Technische Universität Dresden

Fakultät Informatik

Institut für Theoretische Informatik

Lehrstuhl für Automatentheorie

Diplomarbeit

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

Autor
Dorian Merz

Betreuer
Dr.-Ing. Anni-Yasmin Turhan
Dr. rer. nat. Rafael Peñaloza

Betreuender Hochschullehrer
Prof. Dr.-Ing. Franz Baader

31.07.2013

vorgelegt von:	Dorian Merz
geboren am:	10.05.1986
Matrikel-Nummer:	3249454
Studiengang:	Informatik
Angestrebter Abschluss:	Diplom

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig erstellt und verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Dresden, den 31.07.2013

Dorian Merz

Zusammenfassung

Diese Diplomarbeit verwendet – soweit bekannt – erstmals mehrstellige *Fuzzy Constraint Systems* als konkrete Domänen (*Concrete Domains*) in $\mathcal{ALC}$. Diese Erweiterung der bekannten *Constraint Systems* beruht auf der Zuweisung von Tupeln und Relationen auf Mengen möglicher Zugehörigkeitsgrade. Das Hauptziel der Arbeit liegt im Beweis der Entscheidbarkeit von Konzepterfüllbarkeit bezüglich genereller TBoxen in der so erweiterten Beschreibungslogik. Zum Erreichen dieser Entscheidbarkeit müssen einige zusätzliche Anforderungen an *Fuzzy Constraint Systems* und $\mathcal{ALC}$ gestellt werden. In einer Arbeit zum Thema $\mathcal{ALC}$ mit *Constraint Systems* stellten Lutz und Miličić Definitionen und einen Beweis vor, die für die vorliegende Arbeit übernommen werden sollen. Für den hier angewandten *fuzzy*-Fall werden diese Ideen an die neuen Definitionen angepasst und erweitert. Nach der detailierten Einführung der Definitionen von *Fuzzy Constraint Systems* und deren Verwendung im Zusammenhang mit $\mathcal{ALC}$ bilden die Vorstellung eines Tableau-Algorithmus zur Entscheidung der Konzepterfüllbarkeit und der Beweis über dessen Korrektheit den Abschluss der Diplomarbeit.

Abstract

This thesis introduces fuzzy constraint systems as concrete domains in $\mathcal{ALC}$. To the best of our knowledge this is the first time that non-unary predicates are used in this context. The extension of crisp constraint systems to fuzzy is performed by addition of sets of membership degrees for tuples in relations. The main goal of the paper is the proof of decidability of concept satisfiability with respect to general TBoxes in that enhanced description logic. For that purpose fuzzy concrete domains and $\mathcal{ALC}$ are restricted in several ways. The approach given in a paper by Lutz and Miličić on $\mathcal{ALC}(\mathcal{D})$ with crisp constraint systems inspired the adaptation and application of ω-admissibility and the overall proving method on fuzzy constraint systems. We present a detailed definition of the fuzzy extension of constraint systems, their integration into $\mathcal{ALC}$, and a tableau algorithm that decides satisfiability of concepts.

Contents

1. Introduction

The representation of human knowledge in a computer-readable manner is a task that has been approached in manifold ways. A large variety of different notions and calculi have been developed for different aspects of that task. Eventually, after a time of development in separate directions two or more of these different conceptuations turn out to yield even greater expressiveness when they are combined. This thesis presents such a combination of the fuzzy set theory and so-called concrete domains, whose outcome will then be used in a third notion: description logics.

The family of *Description Logics* has been established and examined over time as an adequate syntax and semantics for knowledge-representation. It allows for the definition of classes of individuals, so-called *concepts*, and their inter-dependencies. Other approaches to machine-understandable representation of human knowledge include different kinds of logic, from propositional over modal and first-order predicate logics to higher order kinds. The fuzzy set theory, allowing elements to be member of a set by a certain degree ([Zad65]) enhanced the possibility of applying truth values to logical formulas according to a degree of truth that may lie between zero and one. This calculus was introduced under the name of *fuzzy logic*. It has been just a small step, to add this notion to description logics as well ([Yen91]).

Basic description logics like $\mathcal{ALC}$ have limited expressive power. The definiton of more complex relationships between possible or actual individuals in the domain of interest could be achieved with the introduction of *concrete domains*. A concrete domain is a set of things, the domain, together with a set of relation names and a means of telling when elements may or may not belong to the represented relations. Description logics with concrete domains have been examined for several decades now ([BH91, LM04, LM07]).

Inspired by results on the decidability of reasoning in such description logics with well-chosen restrictions on the concrete domains, the main goal of this thesis is to show that a fuzzy extension of concrete domains can be found, that – in combination with $\mathcal{ALC}$ – leads to decidable reasoning problems. To be exact we show restrictions on fuzzy concrete domains that lead to decidability of concept satisfiability with respect to general TBoxes. That aim of decidability has been examined for crisp $\mathcal{ALC}(\mathcal{D})$ before. For the special case of acyclic i.e. unfoldable terminologies, Baader and Hanschke proved that the use of concrete domains leads to decidable concept satisfiability, provided the concrete domain obeys the restriction of being *admissible* ([BH91]). Since unfoldable terminologies are a severe restriction on the expressive power of DLs we would prefer a possibility of reasoning w.r.t. general TBoxes. A paper by Lutz and Miličić enhanced that notion and presented an algorithm that decides concept satisfiability w.r.t. general $\mathcal{ALC}$-TBoxes using an ω-*admissible* concrete domain ([LM07]). Decidable reasoning in *fuzzy* $\mathcal{ALC}(\mathcal{D})$ with *fuzzy* concrete domains has been examined by Straccia. He presented a fuzzy $\mathcal{ALC}$ with fuzzy concrete domain, in which satisfiability of a concept w.r.t. acyclic TBoxes is decidable. This, however, applies under the assumption of only unary relations

in the concrete domain ([Str05]). The notion adopted in the present thesis relies on crisp $\mathcal{ALC}$ combined with fuzzy constraint systems allowing higher arities of relations. Reasoning is considered w.r.t. general TBoxes. It will turn out that the restrictions given by ω-admissibility as introduced in [LM07] can be transcribed to fuzzy concrete domains in a manner that also leads to decidable reasoning procedures in context of fuzzy conrete domains.

The thesis is structured into three main parts. The first illustrates the now common definitorial basis on which $\mathcal{ALC}(\mathcal{D})$ with fuzzy concrete domains founds. Those preliminary subsections are focused on the definitions as needed for the satisfiability decision procedure.[1] First we describe $\mathcal{ALC}$ in its standard definition. An example domain concerning humans illustrates the functionality of the different concept constructors first. Then syntax and semantics of $\mathcal{ALC}$ are formally defined.

Continuing the example we then characterize the advantage of concrete domains before formally defining them as well. This subsection on concrete domains contains some passages on the mentioned earlier approaches. A closer look at the restrictions the original authors have made shall illuminate the ideas behind them. Those restrictions include ω-admissibility of crisp constraint systems which will be transposed to fuzzy constraint systems later on.

The following subsection on fuzzy sets is held quite short, leading directly to their application on constraint systems.

Fuzzy constraint systems as a multi-valued extension of concrete domains are then explained in detail. All notions that were introduced for crisp constraint systems are re-defined for the fuzzy case. We describe the syntactical structure of fuzzy constraints, fuzzy constraint networks, and fuzzy constraint systems. This includes the notion of satisfiability of fuzzy networks which presupposes the existence of interpretations of fuzzy constraints. Such interpretations are new to fuzzy constraint systems and, basically, ensure that relations can be seen as fuzzy membership functions on tuples. After the introduction of a normal form and a completion for constraint networks, some minor results on fuzzy constraint systems are presented. Those result from syntactical restrictions. Two special cases are examined in particular: The reduction of arbitrary-arity relations to binary ones is helpful in the later definition of $\mathcal{ALC}(\mathcal{D})$-syntax. It will be proved that this does not restrict the expressiveness of fuzzy constraint systems. Contrary to that is the special case of finitely many possible degrees of membership. Considering this, any fuzzy constraint system can be transformed to a crisp constraint system by simple renaming or by first finding the set of canonical interpretations.

We continue the transcription of notions used in the crisp case by finding adequate re-definitions of the patchwork-property and the completeness-property. Those together

[1] This results in skipping unnecessary information. T-Norms, Individuals, ABoxes, or the reduction of different reasoning problems to others will not be part of the paper. Interested readers may refer to [BCM+03] or [Baa09].

with decidability of the satisfiability of finite fuzzy constraint networks form ω-admissible fuzzy constraint systems. It is crucial for the presented decision procedure, that the employed fuzzy concrete domain is ω-admissible.

The next section finally extends the description logic $\mathcal{ALC}$ with fuzzy concrete domains. Again a human-related example illustrates the idea behind the newly introduced concept constructors. To lead over to the main goal of the thesis, i.e. the decision procedure, we show that each $\mathcal{ALC}(\mathcal{D})$-concept and TBox can be transferred into equivalents in *negation normal form* and that TBoxes can be reduced to one single axiom.

After those preliminaries, a tableau based algorithm is presented, that computes if a given concept is satisfiable w.r.t. a given general $\mathcal{ALC}(\mathcal{D})$-TBox with ω-admissible fuzzy concrete domains. The correctness of that algorithm is proved in the common steps i.e. termination, soundness and completeness are shown in consecutive order. A large example that depicts a stepwise application of the algorithm and some ideas from its proof of soundness can be found in Appendix A.

The thesis concludes with some thoughts towards possible optimizations of the presented approach, its applicability in real-life contexts and a list of open tasks on this subject.

2. Description Logics - State of the Art

The addition of new properties to 'well-known' structures presupposes that these structures are indeed well known by the reader. To ensure a common ground to build the novelties on, this section gives a set of definitions as used in the literature. These definitions will then be applied and altered or extended to fit the requirements of vague knowledge in concrete domains.

Understanding human knowledge is an unsolvable task for computers unless they receive help by humans. This presupposes machines and humans talking in the same language. To describe a part of the world in a way that makes this description accessible and processible by a PC several languages have developed during time. A special subset amongst those is that of 'Description Logics'.

