretation of a feature path $P = a\ f$ is the function that maps $d \in \Delta^{\mathcal{I}}$ to $P(d)^{\mathcal{I}} = f^{\mathcal{I}}(a^{\mathcal{I}}(d))$, when this is well-defined. The semantics of the new concept constructors are:*

$$\begin{aligned}\big(\exists(P_1, P_2, R, \sigma)\big)^{\mathcal{I}} &= \big\{d \in \Delta^{\mathcal{I}} \mid \exists v, w \in V_{\mathbb{N}_{\mathcal{I}}}, \exists \sigma' \subseteq \underline{\mathbf{1}} : P_1^{\mathcal{I}}(d) = v \wedge \\ &\qquad P_2^{\mathcal{I}}(d) = w \wedge (v, w, R, \sigma') \in \mathbb{N}_{\mathcal{I}} \wedge \sigma' \subseteq \sigma\big\} \\ \big(\forall(P_1, P_2, R, \sigma)\big)^{\mathcal{I}} &= \big\{d \in \Delta^{\mathcal{I}} \mid \forall v, w \in V_{\mathbb{N}_{\mathcal{I}}}, \forall \sigma' \subseteq \underline{\mathbf{1}} : \big(P_1^{\mathcal{I}}(d) = v \wedge \\ &\qquad P_2^{\mathcal{I}}(d) = w \wedge (v, w, R, \sigma') \in \mathbb{N}_{\mathcal{I}}\big) \implies \sigma' \subseteq \sigma\big\}.\end{aligned}$$

The classical DL $\mathcal{ALC}(\mathcal{D})$ is a special case of $\mathcal{ALC}(\mathbb{D})$, where $\sigma = \{0, 1\}$ and only the constraint restrictions with $\sigma = \{1\}$ are mentioned.

Let $r \in N_R \cup N_{aF}$. An $\mathcal{ALC}(\mathbb{D})$-concept is in *negation normal form* (NNF), if negation only appears in front of concept names. It is easy to see that every $\mathcal{ALC}(\mathbb{D})$-concept can be transformed into NNF by exhaustive application of the following rules.

$$\begin{aligned}\neg\neg C &\to C & \neg(C \sqcap D) &\to (\neg C \sqcup \neg D) \\ \neg(\exists r.C) &\to (\forall r.\neg C) & \neg(C \sqcup D) &\to (\neg C \sqcap \neg D) \\ \neg(\forall r.C) &\to (\exists r.\neg C) & \neg(\exists(P_1, P_2, R, \sigma)) &\to (\forall(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma)) \\ && \neg(\forall(P_1, P_2, R, \sigma)) &\to (\exists(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma))\end{aligned}$$

3 A Tableau Algorithm for Concept Satisfiability

We show that satisfiability of $\mathcal{ALC}(\mathbb{D})$-concepts w.r.t. $\mathcal{ALC}(\mathbb{D})$-TBoxes is decidable for any ω-admissible fuzzy constraint system $\mathbb{D}$ by describing a tableau-based algorithm for this problem. For the rest of this section we consider a fixed concept C in NNF and a TBox $\mathcal{T}$ containing exactly one GCI $\top \sqsubseteq C_{\mathcal{T}}$ with normalized *Rel*-networks. These assumptions are w.l.o.g., since every GCI $D \sqsubseteq E$ can be equivalently rewritten as $\top \sqsubseteq \neg D \sqcup E$, and every concept can be transformed into NNF in linear time using the rules introduced above.

The algorithm keeps as data structure a *completion system* $\mathcal{S} = (\mathfrak{T}, \mathbb{N}, \Sigma)$, where $\mathbb{N}$ is a finite fuzzy *Rel*-network, Σ is a finite set of subsets of $\underline{\mathbf{1}}$ that describes the membership degrees relevant for reasoning, and $\mathfrak{T}$ is a labeled tree $\mathfrak{T} = (V, E, \mathcal{L})$ such that V is partitioned into two sets V_A and V_C, $E \subseteq V_A \times V$ and $\mathcal{L}$ labels every node $v \in V_A$ with a set of concepts $\mathcal{L}(v) \subseteq \mathsf{sub}(C) \cup \mathsf{sub}(C_{\mathcal{T}})$,[4]

[4] Here $\mathsf{sub}(C)$ denotes the set of subconcepts of a concept C, consider e.g. [1]