As all or at least most real-world-modeling languages do, description logics cannot model the whole world but only a special part of it. This part is the *domain of interest (DOI)*. It is a main feature of DL to adopt the *open world assumption* which says that facts we do not know to be true might nevertheless be true. This opposes DLs to database systems in which a fact that is not given in the data base is assumed to be wrong.

The following subsection will depict the creation of an exemplary $\mathcal{ALC}$ terminology, introducing the constructors available in this description logic. Afterwards, a formal definition of $\mathcal{ALC}$ syntax and semantics is given.

2.1. Example Domain

Assume we want to adequately express knowledge concerning humans. Then we just introduce a concept named `Human` and insist on every individual being human to be in that concept. Since this is not very informative or interesting let us add some more concepts. It is quite true, that all humans have a sex which is either male or female. Thus `Male` and `Female` are added. Every human being either male or female means:

1. Any human is male *or* female.
2. A male is not a female.
3. A female is not a male.

Those sentences can directly be translated into $\mathcal{ALC}$-axioms:

$$\texttt{Human} \sqsubseteq \texttt{Male} \sqcup \texttt{Female}$$
$$\texttt{Male} \sqsubseteq \neg\texttt{Female}$$
$$\texttt{Female} \sqsubseteq \neg\texttt{Male}$$

The $\sqsubseteq$-symbol here is read as 'is *sub-concept* of' and means that every individual that is a member of the left-hand-side concept must necessarily also be a member of the right-hand-side concept. Further, $\sqcup$ is the *concept disjunction* and can be read as a union of the

set of (possible) individuals, that is: an individual being member of the disjunction of two concepts is in at least one of both. The *negation*-symbol $\neg$ stands for the complementary set of individuals i.e. an individual of the DOI that is member of the negation of a concept must not be in the original one.

Now, being able to denote an individual's sex, we can define women as female humans and men as male humans:

$$\texttt{Woman} \equiv \texttt{Human} \sqcap \texttt{Female}$$
$$\texttt{Man} \equiv \texttt{Human} \sqcap \texttt{Male}$$

The *concept conjunction* $\sqcap$ denotes the intersection of sets of domain elements i.e. those being in both of the two concepts. The concept equivalence symbol $\equiv$ is an abbreviation for $\sqsubseteq$ and $\sqsupseteq$. In this case it says that every individual that is a woman must be a human and a female and the reverse: every individual that is a human and a female is necessarily also a woman.

If we now would like to introduce children and the fact of some individuals being parents of others, we would need the notion of a role. Roles in description logics are representing binary relations i.e. relationships between two individuals. We want to state:

1. an individual that has a child is called Parent
2. every child of a human is a human

Let *hasChild* be a role representing the obvious, then the following axioms denote the desired facts:

$$\texttt{Parent} \sqsubseteq \exists \mathit{hasChild}.\top$$
$$\texttt{Human} \sqsubseteq \forall \mathit{hasChild}.\texttt{Human}$$

The right-hand-side of the upper line is a so-called *existential restriction* . It represents the set of all things that have at least one child that is some thing. 'Thing' in this case is the top-most concept, coinciding with the full set of domain elements and abbreviated by $\top$. The latter can also be seen as a name for $A \sqcup \neg A$ with an arbitrary concept A. The dual concept $\bot$ then represents 'nothing' or the empty set. In the second line above the *value restriction* $\forall \mathit{hasChild}.\texttt{Human}$ represents all domain elements whose children altogether are human. As desired, the last line expresses that a human has only human children.

If we would now like to know if the parent of a robot can also be a human, we could ask the reasoner if the concept

$$C := \texttt{Human} \sqcap \exists \mathit{hasChild}.\texttt{Robot}$$

of individuals that are human and have children that are robots is satisfiable w.r.t. the given terminology. In other words: Can we find a non-empty domain satisfying all of the

above axioms while also having elements in C? This should, intuitively, not be the case but is, in fact, possible. This is due to the *open world assumption*: we cannot consider missing facts to be wrong i.e. as long as there is no axiom telling that a robot is not a human it might still be true, that there are robots that are humans. Another axiom is needed to fit our intuition:

$$\texttt{Robot} \sqsubseteq \neg\texttt{Human}$$

Provided this last line added to the example the concept C of humans having robotic children is not satisfiable i.e. in our domain of interest there cannot be humans whose children are robots.

2.2. $\mathcal{ALC}$ - Formal Definitions

To formalize the intuitive notions we need a definition of the syntax and semantics of $\mathcal{ALC}$.

Definition 2.2.1 (*Syntax*)

Let N_C and N_R be non-empty disjoint sets of concept names *and* role names, *respectively. The set $\mathcal{C}_{\mathcal{ALC}}$ of $\mathcal{ALC}$-concepts then is defined recursively. Every concept name is an $\mathcal{ALC}$-concept i.e. $N_C \subseteq \mathcal{C}_{\mathcal{ALC}}$. Given $\mathcal{ALC}$-concepts $C, D \in \mathcal{C}_{\mathcal{ALC}}$ and a role $r \in N_R$ the following are concepts:*

1. *the top concept:* $\top$
2. *the bottom concept:* $\bot$
3. *the negated concept:* $\neg C$
4. *the concept conjunction:* $C \sqcap D$
5. *the concept disjunction:* $C \sqcup D$
6. *the existential restriction:* $\exists r.C$
7. *the value restriction:* $\forall r.C$

A general concept inclusion axiom (GCI) *is a construct $C \sqsubseteq D$ for concepts C, D. A general* $\mathcal{ALC}$-TBox *is a finite set of GCIs.*

The concepts that are concept names will be called *basic* concept. The others are *complex* concepts. The semantics of $\mathcal{ALC}$ is defined with the help of interpretations of the given concepts, roles and axioms.

Definition 2.2.2 (*Semantics*)

An interpretation $I = (\Delta_I, \cdot^I)$ consists of a non-empty set Δ_I, the domain *of I and a function $\cdot^I$ that assigns every basic concept a subset of the domain ($A \in N_C : A^I \subseteq \Delta_I$) and every role a binary relation over the domain ($r \in N_R : r^I \subseteq \Delta_I \times \Delta_I$). This mapping is then extended on complex concepts as follows:*

1. $\top^I = \Delta_I$
2. $\bot^I = \emptyset$
3. $(\neg C)^I = \Delta_I \setminus C^I$
4. $(C \sqcap D)^I = C^I \cap D^I$
5. $(C \sqcup D)^I = C^I \cup D^I$
6. $(\exists r.C)^I = \{\delta \in \Delta_I \mid \exists x \in \Delta_I : (\delta, x) \in r^I \wedge x \in C^I\}$
7. $(\forall r.C)^I = \{\delta \in \Delta_I \mid \forall x \in \Delta_I : (\delta, x) \in r^I \Rightarrow x \in C^I\}$

A GCI $C \sqsubseteq D$ is said to be satisfied *by I if $C^I \subseteq D^I$. Furthermore I* satisfies *a TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$. We then call I a* model *for $\mathcal{T}$.*

The way in which GCIs are satisfied by interpretations allows for an abbreviation of $C \sqsubseteq D$ and $C \sqsupseteq D$ to $C \equiv D$ with the obvious extension of I: the interpretation satisfies $C \equiv D$ if $C^I = D^I$.

One of the main advantages of description logics is the possibility of so-called *reasoning*, which is the detection and explicit expression of implicit knowledge.

Reasoning over $\mathcal{ALC}$-TBoxes includes consistency checking of the TBox itself, concept satisfiability, and concept subsumption. We say a TBox is *consistent* if there is an interpretation that satisfies it. Further a concept C is *satisfiable w.r.t. a TBox $\mathcal{T}$* if there exists an interpretation I such that I satisfies $\mathcal{T}$ and $C^I \neq \emptyset$. In other words: Can there be individuals in C while all axioms in $\mathcal{T}$ are satisfied? This is the task we will approach when having integrated fuzzy concrete domains into our version of $\mathcal{ALC}$. For the sake of completeness: a concept C is said to be subsumed by a concept D w.r.t. a TBox $\mathcal{T}$ if every model of $\mathcal{T}$ makes the interpretation of C a subset of that of D i.e. $C \sqsubseteq_\mathcal{T} D \Leftrightarrow C^I \subseteq D^I$ for all models I of $\mathcal{T}$.

With this basic knowledge at hand we are now prepared to extend $\mathcal{ALC}$ with concrete domains. Considering the domain as described in the current subsection to be the 'abstract' domain, a concrete domain is another set of elements equipped with predicates amongst them. Abstract domain elements can then be mapped to concrete domain elements to achieve additional information.

2.3. Concrete Domains - $\mathcal{ALC}(\mathbf{D})$

You may want to extend our example TBox and thus the described domain of interest by numbers i.e. sizes, distances, amounts or similar properties of domain elements and orders or relations amongst them. Information like

1. a *tall* man exceeds 180 centimeters in height

2. every child is younger than his mother

shall be transferred into description logic syntax. We will see, that the first may still be constructed with $\mathcal{ALC}$, while the second shows the limitations of that DL. Consider the concept `HeightOfMoreThan180cm` and a role *hasHeight*. Setting

$$\texttt{TallMan} \sqsubseteq \exists \mathit{hasHeight}.\texttt{HeightOfMoreThan180cm}$$

then yields the desired expression. For the second, however there is no such solution. Try the role *isOlderThan* and set

$$\texttt{Child} \sqsubseteq \exists \mathit{isOlderThan}.\texttt{Mother}$$

This does indeed say, that a child must be younger than *any* mother. In contrast we wanted to say that a child has to be younger than *his* mother. The current definition lacks a means of addressing several item that are connected to the same domain element. This leads to concepts of the form

$$\exists(\boldsymbol{hasAge}, \mathit{hasMother}\, \boldsymbol{hasAge}, <)$$

which shall contain exactly those element whose age is less than that of their mother. Here ***hasAge*** is a *concrete feature* that maps from the DOI to a *concrete domain*.

2.3.1. Constraint Systems

What is a concrete domain, then? There exist several definitions which partly dissent (cf. Appendix B for a comparison of two such definitions). Essentially, a concrete domain is a set of things, i.e. the *domain* itself, together with a set of predicates, i.e. relations on the elements of that domain. Those elements can be considered as numbers or measurable entities that are ordered by the predicates. To avoid the mentioned uncertainty about their definition we now define *constraint systems* which we then use as a concrete domain to extend $\mathcal{ALC}$.

Definition 2.3.1 (*Rel-Constraint, Rel-Network, Constraint System*)

Let V and `Rel` *be countably infinite sets of variables (the domain) and relation names (predicates), respectively. Each relation $R \in$* `Rel` *has an arity denoted as $ar(R)$. A* `Rel`-constraint *is a tuple (t, R) with $t \in V^n$ being an n-tuple over V and $R \in$* `Rel`*, $ar(R) = n$ an n-ary relation from* `Rel`*. A* `Rel`-network *N is a (probably infinite) set of* `Rel`*-constraints. With V_N and $\texttt{Rel}_N$ we denote the sets of variables and relation names that are actually used in constraints in N.*

A constraint system $\boldsymbol{D} = (V, \texttt{Rel}, \mathfrak{M})$ consists of countably infinite sets $V, \texttt{Rel}$ as described above and a set $\mathfrak{M}$ of `Rel`*-networks, the* models *of $\boldsymbol{D}$.*[2]

Intuitively such constraint system fixes a set of `Rel`-networks that are satisfiable in $\mathbf{D}$, where satisfiability is obeying the relations in `Rel` or, to be correct, their interpretations. The models in such constraint system basically reflect these interpretations of relations, i.e. there we can read off how relations interact, in which way elements are allowed to be combined. A model builds a kind of pattern for the relation behavior. A given arbitrary network now only needs to be 'moved under' this pattern and - if it fits - it can be considered satisfying the relations.

Definition 2.3.2 (*Semantics of Constraint Systems*)

A `Rel`*-network N is said to be* satisfiable *in a constraint system $\boldsymbol{D}$ if there exists a model $M \in \mathfrak{M}$ and a mapping $\tau : V_N \to V_M$ from the variables in N to those in M such that $\tau(N) \subseteq M$.*

Here τ is extended on tuples, constraints and networks in the expectable way i.e.

$$\begin{aligned} \tau((v_1, v_2, \ldots, v_n)) &= (\tau(v_1), \tau(v_2), \ldots, \tau(v_n)) \\ \tau((t, R)) &= (\tau(t), R) \\ \tau(N) &= \{\tau(c) \mid c \in N\} \end{aligned}$$

These constraint systems were then used to make connections to $\mathcal{ALC}$-TBoxes.

[2] We use a bold face Latin $\mathbf{D}$ to denote crisp constraint systems contrasting them to the fuzzy version denoted by a calligraphic $\mathcal{D}$

2.3.2. $\mathcal{ALC}(\mathbf{D})$

The literature knows several different approaches to the integration of concrete domains in description logics. In [LM04] Lutz and Miličić allow arbitrary arities for predicates i.e. relations. In a later paper, the same authors restrict them to binary (see [LM07]). This, as we will see later, does not alter the expressiveness of the systems. For his fuzzification of $\mathcal{ALC}$ with concrete domains Straccia even restricted relations to sets i.e. unary predicates (see [Str05]). In preparation on fuzzy concrete domains as this thesis takes into account, we adopt the binary version. The single definitions also differ in the way they 'contact' the concrete domain. The approach that the example already suggested seems quite intuitive, because it uses the well-known $\exists$ and $\forall$ quantifiers in a very similar manner.

Definition 2.3.3 ($\mathcal{ALC}(D)$)

Let $N_C, N_R, N_{cF}, \boldsymbol{D} = (V, \texttt{Rel}, \mathfrak{M})$ *be the sets of* concept names, role names *and* concrete features *and a constraint system. All relations in* `Rel` *are binary i.e.* $ar(R) = 2$ *for all* $R \in$ `Rel`*. Let further* N_R *be the disjoint union of* N_{sR}, N_{aF} *as the sets of* standard roles *and* abstract features*. A* feature *(abstract or concrete) is a functional mapping.* Feature paths *shall, for this overview, consist of a single concrete feature or a concrete feature preceded by an abstract feature i.e.* $P = a\ f$ *or* $P = f$ *for* $a \in N_{aF}, f \in N_{cF}$.[3]

The set of $\mathcal{ALC}(\boldsymbol{D})$*-concepts* $\mathcal{C}_{\mathcal{ALC}(\boldsymbol{D})}$ *is defined recursively*

Every concept name is a concept: $N_C \subseteq \mathcal{C}_{\mathcal{ALC}(\boldsymbol{D})}$*. The complex concepts from Definition 2.2.1 are also concepts in* $\mathcal{ALC}(\boldsymbol{D})$*. Additionally, for* $\mathcal{ALC}(\boldsymbol{D})$*-concepts* $C, D \in \mathcal{C}_{\mathcal{ALC}(\boldsymbol{D})}$*, an arbitrary role* $r \in N_R$*, feature paths* P_1, P_2*, and a relation* $R \in$ `Rel` *the following are* $\mathcal{ALC}(\boldsymbol{D})$*-concepts:*

8. $\exists(P_1, P_2, R)$
9. $\forall(P_1, P_2, R)$

We call those concrete terms.

Concepts that are similar to standard $\mathcal{ALC}$ concepts also have similar interpretations. For the definition of conrete terms we assume the implicit existence of a `Rel`-network. For a domain element δ to be in a concept $\exists(P_1, P_2, R)$ there must exist concrete domain elements mapped to δ via the feature paths P_1, P_2 and a constraint that adds the pair of those concrete elements (ordered by the order of the feature paths) to relation R. The concept $\forall(P_1, P_2, R)$ instead holds all elements, for which the existence of such feature-path-mapped concrete domain elements implies the existence of the requested constraint.

[3] Section 4 will give more details on *feature paths*

Definition 2.3.4 (*Semantics of $\mathcal{ALC}(D)$*)

Let $I = (\Delta_I, \cdot^I, Net_I)$ be an interpretation with the domain Δ_I and a `Rel`*-network Net_I. The function $\cdot^I$ maps:*

- *basic concepts to subsets of the domain: $C^I \subseteq \Delta_I$*
- *standard roles to binary relations: $r^I \subseteq \Delta_I \times \Delta_I$*
- *abstract features to partial functions: $a^I : \Delta_I \to \Delta_I$*
- *concrete features to partial functions: $f^I : \Delta_I \to V$*
- *feature paths of length two to the sequential execution of both features: $(a\ f)^I(\delta) = f^I(a^I(\delta))$*

This mapping is extended to complex concepts as follows:
For $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists r.C$, $\forall r.C$ see Definition 2.2.2.

8. $(\exists(P_1, P_2, R))^I = \left\{\delta \in \Delta_I \mid \exists x, y \in V_{Net_I} : x = P_1^I(\delta), y = P_2^I(\delta), (x, y, R) \in Net_I\right\}$
9. $(\forall(P_1, P_2, R))^I = \left\{\delta \in \Delta_I \mid \forall x, y \in V_{Net_I} : (x = P_1^I(\delta), y = P_2^I(\delta)) \Rightarrow (x, y, R) \in Net_I\right\}$

A concept $C \in \mathcal{C}_{\mathcal{ALC}(\mathbf{D})}$ is satisfied *by I if $C^I \neq \emptyset$ and Net_I is satisfiable in $\mathbf{D}$. The same holds for GCIs and TBoxes, to satisfy anything, an interpretation must provide a satisfiable* `Rel`*-network.*

Most of the common reasoning techniques on description logics require concepts and TBoxes to be in *negation normal form* i.e. to have the concept negation symbol only directly before concept names. To be able to present such reasoning procedures over the different extensions of $\mathcal{ALC}$ with concrete domains, ways had to be sought to handle concept negation on concrete terms on one hand and in case of general TBoxes, the existence of infinite models on the other hand. These were found in different 'admissibilities'.

2.3.3. Admissibility and ω-Admissibility

In [BH91] Baader and Hanschke presented an algorithm that decided concept satisfiability w.r.t. an *acyclic* TBox. This is a severe restriction on TBoxes with which reasoning can even be reduced to reasoning with respect to an empty TBox (see e.g. the DL Handbook [BCM+03] on that). Furthermore they do only use existential concrete terms and assume a single model in $\mathfrak{M}$ (cf. Appendix B). Even for this stripped-down version of $\mathcal{ALC}(\mathbf{D})$ the reasoning required so-called *admissible* concrete domains.[4]

[4]This presented definition is not the same as in [BH91] but transposed as presented in the appendix

Definition 2.3.5 (*Admissible concrete domain*)

A concrete domain $\boldsymbol{D} = (V, \texttt{Rel}, \mathfrak{M})$ *with* $\mathfrak{M} = \{M\}$ *is* admissible *if*

1. *The set of relations is closed under negation and contains a representation of the whole domain i.e.*
   a) *for every relation* $R \in \texttt{Rel}$ *with* $ar(R) = n$ *there is a relation* $\overline{R}$ *with* $ar(\overline{R}) = n$ *such that for* $t \in V^n$ *we have* $(t, R) \in M \Rightarrow (t, \overline{R}) \notin M$ *and* $(t, R) \notin M \Rightarrow (t, \overline{R}) \in M$
   b) *there exists a unary relation* $dom \in \texttt{Rel}$ *such that* $(v, dom) \in M$ *for all* $v \in V$
2. *satisfiability of finite* `Rel`*-networks is decidable*

Negation of concrete terms can be handled with the help of the first condition. Concepts $\exists(P_1, P_2, R)$ and $\exists(P_1, P_2, \overline{R})$ can, obviously, never have common elements. But they are,in fact, not complemental. The absence of adequately mapped concrete domain elements does also add to the negated concepts. This, however, presupposed features to be defined slightly different in [BH91] than in the present thesis, which we will thus not examine further. The aim of handling concept negation can nevertheless be reached in this way, while, as we will see, the possibility of infinite models must still be tackled.