Table 2. Tableau rules for $\mathcal{ALC}(\mathbb{D})$

$\mathbf{R}_{\sqcap}$	if $D_1 \sqcap D_2 \in \mathcal{L}(v)$ and $\{D_1, D_2\} \nsubseteq \mathcal{L}(v)$, then add D_1, D_2 to $\mathcal{L}(v)$
$\mathbf{R}_{\sqcup}$	if $D_1 \sqcup D_2 \in \mathcal{L}(v)$ and $\{D_1, D_2\} \cap \mathcal{L}(v) \neq \emptyset$, then add D_1 or D_2 to $\mathcal{L}(v)$
$\mathbf{R}_{\exists}$	if $\exists r.D \in \mathcal{L}(v)$, v is not blocked, and there is no r-successor w of v such that $D \in \mathcal{L}(w)$, then **extend** $\mathfrak{T}$ with a fresh r-successor x of v and add D to $\mathcal{L}(x)$
$\mathbf{R}_{\forall}$	if $\forall r.D \in \mathcal{L}(v)$ and there is an r-successor w of v such that $D \notin \mathcal{L}(w)$, then add D to $\mathcal{L}(w)$
$\mathbf{R}_{\text{ə}}$	if $\exists(P_1, P_2, R, \sigma) \in \mathcal{L}(v)$, v is not blocked, and there are no $c_1, c_2 \in V_C, \sigma' \in \Sigma$ with $P_i(v) = c_i, i \in \{1,2\}$, $(c_1, c_2, R, \sigma') \in \mathbb{N}$ and $\sigma' \subseteq \sigma$, then **extend** $\mathfrak{T}$ with fresh P_i-successors x_i of $v, i \in \{1,2\}$ and add (x_1, x_2, R, σ) to $\mathbb{N}$ and σ to Σ
$\mathbf{R}_{\text{ɐ}}$	if $\forall(P_1, P_2, R, \sigma) \in \mathcal{L}(v)$ and there are $c_1, c_2 \in V_C, \sigma' \in \Sigma$ with $P_i(v) = c_i$, $i \in \{1,2\}$ and $(c_1, c_2, R, \sigma') \notin \mathbb{N}$ for all $\sigma' \subseteq \sigma$, then add (c_1, c_2, R, σ) to $\mathbb{N}$ and σ to Σ

every edge $(v, w) \in V_A \times V_A$ with a role name $\mathcal{L}(v, w) \in N_R \cup N_{aF}$, and each edge $(v, c) \in V_A \times V_C$ with a concrete feature $\mathcal{L}(v, c) \in N_{cF}$. $\mathfrak{T}$ is called a *tableau tree*, which intuitively describes a (partial) tree-shaped interpretation. The nodes in V_A correspond to the abstract domain elements, and V_C contains concrete domain elements. Each abstract element $x \in V_A$ is labeled with the set of concepts that it satisfies. Similarly, edges are labeled with the role or feature that associates its endpoints. The *Rel*-network $\mathbb{N}$ stores the set of constraints that must be satisfied among the concrete domain elements appearing in $\mathfrak{T}$. For each node $v \in V_A$, we define the *local network*

$$\mathbb{N}(v) := \{((a, b), R, \sigma) \in \mathbb{N} \mid (v, a) \in E \text{ or } (v, b) \in E\};$$

that is, $\mathbb{N}(v)$ contains all the fuzzy *Rel*-constraints that are related to the abstract element v. We say that the local networks of two nodes $v, w \in V_A$ are *isomorphic*, denoted as $\mathbb{N}(v) \sim \mathbb{N}(w)$, if there exists a bijective function $\mu : V_{\mathbb{N}(v)} \to V_{\mathbb{N}(w)}$ such that $\mathbb{N}(w) = \mu(\mathbb{N}(v))$. Finally, the component Σ in a completion system $\mathcal{S} = (\mathfrak{T}, \mathbb{N}, \Sigma)$ keeps track of all relevant sets of fuzzy degrees that may be used for satisfying $\mathbb{N}$.