Due to the special properties of constraint systems we cannot assume the existence of finite models for general TBoxes. Consider the TBox

$$\mathcal{T} = \{\top \sqsubseteq \exists(\boldsymbol{hasAge}, hasMother\, \boldsymbol{hasAge}, <)\}$$

setting our example concept to contain all domain elements. Here '$<$' is interpreted as an irreflexive, transitive and anti-symmetric binary relation, resembling the common 'less-than-and-not-equal' relation on numbers. Now assume an interpretation $I = (\Delta_I, \cdot_I, Net_I)$. According the semantics of $\mathcal{ALC}(\mathbf{D})$ we have

$$\begin{aligned}
&(\exists(\boldsymbol{hasAge}, hasMother\, \boldsymbol{hasAge}, <))^I \\
&= \big\{\delta \in \Delta_I \mid \exists x, y \in V_{Net_I} : \\
&\quad x = \boldsymbol{hasAge}^I(\delta), y = \boldsymbol{hasAge}^I(hasMother^I(\delta)), (x, y, <) \in Net_I\big\}
\end{aligned}$$

Now let $k_1, k_2, \ldots, k_n \in \Delta_I$, $a_1, a_2, \ldots, a_n \in V$, and assume

1. $(k_i, k_{i+1}) \in hasMother^I$ for $1 \leq i < n$
2. $(k_i, a_i) \in \boldsymbol{hasAge}^I$ for $1 \leq i \leq n$

For $\mathcal{T}$ to be satisfied this supposes $(a_i, a_{i+1}, <) \in Net_I$ for $1 \leq i < n$, ordering the a_i according their index i.e. $a_i < a_j \Leftrightarrow i < j$. Hence we have a_n as the greatest element in

this order. Further, k_n is the last in the line of mothers. This contradicts I satisfying $\mathcal{T}$. Facilitated by the open world assumption we could set some k_m with $m < n$ to be k_n's mother $(k_n, k_m) \in \mathit{hasMother}^I$. This, however leads to $(a_n, a_m, <) \in Net_I$, while, due to the transitivity of $<$ we have $(a_m, a_n, <) \in Net_I$ and thus $(a_n, a_n, <) \in Net_I$ which contradicts irreflexivity of $<$ and hence makes Net_I not satisfiable in $\mathbf{D}$. Thus the sequence of k's and that of a's mapped to them can never end or build a cycle.

Infinite models might lead to infinite reasoning procedures. To avoid this a kind of isomorphism-detection named *blocking* found its way into DLs which do not provide finite models. This works under the assumption that some domain elements and the concepts they belong to lead to similar patterns. This enhances the possibility to compute such sub-interpretation just once, mirroring the isomorphic parts implicitly. Such blocking mechanism, however, must handle infinite `Rel`-networks in concrete domains.

The notion of *ω-admissibility* as it was introduced in [LM07] requires slight adaptations of the concrete domain in focus. Only a finite set `Rel` of relation names is allowed. Further, it uses so-called *complete `Rel`-networks* to ensure the negation of concrete terms.

Definition 2.3.6 (*Complete `Rel`-Network, [LM07]*)

A `Rel`-network N in a constraint system $\boldsymbol{D} = (V, \mathtt{Rel}, \mathfrak{M})$ is complete *if for each pair of variables $(v, w) \in V_N \times V_N$ there is exactly one constraint $(v, w, R) \in N$.*

Each model in $\mathbf{D}$ must then be a complete network. Additionally, their special syntax of $\mathcal{ALC}(\mathbf{D})$ allows a conjunction of relation names in concrete terms. These facts in combination allow for the following negations of concrete terms. To illustrate the idea consider a simple technical example: let the set of relation names be of size three $\mathtt{Rel} = \{R, S, T\}$ and assume the concept $\exists(f_1, f_2, R \vee S)$. This concept shall, by definition contain, all elements δ for which

1. there exist concrete variables $x_1, x_2 \in V$ such that $f_i^I(\delta) = x_i$

2. and the `Rel`-network Net_I contains a constraint (x_1, x_2, R) or a constraint (x_1, x_2, S)

The definition of satisfiability of `Rel`-networks in presence of complete models forbids Net_I to contain both: since those models do contain exactly one constraint (a, b, R) per pair $(a, b) \in V_M \times V_M$, no `Rel`-network, renamed or not, that contains two different constraint for a pair can be a subset of such model. The negation of the above concept should consequently contain all elements ε for which

1. one of the above introduced x_1, x_2 does not exist

2. or Net_I cannot contain either of the introduced constraints

Thus the adequate negation of $\exists(f_1, f_2, R \vee S)$ may be

$$\begin{gathered}(\forall(f_1, f_1, R) \sqcap \forall(f_1, f_1, S)) \\ \sqcup \\ (\forall(f_2, f_2, R) \sqcap \forall(f_2, f_2, S)) \\ \sqcup \\ \forall(f_1, f_2, T)\end{gathered}$$

The first line leads to an inconsistency in Net_I if $f_1^I(\varepsilon)$ exists. The same is done by the second line for f_2. Together with these the last line expresses that, in case both f_1 and f_2 are defined for ε, then $(f_1^I(\varepsilon), f_2^I(\varepsilon), T)$ must be in Net_I. The negation of the $\forall$-concrete term may be taken from [LM07].

The handling of infinite networks was simplified with the assumption of $\mathbf{D}$ having the patchwork and the compactness property. Since these or at least their fuzzy equivalents are described in detail in the section on fuzzy concrete domains, we will only give a glimpse on the intuition here.

The patchwork property says: If two networks agree on their interface i.e. overlapping parts and each is satisfiable, then their union is satisfiable.

The compactness property makes infinite `Rel`-networks satisfiable if and only if all of their sub-networks with finite variable sets are satisfiable.

These are then combined with the satisfiability of finite networks to ω-admissibility.

Definition 2.3.7

A constraint system $\boldsymbol{D} = (V, \texttt{Rel}, \mathfrak{M})$ *is* ω-admissible *if*

- *it has the patchwork property*
- *it has the compactness property*
- *the satisfiability of finite* `Rel`*-networks is decidable*

The original authors could present a correct decision procedure for concept satisfiability w.r.t. general TBoxes in $\mathcal{ALC}(\mathbf{D})$, provided that $\mathbf{D}$ is ω-admissible and thus proved the decidability of that reasoning problem for the presented description logic.

The present paper will use a major part of the ideas from [LM07]. Eventually, adaptations had to be made according the varied definition of constraint systems, which are extended to multi-valued i.e. fuzzy constraint systems.

2.4. Fuzzy Sets

> "**Fuzzy logic** washing machines are gaining popularity. [...] As there is no standard for fuzzy logic, different machines perform in different manners." [5]

This subsection will give a short introduction on fuzzy sets, which are enabling to talk about vague or uncertain knowledge.[6] Due to great similarities in their semantics, it has been just a small step from fuzzy sets and fuzzy logic to fuzzy description logics which then were examined for their expressive power and complexity.

Traditional set theory is the one intuitively used by everybody talking about sets or even talking about things: 'Mr. XYZ is a human' clearly and definitely adds Mr. XYZ to the set of humans. In this approach an individual is in full amount or not at all member of a set. Difficulties arise when talking about uncertain assignments.

Example

Considering the set of all old people, is a person of age 50 in that set or not? One approach might be to fix that set and say: "Old describes all persons over 40." - "Isn't that offending?" - "Alright set it to 90." - "But aren't people in their seventies old, too?"

To avoid fixing a crisp border that decides between in and not in the set, a fuzzy set allows elements to be in this set by some degree.

Definition 2.4.1 (*Fuzzy Set*)

A fuzzy set Z *over a given domain* Δ *is characterized by a function* $\mu_Z : \Delta \to [0, 1]$ *from that domain to the real unit interval. The result* $\mu_Z(x)$ *for* $x \in \Delta$ *is called the* degree of membership *of* x *in* Z.

In the literature - as in the present paper - Z and μ_Z are used equally i.e. we see the fuzzy set Z as its own membership function.

We can now define 'old' with the help of the following membership function for a person p and $a(p)$ denoting a's age:

$$old(p) = \begin{cases} 0 & \text{if } a(p) < 30 \\ \frac{a(p)-30}{60} & \text{if } 30 \leq a(p) \leq 90 \\ 1 & \text{if } a(p) > 90 \end{cases}$$

This makes a 50 year old being 'old' by a degree of 1/3. It is still an opinion of the function's inventor, that people over 90 are definitely old while those under 30 are

[5] taken from '*[FAQs] Top Loader : What is Fuzzy Logic in a Washing Machine?*' on-line at 'http://skp.samsungcsportal.com/integrated/popup/FaqDetailPopup3.jsp?cdsite=in&seq=138486'; as of July 25, 2013

[6] For more information read the original paper [Zad65]

definitely not old. But, at least in between there is a slight increase of membership instead of a hard boundary.

Several useful membership functions have found use during the years. The above example is a so-called right-shoulder-function for the obvious reason. Figure 2.4.1 shows some of the most common functions.

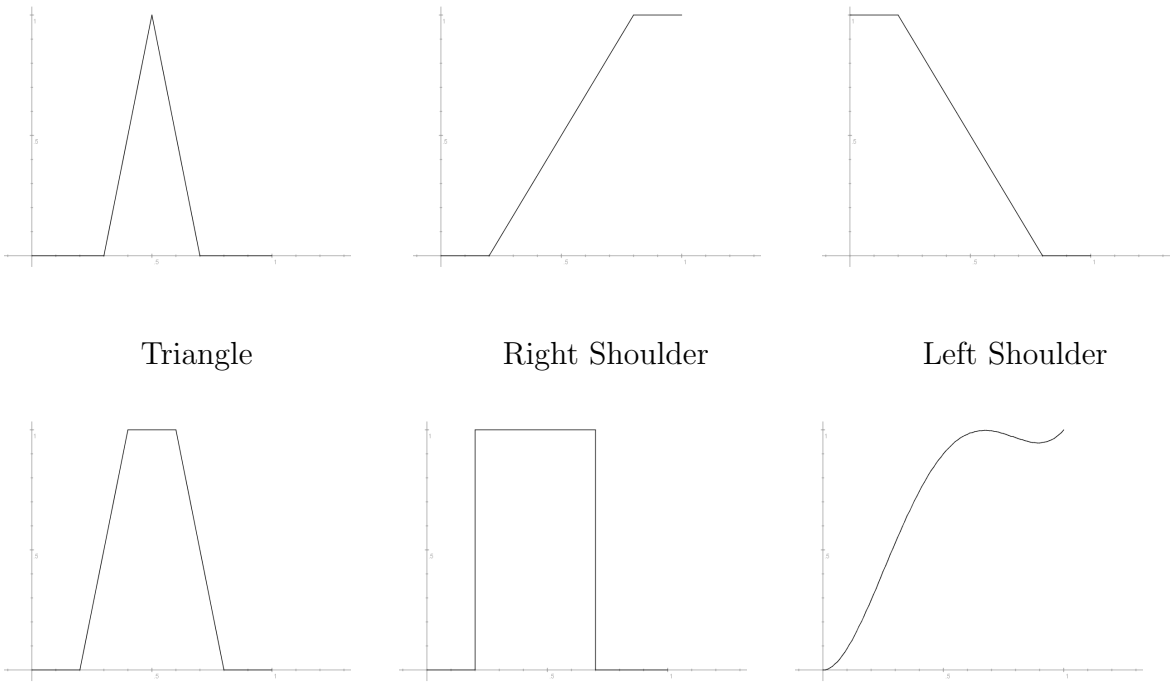


Figure 2.4.1: Fuzzy Membership Functions

To use the definition of one fuzzy set for the creation of others, fuzzy modifiers have found application. Attributes like *very, slightly, not* can then be expressed by functions like x^2, $\sqrt{x}$, $1-x$, respectively where x is the result of the original membership function. Reconsidering the example then makes a 60-year-old p being in the fuzzy set 'not very old' by a degree of $1 - old(p)^2 = 3/4$. The interpretation of such numbers is still in the eye of the beholder.