The completion system is initialized to the tuple $\mathcal{S} = (\mathfrak{T}_0, \emptyset, \{\underline{\mathbf{1}}\})$, where $\mathfrak{T}_0 = (\{v_0\}, \emptyset, \mathcal{L})$ is the tableau tree containing only one node v_0 labeled as $\mathcal{L}(v_0) = \{C, C_{\mathcal{T}}\}$. The idea is to try to build a model for $\mathcal{T}$ that makes the interpretation of C non-empty. Thus, we start with one single domain element, namely v_0, that is considered to belong to this concept C. Since the interpretation must be a model of $\mathcal{T}$, v_0 must also belong to $C_{\mathcal{T}}$.

The completion system is then extended by application of the rules from Table 2. Each rule application extends the system and never removes information from it. Only the rule $\mathbf{R}_{\sqcup}$ allows for a non-deterministic choice, which corresponds to deciding which disjunct is used to satisfy the concept $D_1 \sqcup D_2$. Additionally, the two rules for handling existential restrictions $\mathbf{R}_{\exists}$ and $\mathbf{R}_{\text{ə}}$ have a special pre-condition as they are the only ones that add new nodes to the tree $\mathfrak{T}$. Specifically, these rules are only applicable if the node v is *not blocked*, and their application *extends* $\mathfrak{T}$ with either a new r-successor, for r a role or feature name, or P-successor, for P a feature path.

Since the GCIs in the TBox may contain cycles, termination needs to be ensured by detecting cycles in the construction of the model. This can be done by the well-known blocking technique, which is a detection of repetitions in partially constructed models. In *anywhere blocking* [5] an element v in $\mathfrak{T}$ is blocked, if there is another node w that has been introduced before v and that requires the same conditions in the model as v does—in case of $\mathcal{ALC}(\mathbb{D})$ additionally isomorphism of their local *Rel*-networks is required. In that case, it suffices to use the node w as a template to extend v into a model. Hence, there is no need to explicitly extend v during the execution of the tableau algorithm.

The extension of the tree depends on the kind of roles used. Essentially, the idea is that one or more new nodes are added to the tree in order to satisfy the existential restriction. However, recall that abstract and concrete features are restricted to be functional; that is, if g is a feature, then there is at most one g-successor of any given node v. When extending the tree $\mathfrak{T}$, we need to ensure that this functionality is preserved. If there exists already a g-successor, then it must be reused. Formally, let $\mathfrak{T}$ be a tableau tree. For $r \in N_R$, the *extension* of $\mathfrak{T}$ with a fresh r-successor x of v is the tree $\mathfrak{T}'$ obtained from $\mathfrak{T}$ such that:

- if $r \in N_R$ or $r \in N_{aF}$, but v has no r-successors, then $\mathfrak{T}'$ contains a new abstract node $x \in V_A$ and the edge $(v, x) \in E$ with $\mathcal{L}(x) = \{C_{\mathcal{T}}\}$ and $\mathcal{L}(v, x) = r$;
- otherwise, i.e., if $r \in N_{aF}$ and v has an r-successor w, rename w to x.

Similarly, for a concrete feature f, the *extension* of $\mathfrak{T}$ with a fresh f-successor x is the tree where:

- if v has no f-successors, then $\mathfrak{T}'$ contains a new concrete node $x \in V_C$ and the edge $(v, x) \in E$ with $\mathcal{L}(v, x) = f$;
- otherwise, i.e., if v has an f-successor w, rename w to x.

Given a feature path $P = a\ f$, the extension of $\mathfrak{T}$ with a fresh P-successor of v is obtained by extending $\mathfrak{T}$ with an a-successor x of v, and an f-successor of x. If at some point the completion system is *saturated*, i.e., no tableau rule is applicable to it, then the algorithm decides satisfiability of C by searching for an obvious contradiction, or clash. The completion system $\mathcal{S} = (\mathfrak{T}, \mathsf{N}, \Sigma)$ contains a *clash* if N is unsatisfiable or there exist a node $v \in V_A$ and a concept $D \in \mathsf{sub}(C) \cup \mathsf{sub}(C_{\mathcal{T}})$ such that $\{D, \neg D\} \subseteq \mathcal{L}(v)$. Starting from the initial completion system $(\mathfrak{T}_0, \emptyset, \{\underline{\mathbf{1}}\})$, the algorithm applies the completion rules in any order until a saturated system $\mathcal{S}$ is found. If $\mathcal{S}$ contains a clash, then the algorithm answers that the concept C is *unsatisfiable* w.r.t. $\mathcal{T}$; otherwise, i.e., if there is no clash in $\mathcal{S}$, then C is *satisfiable.* We show that this tableau algorithm is indeed a decision procedure for concept satisfiability, i.e., we show that it is sound, complete, and terminating.

We first show that the algorithm is sound. To show this, we will construct, given a finite completion system $\mathcal{S} = (\mathfrak{T}, \mathbb{N}, \Sigma)$, a model $\mathcal{I}_{\mathcal{S}}$ of $\mathcal{T}$ that satisfies C. The idea is to use $\mathfrak{T}$ as a template for building this model, and when a blocked node is reached, iterate using copies of the blocking node and its successors. A $\mathfrak{T}$-*chain* is a sequence $\chi = \frac{v_1}{w_1} \cdots \frac{v_n}{w_n}$ such that for every $i, 1 \leq i < n$, $v_i, w_i \in V_A$, $(v_i, w_{i+1}) \in E$, and either (i) v_{i+1} is not blocked and $w_{i+1} = v_{i+1}$, or (ii) v_{i+1} is blocked by w_{i+1}. In this case, we say that $\frac{v_n}{w_n}$ is the *tail* of χ, written $\mathsf{tl}(\chi)$. We also express as $f(\chi)$, for a concrete feature f, the concrete element $f(w)$ where $\mathsf{tl}(\chi) = \frac{v}{w}$. We denote as chains the set of all chains in $\mathfrak{T}$ that start with $\frac{v_0}{v_0}$.

Let $\mathcal{I}_{\mathcal{S}} = (\Delta^{\mathcal{I}_{\mathcal{S}}}, \cdot^{\mathcal{I}_{\mathcal{S}}}, \mathbb{N}_{\mathcal{I}_{\mathcal{S}}})$ be the interpretation where $\Delta^{\mathcal{I}_{\mathcal{S}}} = \mathsf{chains}$, for every $A \in N_C$, $A^{\mathcal{I}_{\mathcal{S}}} = \{\chi \mid \mathsf{tl}(\chi) = \frac{v}{w}, A \in \mathcal{L}(v)\}$, and for every role name $r \in N_R$, $r^{\mathcal{I}_{\mathcal{S}}} = \{(\chi, \chi\frac{v'}{w'}) \mid \mathsf{tl}(\chi) = \frac{v}{w}, (v, w') \in E, \mathcal{L}(v, w') = r, v' \in V_A\}$. The network $\mathbb{N}_{\mathcal{I}_{\mathcal{S}}}$ is defined over the variables $V_{\mathcal{I}_{\mathcal{S}}} = \{f(\chi) \mid \chi \in \mathsf{chains}\}$ and contains all constraints

$$((f_1(\chi_1), f_2(\chi_2)), R, \sigma)$$

where for $i \in \{1, 2\}$:

$$\mathsf{tl}(\chi_i) = \frac{v_i}{w_i}, (f_1(w_1), f_2(w_2), R, \sigma) \in \mathbb{N}, v_i \in V_A.$$

Notice that $\mathcal{I}_{\mathcal{S}}$ is infinite, and also contains an infinite fuzzy *Rel*-network $\mathbb{N}_{\mathcal{I}_{\mathcal{S}}}$. However, this network is built using copies of a satisfiable *Rel*-network $\mathbb{N}$. The patchwork property guarantees that each finite union of these copies remains satisfiable, and hence, by compactness, the whole system is satisfiable. It can thus be shown by induction on the structure of the concepts, and using the properties of ω-admissibility that if $\mathcal{S}$ does not contain a clash, then $\mathcal{I}_{\mathcal{S}}$ is a model of $\mathcal{T}$ and $v_0 \in C^{\mathcal{I}_{\mathcal{S}}}$.