The union and intersection and the sub-set-of relation for fuzzy sets do exist but need to be re-defined to fit the notion of fuzzy sets. For subsets this is mostly done over the order of degrees i.e. for two fuzzy sets Y and Z we have $Y \subseteq Z$ if and only if $\mu_Y(a) \leq \mu_Z(a)$ for all $a \in \Delta$. For the intersection *triangular norm*s were chosen, which had already

been examined for different purposes earlier.[7] The fuzzy extension of description logics as initiated by works like [Yen91] relies on such t-norms and related notions which are used to interpret concept constructors under the assumption of fuzzy concepts. Several different conceptions exist in the literature of how these interpretations are settled (compare e.g. [BP12] or [Str05]). Since neither t-norms nor fuzzy concept constructors are considered in the remainder of the paper i.e. in the definition of fuzzy concrete domains, this coarse grain overview will suffice.

As a last step before going deeply into the fuzzyfication of concrete domains it must be mentioned that Straccia and Bobillo, the inventors of the *fuzzyDL* syntax and reasoner (see [BS08]) and based on that Mailis et al. have already worked on fuzzy concrete domains. So, they are not a new idea ([Str05],[SB08],[MSS$^+$12]). However, the former only used unary relations which can be reflected by standard (fuzzy) $\mathcal{ALC}$, while the latter employed $\mathcal{EL}^{++}$, a restricted description logic.

[7] Mostert and Shields, for example, published their well-known theorem about continuous t-norms in 1957 [MS57]

3. Fuzzy Constraint Systems

Allowing vague or uncertain knowledge in constraint systems must be done carefully to reasoning decidable when using them as concrete domains in description logics. Based on the work of former authors, the oft-cited [LM07], a version of constraint systems shall be introduced, that shows great similarities to that definition. Instead of forcing tuples to be in a relation or not to be in there, we would, however, like to allow them to relate with a degree between zero and one. In fact, we equip them with a set of degrees their membership in a distinct relation can possibly have. By considering this probably wide range of degrees, a relation as used in such system must not necessarily be a standard fuzzy membership function. The actual fixation of such functions and thus the assignment of membership degrees to tuples and relations must be set by a so-called *interpretation*.

3.1. Formal Definition

The presented fuzzy constraint system consists of a domain, a set of relation names and a set of models, as known from the crisp case. In contrast to this we add a set of membership degrees. Constraints in fuzzy constraint systems are then equipped with sub-sets of that, to denote the given tuple being in the given relation by one of the degrees in the connected set.

Definition 3.1.1 (*Fuzzy Constraint System*)

Let V and $\texttt{Rel}$ be countably infinite sets of variables and relation names, respectively and $\underline{\mathbf{1}} \subseteq [0,1]$ a subset of the real unit interval. Every relation $R \in \texttt{Rel}$ shall have an arity denoted by $ar(R)$.

A fuzzy $\texttt{Rel}$-constraint *then is a triple of the form (t, R, σ) with $t \in V^n, R \in \texttt{Rel}, ar(R) = n, \sigma \subseteq \underline{\mathbf{1}}$. A* fuzzy $\texttt{Rel}$-network *N is a (finite or infinite) set of fuzzy $\texttt{Rel}$-constraints. The set of variables used in N is abbreviated with V_N. Accordingly $\texttt{Rel}_N$ denotes the used relation names. Conversely, the* restriction *of N to a given set of variables $W \subseteq V$ is $N|_W = \{(t, R, \sigma) \in N \mid t \in W^n\}$.*

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

For the sake of readability we use lowercase Latin letters for variables in V, uppercase latin letters for relations in $\texttt{Rel}$, and lowercase Greek letters for sets of degrees. Further, tuples are notated without the surrounding parentheses if they are explicitly written in a constraint, i.e. the tuple $(a, b) \in V^2$, the relation $R \in \texttt{Rel}$, and a set of degrees $\sigma \subseteq \underline{\mathbf{1}}$ form the constraint (a, b, R, σ).

For the remainder of the paper we use the following abbreviations for special subsets of $\underline{\mathbf{1}}$:

Definition 3.1.2 (*Special Sets of Degrees, Simple Networks*)

Let $d \in \underline{\mathbf{1}}$, then:

$$\begin{aligned}
\sigma_{=d} &= \{d\} \\
\sigma_{\leq d} &= \{a \in \underline{\mathbf{1}} \mid a \leq d\} \\
\sigma_{<d} &= \{a \in \underline{\mathbf{1}} \mid a < d\} \\
\sigma_{\geq d} &= \{a \in \underline{\mathbf{1}} \mid a \geq d\} \\
\sigma_{>d} &= \{a \in \underline{\mathbf{1}} \mid a > d\} \\
\sigma_{\neq d} &= \underline{\mathbf{1}} \setminus \{d\}
\end{aligned}$$

A constraint of the form $(t, R, \sigma_{=d})$, in which the set of possible degrees is a singleton, shall be called a simple constraint*. A fuzzy* `Rel`*-network only containing simple constraints is a* simple network

A *fuzzy concrete domain* intuitively represents a set of `Rel`-networks, which are characterized by being satisfiable in $\mathcal{D}$. The notion of satisfiability of fuzzy `Rel`-networks is nearly identical to that of crisp constraint networks.

Definition 3.1.3 (*Interpretation, Model, Satisfiability*)

An interpretation of a fuzzy `Rel`*-network N is a function $I : V^n \times \texttt{Rel} \to \underline{\mathbf{1}}$ that maps a tuple of variables and a relation to a fuzzy degree from $\underline{\mathbf{1}}$. Here we insist on $ar(R) = n$. An interpretation I* satisfies *a constraint (t, R, σ) if $I(t, R) \in \sigma$. If such I satisfies all constraints in a fuzzy* `Rel`*-network N, then we say I* satisfies *N.*

A `Rel`*-network N is satisfiable in a fuzzy constraint system $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ if there exists a model $M \in \mathfrak{M}$, a mapping $\tau : V_N \to V_M$ from the variables in N to those in the model M (extended as shown on page 9), and an interpretation that satisfies both M and $\tau(N)$.*

Since fuzzy `Rel`*-constraints in which every degree of membership is allowed, i.e. those of the form $(t, R, \underline{\mathbf{1}})$, are satisfied by all possible interpretations, we call those* tautological *constraints.*

An interpretation I intuitively assigns every relation a fuzzy membership function $\mu_R : V^n \to \underline{\mathbf{1}}$ such that $\mu_R(t) = d$ if $I(t, R) = d$. To witness the satisfiability of a chosen network N we show that there is one among the models $M \in \mathfrak{M}$ such that we can rename the variables in N to fit those used in M and then find an interpretation of both. Intuitively this means, the interpreted network N is a subset of the equally interpreted model M.

For a simple network N there exists only one single interpretation that satisfies it, at least if only the results for $I(t, R)$ with $t \in V_N^n, R \in \texttt{Rel}_N$ are considered. In reality there, of course, still is an infinite number of interpretations that satisfy N, since for

tuples created from variables amongst which at least one is not in V_N and for relations $R \in \texttt{Rel} \setminus \texttt{Rel}_N$ there is no necessary value.

Definition 3.1.4 (*Canonical Interpretation*)

For a simple network $N \subseteq \{(t, R, \sigma_{=d}) \mid t \in V^n, R \in \texttt{Rel}, ar(R) = n, d \in \underline{\mathbf{1}}\}$ the interpretation I_N is a canonical interpretation if for $(t, R, \sigma_{=d}) \in N$: $I_N(t, R) = d$.

From [LM07] we know that in crisp constraint-networks a tuple can only be in a given relation if it is in no other relation (triggered by complete models, also see Definition 2.3.6). So, at least for complete networks we can definitely decide the membership of a tuple in a relation. A pair (t, R) having no respective constraint in such a network would mean t is not in relation R if we adopt that notion. Contrary, in the fuzzy context the absence of such constraint has the same effect as the presence of the tautological constraint $(t, R, \underline{\mathbf{1}})$. The fact '$t$ is not in the relation R' must thus be represented by a constraint that holds a set of degrees implying the non-relation. Mostly this would be symbolized by a degree of exactly zero, i.e. $(t, R, \sigma_{=0})$.

The special form of the constraints and the way in which networks are satisfied by distinct interpretations encourages the introduction of a normal form. This normal form assumes every pair of a tuple and a relation to appear exactly once in every Rel-network. We will show that such normal form exists for every network.

Definition 3.1.5 (*Fuzzy Network Normal Form*)

A fuzzy Rel-network N is said to be in fuzzy network normal form *if it contains exactly one constraint (t, R, σ) for every tuple $(t, R) \in V_N^n \times \texttt{Rel}_N$ with $ar(R) = n$*

The term 'fuzzy network' will be omitted when it is obvious that 'normal form' refers to fuzzy Rel-networks.

Lemma 3.1.6

Every fuzzy Rel-network N can be converted into a network N' such that N' is in normal form and satisfied by the same interpretations as N.

Proof: For every pair $(t, R) \in V_N^n \times \texttt{Rel}_N$ with $ar(R) = n$ let

$$\zeta(t, R) = \begin{cases} (t, R, \bigcap\{\sigma \mid (t, R, \sigma) \in N\}) & \text{if } \exists (t, R, \sigma) \in N \text{ for some } \sigma \subseteq \underline{\mathbf{1}} \\ (t, R, \underline{\mathbf{1}}) & \text{otherwise} \end{cases}$$

Then set

$$N' = \{\zeta(t, R) \mid (t, R) \in V_N^n \times \texttt{Rel}_N\}$$

Now assume an interpretation I. If $I(t, R) = d$ then I satisfies a constraint (t, R, σ) if and only if $d \in \sigma$. Thus for the subset $Z(t, R) = \{c \mid c = (t, R, \sigma_c) \in N\}$ of N consisting of only those constraints belonging to t and R we have

- For non-empty sets $Z(t, R)$

$$\begin{aligned} & I \text{ satisfies } Z(t,R) \\ \Leftrightarrow\ & d \in \sigma_c \text{ for each } c \in Z(t,R) \\ \Leftrightarrow\ & d \in \bigcap\{\sigma_c \mid c \in Z(t,R)\} \\ \Leftrightarrow\ & I \text{ satisfies } \zeta(t,R) \end{aligned}$$

- Empty constraint systems are – by definition – satisfied by all possible interpretations. If now there is no constraint (t, R, σ) for a given triple (t, R) then
  - $Z(t, R) = \emptyset$ which is satisfied by every interpretation
  - $\zeta(t, R) = (t, R, \underline{\mathbf{1}})$ which is also satisfied by every interpretation

This is true for every tuple $(t, R) \in V_N^n \times \mathtt{Rel}_N$. From that and $N' = \bigcup\{Z(t, R) \mid t \in V_N^n, R \in \mathtt{Rel}_N, ar(R) = n\}$ it directly follows that I satisfies N if and only if it satisfies N'. ■

From now on we can assume every fuzzy `Rel`-network to be in fuzzy network normal form.

Since some networks cannot be satisfied by any possible interpretation and thus are not satisfiable in any constraint system, it seems useful to define the notion of *`Rel`-clashes* which induces another, simpler definition of satisfiability.[8]

Definition 3.1.7 (*`Rel`-Clash*)

A clash in a fuzzy `Rel`*-network N is either a single constraint of the form $(a, b, R, \emptyset)$ or a set of constraints*

$$\{(t, R, \sigma_i) \mid i \in \mathbb{N}, i \leq n, n \in \mathbb{N}\}$$

with the same tuple of variables and relation, such that

$$\bigcap_{i \leq n} \sigma_i = \emptyset$$

It should be obvious, that a network contains a `Rel`-clash if and only if its normal form contains a constraint of the form $(t, R, \emptyset)$. Further, a `Rel`-network can now be identified as satisfiable in a fuzzy constraint system in a slightly different way.

[8] The name *`Rel`-clash* is chosen to distinguish those from obvious inconsistencies in $\mathcal{ALC}$-TBoxes which are also called *clashes*

Lemma 3.1.8 (*Satisfiability revised, cf. Definition 3.1.3*)

A fuzzy $\texttt{Rel}$*-network* N *is satisfiable in a fuzzy constraint system* $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ *if and only if there exists a model* $M \in \mathfrak{M}$ *and a mapping* $\tau : V_N \to V_M$ *such that* $\tau(N) \cup M$ *is* $\texttt{Rel}$*-clash-free.*

Proof: First notice, that a network that contains a $\texttt{Rel}$-clash itself cannot be satisfiable. This is due to the definition of satisfiability of constraints. For a constraint $(t, R, \emptyset)$ there cannot be an interpretation such that $I(t, R) \in \emptyset$.

'$\Rightarrow$' Let N be satisfiable in $\mathcal{D}$. Then there exists a model $M' \in \mathfrak{M}$, a mapping $\tau' : V_N \to V_{M'}$, and an interpretation I such that I satisfies all constraints in M' as well as all constraints in $\tau'(N)$. Assume $M' \cup \tau'(N)$ to contain a clash. Since both M' and N are $\texttt{Rel}$-clash-free, the clash can only be introduced by either τ' or the union of M' and $\tau'(N)$.

1. Let $\tau'(N)$ contain a clash. Then $\tau'(N)$ is not satisfiable.
2. Let $(t, R, \sigma) \in \tau'(N)$ and $(t, R, \psi) \in M'$ such that $\sigma \cap \psi = \emptyset$. Then for any interpretation I it holds: if $I(t, R) \in \sigma$ then $I(t, R) \notin \psi$ and vice versa, thus no interpretation can satisfy both networks.

So, if N is satisfiable then M' and τ' make $\tau'(N) \cup M'$ clash-free.

'$\Leftarrow$' Let $M \in \mathfrak{M}$ and $\tau : V_N \to V_M$ be the model and renaming as required. By definition $M \cup \tau(N)$ is clash-free if for all $(t, R, \sigma) \in \tau(N)$ and $(t, R, \psi) \in M$ we have $\sigma \cap \psi \neq \emptyset$. Thus for every $t \in V_M^n$ and $R \in \texttt{Rel}_M$ we can find a degree that an interpretation can map to i.e. there is a possibility to choose $I(t, R)$ such that $I(t, R) \in \sigma \cap \psi$ ■

For satisfiable networks we now define a completion. Such completion is a simple network whose canonical interpretation is an interpretation of a model and the (renamed) network to be completed.

Definition 3.1.9 (*Completion*)

Let N *be a fuzzy* $\texttt{Rel}$*-network which is satisfiable in* $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$*,* $M \in \mathfrak{M}$ *a model,* $\tau : V_N \to V_M$ *a renaming function, and* $I : V^n \times \texttt{Rel} \to \underline{\mathbf{1}}$ *an interpretation such that* I *satisfies* $\tau(N)$ *and* M*. Then we call*

$$N^{\tau,I} = \{(t, R, \sigma_{=d}) \mid t \in V_N^n, R \in \texttt{Rel}_N, d \in \underline{\mathbf{1}}, d = I(\tau(t), R)\}$$

the completion of N *under* τ *and* I*.*

Accordingly, we call a satisfiable network N *complete*, if there exist a mapping τ and an interpretation I satisfying a model $M \in \mathfrak{M}$ such that $N = N^{\tau,I}$. Note that the completion of such satisfiable network is necessarily also satisfiable, that the completion

of a finite network is still finite, and that the sets of variable and relation names do not change. Further $N^{\tau,I}$ is a simple network and for every pair $(t, R) \in V_N^n \times \mathtt{Rel}_N$ there exists exactly one constraint in $N^{\tau,I}$ i.e. $N^{\tau,I}$ is in normal form. For simpler use of fuzzy constraint systems we can assume any model in $\mathfrak{M}$ to be complete. This does not restrict the expressive power, but may increase the size of $\mathfrak{M}$ in comparison to arbitrary models. The clear advantage lies in the possibility of seeing models directly as their canonical interpretations (cf. 3.1.4). This assumption facilitates a re-definition of the completion of a network under a renaming and model instead of an interpretation i.e. $N^{\tau,M}$ is the completion of N under the canonical interpretation of M with a mapping $\tau : V_N \to V_M$.

Before defining an adequate transcription of ω-admissibility to fuzzy constraint systems, two special restrictions on fuzzy constraint systems shall be examined.

3.2. Special Cases

First we will show, that the requirement of allowing arbitrary arities for relations and tuples is not necessary, since all such fuzzy constraint systems can be transformed into such in which only arities of exactly two are considered. This is proved by giving such transformation and showing that it preserves satisfiability of fuzzy $\mathtt{Rel}$-networks. The other restriction to be inspected is given by the assumption of a finite set $\underline{\mathbf{1}}$ of possible membership degrees. We will see that those have no greater expressive power than crisp constraint systems.

3.2.1. Conversion of arbitrary arity Constraint Systems to binary Constraint Systems

In this subsection we show that starting at an arbitrary fuzzy constraint system it is possible to derive a fuzzy constraint system in which all relations are binary while keeping the satisfiability of given $\mathtt{Rel}$-networks. The desired goal is to transform arbitrary networks and models from the first version to the second (i.e. binary) one such that a transformed network is satisfiable in the new constraint system if and only if its source network was. This will lead to a remarkably simpler definition of the connection of $\mathcal{ALC}$-concepts with fuzzy concrete domains. The reduction is realized by de-composition of the tuples in the original constraint system. A combination of several fresh relation names will be used to form sub-networks that ensure that this de-composition is reversible. We will use the sets V of variables as already used in the source system and add a set Π of new variables used for the mentioned encoding.

Throughout the explanation and proof of the approach we will use $\mathcal{D} = (V, \mathtt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ to denote the arbitrary fuzzy constraint system while $\mathcal{D}_2 = (V, \mathtt{Rel}_2, \underline{\mathbf{1}}, \mathfrak{M}_2)$ denotes the binary case. Here $\mathtt{Rel}_2$ shall only contain binary relations and $\mathfrak{M}_2$ only such models holding binary relations.

Let $\Pi = \left\{\pi_{(i,n,k)} \mid i, n, k \in \mathbb{N}\right\}$ be a set of new variables, i.e. $V \cap \Pi = \emptyset$, and for

arbitrary $n \in \mathbb{N}$ let $\mu_n : V^n \to \mathbb{N}$ be a bijective mapping, that enumerates the n-tuples. The indices of those $\pi_{(i,n,k)}$ are then combined with variables from V such that i mirrors the position of v in a tuple, while n is the arity and k the number (via μ_n) of this tuple. The conversion is performed in two phases. The first is the conversion of all models, the second is the conversion of arbitrary networks which will use similar networks with slight differences.

For the models in $\mathfrak{M}_2$ we will construct constraints that, when united with an arbitrary network lead to `Rel`-clashes if the latter was not holding the correct tuples before.