Lemma 8. *Let $\mathcal{S}$ be a saturated completion system obtained by application of the tableau rules to $(\mathfrak{T}_0, \emptyset, \{\underline{\mathbf{1}}\})$ where $\mathcal{L}(v_0) = \{C, C_{\mathcal{T}}\}$. If $\mathcal{S}$ contains no clash then C is satisfiable w.r.t. $\mathcal{T}$.*

Suppose now that C is satisfiable w.r.t. $\mathcal{T}$. To prove that the algorithm is complete, we need to show that it can produce a clash-free completion system $\mathcal{S}$. Since C is satisfiable, there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. We use this model to guide the construction of the completion system through rule applications. The idea is to identify, for each node of the tree $\mathfrak{T}$, an element in $\Delta^{\mathcal{I}}$ that will serve as its *pattern.* The root node is associated to an arbitrary element in $C^{\mathcal{I}}$. When the rule requires a non-deterministic choice ($\mathbf{R}_{\sqcup}$) or the insertion of new elements ($\mathbf{R}_{\exists}$, $\mathbf{R}_{\ni}$), the choice is made based on the properties of the associated node from $\mathcal{I}$. Since $\mathcal{I}$ is a model, the completion system built this way is guaranteed to be clash-free. This is shown using a variant of relatively standard proof techniques for tableau algorithms, see [7, 10] for full details.

Lemma 9. *If every saturated completion system obtained by the application of tableau rules to $(\mathfrak{T}_0, \emptyset, \{\underline{\mathbf{1}}\})$ contains a clash, then C is not satisfiable w.r.t. $\mathcal{T}$.*

These two lemmas show that the tableau algorithm is sound and complete. The only remaining issue is to show that it terminates on every input, which is a consequence of the following observations. First, every concrete node in the tree $\mathfrak{T}$ is labeled with a set $\mathcal{L}(v) \subseteq \mathsf{sub}(C) \cup \mathsf{sub}(C_{\mathcal{T}})$. Similarly, every edge is labeled with a role name appearing in C or $C_{\mathcal{T}}$. Since $\mathsf{sub}(C)$ and $\mathsf{sub}(C_{\mathcal{T}})$ are both finite, there are finitely many different such labels. Second, the fuzzy *Rel*-network $\mathbb{N}$ only contains constraints of the form $((c_1, c_2), R, \sigma)$ where R and σ appear explicitly in C or $\mathcal{T}$. Hence, there are finitely many pairs (R, σ) appearing in $\mathbb{N}$. Third, every rule application adds at least one concept to the label of a node, or a constraint to $\mathbb{N}$, but never deletes any previous assertions. Thus, to prove termination it suffices to show that the tree $\mathfrak{T}$ has finitely many nodes.

Notice that new nodes are introduced to the tree $\mathfrak{T}$ only through applications of the rules $\mathbf{R}_{\exists}$ and $\mathbf{R}_{\ni}$. Each application of any of these rules adds at most two abstract and at most two concrete nodes. Thus, the number of successors of any node is bounded linearly by the number of existential restrictions in $\mathsf{sub}(C) \cup \mathsf{sub}(C_{\mathcal{T}})$, which is finite. In other words, $\mathfrak{T}$ has finite branching. As described before, the number of different node labels $\mathcal{L}(v)$ is bounded by the number of sets of subconcepts of C and $C_{\mathcal{T}}$; call this number n_C. Similarly, each local network $\mathbb{N}(v)$ is finite, bounded by the number $n_{\mathbb{N}}$ of combinations of concrete features f, relations R and membership degrees Σ allowed. It thus follows that every path of length greater than $n_C \cdot n_{\mathbb{N}}$ must contain at least one directly blocked node. Since the rules $\mathbf{R}_{\exists}$ and $\mathbf{R}_{\ni}$ are only applicable to nodes that are not blocked, the depth of the tree $\mathfrak{T}$ is also finite. Overall, this implies that $\mathfrak{T}$ must be finite, which yields the following result.

Lemma 10. *The tableau algorithm terminates.*

Summarizing, we showed that our tableau algorithm always terminates, is sound and complete for testing whether a concept C is satisfiable w.r.t. a TBox $\mathcal{T}$.

Theorem 11. *The tableau algorithm is a decision procedure for $\mathcal{ALC}(\mathbb{D})$ concept satisfiability.*

Thus, the problem is decidable. A more fine-grained analysis of the bounds used to prove termination reveals that this algorithm applies exponentially many rules, in the worst case, until the completion system is saturated. At this point, the *Rel*-network $\mathbb{N}$ contains exponentially many constraints and needs to be checked for satisfiability. This satisfiability check for the *Rel*-network is only sufficient for concrete domains that are ω-admissible. Assuming a constant-time oracle for testing $\mathbb{N}$ and since the algorithm is non-deterministic, due to $\mathbf{R}_{\sqcup}$, overall we obtain that concept satisfiability in $\mathcal{ALC}(\mathbb{D})$ is in NExpTime, with an oracle for $\mathbb{D}$. Next, we show that reasoning in $\mathcal{ALC}(\mathbb{D})$ can be reduced to reasoning in $\mathcal{ALC}(\mathcal{D})$ for some, well-chosen (crisp) constraint system $\mathcal{D}$.

4 Translating Fuzzy to Crisp Constraints

The extension of $\mathcal{ALC}$ with fuzzy concrete domains with membership degree sets, which are closed under intersection and negation, is not more expressive than

$\mathcal{ALC}$ with (crisp) concrete domains. To be more precise, for any fuzzy constraint system $\mathbb{D}$ and $\mathcal{ALC}(\mathbb{D})$-TBox $\mathbb{T}$, we can effectively construct a constraint system $\mathbf{D}_\mathbb{T}$ and an $\mathcal{ALC}(\mathbf{D}_\mathbb{T})$-TBox $\mathcal{T}$ that preserves the consequences of $\mathbb{T}$. In this section, we assume that $\mathbb{D}$ is an arbitrary, but fixed, fuzzy constraint system. Given an $\mathcal{ALC}(\mathbb{D})$-TBox $\mathbb{T}$, let $\Sigma_\mathbb{T}$ be the set of all sets $\sigma \subseteq \underline{\mathbf{1}}$ such that σ appears in $\mathbb{T}$, extended with $\underline{\mathbf{1}}$. Since $\mathbb{T}$ is finite, so is $\Sigma_\mathbb{T}$, and its closure under complementation and intersection $\Lambda_\mathbb{T}$. Moreover, the $|\Lambda_\mathbb{T}|$ is bounded exponentially by $\Sigma_\mathbb{T}$ and is in the worst case exponential on the size of $\mathbb{T}$.

Let $\Pi_\mathbb{T}$ be the set of all relation names appearing in $\mathbb{T}$. Obviously, $|\Pi_\mathbb{T}|$ is linear in $|\mathbb{T}|$. Finally, let $R_1, \ldots, R_m$ be an arbitrary, but fixed, enumeration of the elements of $\Pi_\mathbb{T}$. We define the set of binary relations containing every sequence of length m of elements of $\Lambda_\mathbb{T}$ as $\mathsf{Rel} := \{\lambda_1 \cdots \lambda_m \mid \lambda_i \in \Lambda_\mathbb{T}, 1 \leq i \leq m\}$. Clearly, Rel has $|\Lambda_\mathbb{T}|^m$ relation names.

Intuitively, the relation $\lambda_1 \cdots \lambda_m$ is interpreted to include all the pairs (a, b) of elements of the constraint model such that $R_i(a, b) \in \lambda_i$, for all i, $1 \leq i \leq m$. That is, each of these relations describes, in a crisp manner, the degrees to which the pair belongs to all the relevant fuzzy relations. Following this intuition, we denote as $\sigma^{(i)}$ the relation in Rel that has σ in its i-th position, and $\underline{\mathbf{1}}$ in all other positions. It is interpreted as all pairs of individuals that are related via R_i with a degree in σ, regardless of the degrees associated with the other fuzzy relations.

Our translation function ν maps $\mathcal{ALC}(\mathbb{D})$ concepts to $\mathcal{ALC}(\mathbf{D})$ concepts, such that all consequences from $\mathbb{T}$ are preserved by $\nu(\mathbb{T})$. This translation is defined inductively over the structure of concepts. Let C, D be $\mathcal{ALC}(\mathbb{D})$-concepts and $r \in N_R \cup N_{aF}$, $R_i \in \Pi_\mathbb{T}$, $\sigma \in \Lambda_\mathbb{T}$, and P_1, P_2 two feature paths, the translation ν of the fuzzy constraint restrictions is defined by:

$$
\begin{aligned}
\nu(A) &:= A \quad \text{for } A \in N_C \cup \{\top, \bot\}, & \nu(\exists r.C) &:= \exists r.\nu(C),\\
\nu(\neg C) &:= \neg \nu(C), & \nu(\forall r.C) &:= \forall r.\nu(C),\\
\nu(C \sqcap D) &:= \nu(C) \sqcap \nu(D), & \nu(\exists(P_1, P_2, R_i, \sigma)) &:= \exists(P_1, P_2, \sigma^{(i)}),\\
\nu(C \sqcup D) &:= \nu(C) \sqcup \nu(D) \text{ and} & \nu(\forall(P_1, P_2, R_i, \sigma)) &:= \forall(P_1, P_2, \sigma^{(i)}).
\end{aligned}
$$