With V_M being the set of all variables used in M set

$$Ar(M) = \{ar(R) \mid R \in \mathtt{Rel}, \exists (t, R, \sigma) \in M \text{ for some } t \in V_M^n, \sigma \subseteq \underline{\mathbf{1}}\}$$

as the set of all arities used in M and

$$\Pi_M = \left\{\pi_{(i,n,k)} \mid n \in Ar(M); \exists t \in V_M^n : \mu_n(t) = k; 1 \leq i \leq n\right\}$$

being the set of all elements of Π that are used to encode tuples which can be created from variables in M.

The fuzzy $\mathtt{Rel}_2$-networks introduced next shall ensure correct encoding of the original tuples.

The relation symbols R_Π and $R_{(*,*,\square)}$ and for all $i, n \in \mathbb{N}$: R_{μ_n}, $R_{(*,n,*)}$, $R_{(i,*,*)}$ shall be fresh i.e. $R_{\mu_n}, R_{(*,n,*)}, R_{(i,*,*)}, R_{(*,*,\square)}, R_\Pi \notin \mathtt{Rel}$. This means we set

$$\mathtt{Rel}_2 = \mathtt{Rel} \cup \{R_\Pi, R_{(*,*,\square)}\} \cup \{R_{\mu_n} \mid n \in \mathbb{N}\} \cup \{R_{(i,*,*)} \mid i \in \mathbb{N}\} \cup \{R_{(*,n,*)} \mid n \in \mathbb{N}\}$$

First of all we add the decomposed tuples to their originally used relations. For every constraint $c = (t, R, \sigma) \in M$ with $t = (v_1, v_2, \cdots, v_n)$ assume the network

$$\alpha'(c) = \left\{(v_i, \pi_{(i,n,\mu_n(t))}, R, \sigma) \mid 1 \leq i \leq n\right\}$$

Here we see R as a relation name in both $\mathcal{D}$ and $\mathcal{D}_2$. This means the arities of relations in `Rel` are ignored after conversion and we assume $ar(R) = 2$ for $R \in \mathtt{Rel}_2$.

These combinations of variables with their positioning must then be verified by multiple new constraints.

First we connect only those vs and πs that are really used together. For a given arity n let

$$M_{\mu_n} = \left\{(v, \pi_{(i,n,k)}, R_{\mu_n}, \sigma_{=\delta}) \mid v \in V_M, \pi_{(i,n,k)} \in \Pi_M, \delta = \delta_{\mu_n}(v, \pi_{(i,n,k)})\right\}$$

with

$$\delta_{\mu_n}(v, \pi_{(i,n,k)}) = \begin{cases} 1 & \text{if } \exists t \in V^n : t = (v_1, \cdots, v_{i-1}, v, v_{i+1}, \cdots, v_n) \wedge \mu_n(t) = k \\ 0 & \text{otherwise} \end{cases}$$

For every member of a special tuple t we thus keep its membership to this tuple and its position in there. A combination of v and $\pi_{(i,n,k)}$ is in relation R_{μ_n} with degree 1 if v was the i-th member of the very n-tuple that is numbered with k. All variables that were not part of that very tuple or on the wrong position get a degree of 0 when combined with that same $\pi_{(i,n,k)}$.

Next, to keep the members of each tuple at the right position in that very tuple, for each position number i let

$$M_{(i,*,*)} = \left\{(\pi_{(j,n,k)}, \pi_{(j,n,k)}, R_{(i,*,*)}, \sigma_{=\delta}) \mid \pi_{(j,n,k)} \in \Pi_M, \delta = \delta_{(i,*,*)}(\pi_{(j,n,k)})\right\}$$

with

$$\delta_{(i,*,*)}(\pi_{(j,n,k)}) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

To use $\pi_{(i,n,k)}$ only for those tuples with arity n: for a given arity n let

$$M_{(*,n,*)} = \left\{(\pi_{(i,o,k)}, \pi_{(i,o,k)}, R_{(*,n,*)}, \sigma_{=\delta}) \mid \pi_{(i,o,k)} \in \Pi_M, \delta = \delta_{(*,n,*)}(\pi_{(i,o,k)})\right\}$$

with

$$\delta_{(*,n,*)}(\pi_{(i,o,k)}) = \begin{cases} 1 & \text{if } o = n \\ 0 & \text{otherwise} \end{cases}$$

The next network shall ensure two members coming from the same tuple, to be recognizable with this characteristic by giving them degree 1 if they have the same k and 0 otherwise.

$$M_{(*,*,\square)} = \left\{(\pi_{(i,n,k)}, \pi_{(j,o,l)}, R_{(*,*,\square)}, \sigma_{=\delta}) \mid \pi_{(i,n,k)}, \pi_{(j,o,l)} \in \Pi_M, \delta = \delta_{(*,*,\square)}(\pi_{(i,n,k)}, \pi_{(j,o,l)})\right\}$$

with

$$\delta_{(*,*,\square)}(\pi_{(i,n,k)}, \pi_{(j,o,l)}) = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{otherwise} \end{cases}$$

At last, to ensure that normal variables are not mistaken for positioning variables and vice versa let

$$M_\Pi = \{(a, a, R_\Pi, \sigma_{=\delta}) \mid a \in V_M \cup \Pi_M, \delta = \delta_\Pi(a)\}$$

with

$$\delta_\Pi(a) = \begin{cases} 1 & \text{if } a \in \Pi \\ 0 & \text{otherwise} \end{cases}$$

Finally set

$$\begin{aligned}\alpha(M) =&(\bigcup_{i\leq max(Ar(M))} M_{(i,*,*)})\\ &\cup(\bigcup_{n\in Ar(M)} M_{(*,n,*)})\\ &\cup(\bigcup_{n\in Ar(M)} M_{\mu_n})\\ &\cup(\bigcup_{c\in M} \alpha'(c))\\ &\cup M_{(*,*,\square)} \cup M_{\Pi}\end{aligned}$$

It directly follows, that $V_{\alpha(M)} = V_M \cup \Pi_M$.

The α-conversion is very strict in the sense that it adds constraints with a $\sigma_{=0}$-degree set for all tuples that are not used. The application of that same conversion on fuzzy `Rel`-networks that are not complete would mean that no further constraints can be added. The conversion of arbitrary `Rel`-networks into $\texttt{Rel}_2$-networks thus works quite similar while not using all the $\sigma_{=0}$-constraints in the encoding part. This preserves the possibility to add new constraints to incomplete networks.

Let N be an arbitrary `Rel`-network. Then $\beta(N)$ is a $\texttt{Rel}_2$-network constructed as follows: The relation symbols $R_{\mu_n}, R_{(i,*,*)}, R_{(*,n,*)}, R_{(*,*,\square)}, R_{\Pi}$ and the bijections μ_n are the same as for the α-conversion. The networks added here are subsets of those introduced for α-conversion and are included for the same reasons. Thus the explanations are kept even shorter.

Let $c = (t, R, \sigma) \in N$ with $t = (v_1, v_2, \cdots, v_n)$

First

$$\beta'(c) = \big\{(v_i, \pi_{(i,n,\mu_n(t))}, R, \sigma) \mid 1 \leq i \leq n\big\}$$

To keep the positions

$$N_{(i,*,*)}(c) = \big\{(\pi_{(i,n,\mu_n(t))}, \pi_{(i,n,\mu_n(t))}, R_{(i,*,*)}, \sigma_{=1}) \mid 1 \leq i \leq n\big\}$$

To keep the right arities let

$$N_{(*,n,*)}(c) = \big\{(\pi_{(i,n,\mu_n(t))}, \pi_{(i,n,\mu_n(t))}, R_{(*,n,*)}, \sigma_{=1}) \mid 1 \leq i \leq n\big\}$$

To keep the elements members of the same tuple

$$N_{(*,*,\square)}(c) = \big\{(\pi_{(i,n,\mu_n(t))}, \pi_{(j,n,\mu_n(t))}, R_{(*,*,\square)}, \sigma_{=1}) \mid 1 \leq i, j \leq n\big\}$$

Next, let

$$N_{\mu_n}(c) = \big\{(v_i, \pi_{(i,n,\mu_n(t))}, R_{\mu_n}, \sigma_{=1}) \mid 1 \leq i \leq n\big\}$$

Let further

$$N_\Pi(c) = \left\{(\pi_{(i,n,\mu_n(t))}, \pi_{(i,n,\mu_n(t))}, R_\Pi, \sigma_{=1}) \mid 1 \leq i \leq n\right\} \cup \left\{(v_i, v_i, R_\Pi, \sigma_{=0}) \mid 1 \leq i \leq n\right\}$$

Finally set

$$\beta(c) = N_{(i,*,*)}(c) \cup N_{(*,n,*)}(c) \cup N_{(*,*,\square)}(c) \cup N_{\mu_n}(c) \cup N_\Pi(c) \cup \beta'(c)$$

and

$$\beta(N) = \bigcup_{c \in N} \beta(c)$$

It should be clear, that $\beta'(c) = \alpha'(c)$ and $\beta(N) \subseteq \alpha(N)$ for all constraints and networks, respectively.

It shall now be shown that β is injective and satisfiability-preserving i.e. no two different fuzzy `Rel`-networks are converted into the same and a fuzzy `Rel`-network is satisfiable in $\mathcal{D}$ if and only if $\beta(N)$ is satisfiable in $\mathcal{D}_2$.

Lemma 3.2.1

The α-conversion and the β-conversion as defined above are injective.

Proof: For two `Rel`-networks N, N' let $N \neq N'$ and w.l.o.g. $N \setminus N' \neq \emptyset$. Then there exists a constraint $c = (t, R, \sigma) \in N$ such that $c \notin N'$. Then $\beta(N) \supseteq \beta'(c)$. Let $(v_i, \pi_{(i,n,k)}, R, \sigma) \in \beta'(c)$. If now $(v_i, \pi_{(i,n,k)}, R, \sigma) \in \beta(N')$ there must be a constraint $(t', R, \sigma) \in N'$ such that $\mu_n(t') = k = \mu_n(t)$. Due to the bijectivity of μ_n we get $t' = t$ and thus $c \in N'$ which is wrong by assumption, thus $N \neq N' \Rightarrow \beta(N) \neq \beta(N')$. This proof does also hold for α-conversion. ■

It can be shown, that clash-free networks are always converted into clash-free networks.

Claim 3.2.2

A `Rel`-network N is clash-free if and only if $\beta(N)$ is also clash-free.