We define $\nu(\mathbb{T}) := \{\nu(C) \sqsubseteq \nu(D) \mid C \sqsubseteq D \in \mathbb{T}\}$. Obviously, this construction preserves all consequences of the original TBox. Additionally, $|\nu(\mathbb{T})|$ is linear in $|\mathbb{T}|$. The main difference is that the crisp constraint system $\mathbf{D}$ obtained has exponentially many more relation functions than $\mathbb{D}$. This is not problematic for reasoning, since the system of constraints is solved by an external oracle. However, it must be noted that these relations are not JEPD as assumed in [7]. To obtain a constraint system satisfying this condition, a rewriting of each concrete domain restriction into a possibly exponential disjunction of restrictions is needed, which causes a blow-up in $|\mathbb{T}|$.

Observe, that the translation presented depends on the specific sets of degrees σ that appear in $\mathbb{T}$. Indeed, to produce one constraint system that can be used for any arbitrary $\mathcal{ALC}(\mathbb{D})$-TBox, we would need to be able to handle arbitrary subsets of $\underline{\mathbf{1}}$.

5 Conclusions

We introduced the DL $\mathcal{ALC}(\mathbb{D})$ that extends $\mathcal{ALC}$ with fuzzy concrete domain restrictions. These in turn introduce fuzzy relations between elements of the concrete domain, i.e., functions that map tuples of concrete elements to a membership degree in $[0, 1]$. We extended the approach from [7] for regaining decidability of reasoning in the presence of general TBoxes, to the fuzzy setting. The required conditions on the concrete domain are the *patchwork property* and *compactness*, which together yield ω-admissibility. Decidability of concept satisfiability w.r.t. TBoxes is proven by a sound, complete and terminating tableau-based algorithm which builds a finite representation of an infinite tree-like model of the TBox and the concept. We show that this algorithm requires (non-deterministic) exponential time, if the constraint systems can be solved in constant time. Our proofs of correctness depend strongly on the notion of ω-admissibility. Thus, it is an open question whether relaxed conditions would still guarantee decidability.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
2. F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, 1991.
3. F. Bobillo and U. Straccia. fuzzydl: An expressive fuzzy description logic reasoner. In *In Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 923–930. IEEE, 2008.
4. S. Borgwardt and R. Peñaloza. Undecidability of fuzzy description logics. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 232–242. AAAI Press, 2012.
5. B. Glimm, I. Horrocks, and B. Motik. Optimized description logic reasoning via core blocking. In *Proc. of the 5th Int. Joint Conf. on Automated Reasoning (IJCAR-10)*, volume 6173 of *LNCS*, pages 457–471. Springer, 2010.
6. C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logic 2002 (AiML 2002)*, Toulouse, France, 2002.
7. C. Lutz and M. Miličić. A Tableau Algorithm for DLs with Concrete Domains and GCIs. *Journal of Automated Reasoning*, 38(1–3):227–259, 2007.
8. T. Mailis, R. Peñaloza, and A.-Y. Turhan. Conjunctive query answering in finitely-valued fuzzy description logics. In *Proceedings of 8th International Conference Web Reasoning and Rule Systems (RR 2013)*, LNCS. Springer, 2014. To appear.
9. T. P. Mailis, G. Stoilos, N. Simou, G. B. Stamou, and S. D. Kollias. Tractable reasoning with vague knowledge using fuzzy $\mathcal{EL}^{++}$. *Journal of Intelligent Information Systems*, 39(2):399–440, 2012.
10. D. Merz. Decidability of reasoning in $\mathcal{ALC}$ with fuzzy concrete domains. Diplomarbeit, Technische Universität Dresden, 2013. See: `http://lat.inf.tu-dresden.de/research/mas/`.
11. U. Straccia. Description logics with fuzzy concrete domains. In *Proc. of the 21st Conf. in Uncertainty in AI (UAI'05)*, pages 559–567. AUAI Press, 2005.