Proof: Let N be clash-free, then for every two constraints $c_1 = (t, R, \sigma)$ and $c_2 = (t, R, \sigma')$ it holds $\sigma \cap \sigma' \neq \emptyset$.

Since the additional networks $N_{(i,*,*)}, N_{(*,n,*)}, N_{(*,*,\square)}, N_{\mu_n}, N_\Pi$ do contain at most one instance of every pair of variables in combination with a distinct relation symbol, the only possible clashes are in $P := \bigcup_{c \in N} \beta'(c)$. Let $(v, \pi, R, \sigma) \in \beta'(c_1)$ and $(v, \pi, R, \sigma') \in \beta'(c_2)$. This pair would only be a clash if $\sigma \cap \sigma' = \emptyset$ which is already proved wrong. ■

Lemma 3.2.3

For a `Rel`*-network* N *it holds that* $\beta(N)$ *is satisfiable in* $\mathcal{D}_2$ *if and only if* N *is satisfiable in* $\mathcal{D}$*.*

Proof: We will base this proof on the following result: For an arbitrary `Rel`-network N and a $\mathcal{D}$-model M we have: $\tau(\beta(N)) \cup \alpha(M)$ is clash-free if and only if $\tau(N) \cup M$ is clash-free.

For this recall: a $\texttt{Rel}_2$-network $N' = \beta(N)$ is satisfiable in $\mathcal{D}$ if and only if there exists a model $M' \in \mathfrak{M}_2$ and a mapping $\tau : V_{N'} \to V_{M'}$ such that $\tau(N') \cup M'$ is clash-free. Here M' must be a converted $\mathcal{D}$-model i.e. $M' = \alpha(M)$ with $M \in \mathfrak{M}$.

At first we will show, that the encoding works correctly. Let $\tau(\beta(N)) \cup \alpha(M)$ be clash-free.

1. Assume for some $\pi \in \Pi$
   that $\tau(\pi) = v \in V$
   then $(\pi, \pi, R_\Pi, \sigma_{=1}) \in \beta(N) \Rightarrow (v, v, R_\Pi, \sigma_{=1}) \in \tau(\beta(N))$
   while $(v, v, R_\Pi, \sigma_{=0}) \in \alpha(M)$ which is a clash.
2. Assume for some $v \in V$
   that $\tau(v) = \pi \in \Pi$
   then $(v, v, R_\Pi, \sigma_{=0}) \in \beta(N) \Rightarrow (\pi, \pi, R_\Pi, \sigma_{=0}) \in \tau(\beta(N))$
   while $(\pi, \pi, R_\Pi, \sigma_{=1}) \in \alpha(M)$ which is a clash.
3. Assume for some $\pi_{(i,n,k)} \in \Pi$
   that $\tau(\pi_{(i,n,k)}) = \pi_{(j,o,l)}$
   with $i \neq j$
   then $(\pi_{(i,n,k)}, \pi_{(i,n,k)}, R_{(i,*,*)}, \sigma_{=1}) \in \beta(N) \Rightarrow (\pi_{(j,o,l)}, \pi_{(j,o,l)}, R_{(i,*,*)}, \sigma_{=1}) \in \tau(\beta(N))$
   while $(\pi_{(j,o,l)}, \pi_{(j,o,l)}, R_{(i,*,*)}, \sigma_{=0}) \in \alpha(M)$ which is a clash.
4. Assume for some $\pi_{(i,n,k)} \in \Pi$
   that $\tau(\pi_{(i,n,k)}) = \pi_{(j,o,l)}$
   with $n \neq o$
   then $(\pi_{(i,n,k)}, \pi_{(i,n,k)}, R_{(*,n,*)}, \sigma_{=1}) \in \beta(N) \Rightarrow (\pi_{(j,o,l)}, \pi_{(j,o,l)}, R_{(*,n,*)}, \sigma_{=1}) \in \tau(\beta(N))$
   while $(\pi_{(j,o,l)}, \pi_{(j,o,l)}, R_{(*,n,*)}, \sigma_{=0}) \in \alpha(M)$ which is a clash.
5. Assume for some $\pi_{(i,n,k)}, \pi_{(j,o,k)} \in \Pi$
   that $\tau(\pi_{(i,n,k)}) = \pi_{(i',n',k')}$
   and $\tau(\pi_{(j,o,k)}) = \pi_{(j',o',l)}$
   with $k' \neq l$
   then $(\pi_{(i,n,k)}, \pi_{(j,o,k)}, R_{(*,*,\square)}, \sigma_{=1}) \in \beta(N) \Rightarrow (\pi_{(i',n',k')}, \pi_{(j',o',l)}, R_{(*,*,\square)}, \sigma_{=1}) \in \tau(\beta(N))$
   while $(\pi_{(i',n',k')}, \pi_{(j',o',l)}, R_{(*,*,\square)}, \sigma_{=0}) \in \alpha(M)$ which is a clash.

Those arguments lead to:

1. $v \in V \Rightarrow \tau(v) \in V$
2. $\pi \in \Pi \Rightarrow \tau(\pi) \in \Pi$
3. $\tau(\pi_{(i,n,k)}) = \pi_{(i,n,k')}$
4. if for some $\pi_{(i,n,k)}$ it holds $\tau(\pi_{(i,n,k)}) = \pi_{(i,n,k')}$ for a distinct k', then for all $\pi_{(j,n,k)}$ we get $\tau(\pi_{(j,n,k)}) = \pi_{(j,n,k')}$ i.e. all πs with the same k must be renamed into π with the same k'.

For a given constraint $c = (t, R, \sigma) \in N$ let $\tau(\beta(c)) \cup \alpha(M)$ be clash-free. Let $\tau(\pi_{(i,n,k)} = \pi_{(i,n,l)}$. Then $(v_i, \pi_{(i,n,k)}, R, \sigma) \in \beta'(c)$ implies $(\tau(v_i), \pi_{(i,n,l)}, R, \sigma) \in \tau(\beta'(c))$ and thus $(\tau(v_i), \pi_{(i,n,l)}, R_{\mu_n}, \sigma_{=1}) \in N_{\mu_n}(c)$. This is not clashing with $\alpha(M)$ only if $(\tau(v_i), \pi_{(i,n,l)}, R_{\mu_n}, \sigma_{=1}) \in M_{\mu_n}$ which only is the case if $\tau(t)$ can be formed from variables in M and $\mu(\tau(t)) = l$. Thus necessarily $\tau(\beta(c)) = \beta(\tau(c))$. Since this holds for every constraint $c \in N$ it follows $\tau(\beta(N)) = \beta(\tau(N))$. With $\tau(\beta(N) \cup \alpha(M)) = \beta(\tau(N)) \cup \alpha(M) = \alpha(\tau(N) \cup M)$ and Claim 3.2.2 it becomes obvious, that $\tau(N) \cup M$ is clash-free if and only if $\tau(\beta(N)) \cup \alpha(M)$ is. Consequently N is satisfiable in $\mathcal{D}$ if and only if $\beta(N)$ is satisfiable in $\mathcal{D}_2$. ■

We have thus shown, that it is no restriction on the expressiveness of a fuzzy constraint system $\mathcal{D} = (V, \mathtt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ to assume having $ar(R) = 2$ for all $R \in \mathtt{Rel}$.

This result shall lead over to another syntactical restriction which, in this case, is a true semantical restriction. The assumption of finitely many values in $\underline{\mathbf{1}}$ makes every such fuzzy constraint system equivalent to a crisp constraint system.

3.2.2. Finitely Valued Fuzzy Constraint Systems

Considering a finite set $\underline{\mathbf{1}}$ of membership degrees enhances the possibility of reducing finitely valued fuzzy constraint systems (FVFCS) to crisp constraint systems.

Lemma 3.2.4

Every finitely valued fuzzy constraint system can be represented by an isomorphic crisp constraint system.

Proof: Let $\mathtt{Rel}^*$ be a new set of relation names with $|\mathtt{Rel}^*| = |\mathtt{Rel} \times 2^{\underline{\mathbf{1}}}|$ and let $\mu^* : \mathtt{Rel} \times 2^{\underline{\mathbf{1}}} \to \mathtt{Rel}^*$ be a bijective naming function. Then for every fuzzy network N: $N^* := \{(a, b, \mu^*(R, \sigma)) \mid (a, b, R, \sigma) \in N\}$ is a crisp $\mathtt{Rel}^*$-network which can losslessly be mapped back to N. The application of that renaming to all networks in a FVFCS $\mathcal{D} = (V, \mathtt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ makes $\mathcal{D}^* = (V, \mathtt{Rel}^*, \mathfrak{M}^*)$ with $\mathfrak{M}^* = \{M^* \mid M \in \mathfrak{M}\}$ a crisp constraint system. ■

Additionally, it can be shown, that in *FVFCS*s every clash-free network can be transformed into a set of simple networks which then directly represent the interpretations that

satisfy the original network. This will be done by stepwise decomposition of constraints into finite sets of simple constraints.

Definition 3.2.5

Let $T = (\mathfrak{N}, E, lab)$ be a labeled tree where $\mathfrak{N}$ is the set of nodes which are fuzzy `Rel`*-networks, $E \subseteq \mathfrak{N} \times \mathfrak{N}$ is the set of edges, and $lab : \mathfrak{N} \to \{open, closed\}$ is a labeling function.*

The *equalization*-process extends this tree T stepwise – with the intuition of preserving satisfiability along the paths in the tree. It is initialized with $\mathfrak{N} = \{N\}$, $E = \emptyset$, $lab(N) = open$. The set $\underline{\mathbf{1}}$ being finite implies all possible sets of degrees σ as used in fuzzy `Rel`-constraints to also be finite. This facilitates splitting such constraints into a set of simple constraints such that the union of their degree-sets yields σ again. Those simple constraints are then added to different successor nodes. While there are such, this method is applied on leaves in the tree that are labeled 'open', i.e. they have not been visited or still contain constraints with $|\sigma| \neq 1$. The application of that rule adds new networks to the tree which always have less non-simple constraints than their predecessor in the tree.

Algorithm 3.2.1 Equalization

Require: N^* - fuzzy constraint network
  function EQUALIZE(N^*)
    $\mathfrak{N} \leftarrow \{N^*\}$
    $E \leftarrow \emptyset$
    $lab(N^*) \leftarrow open$
    $T \leftarrow (\mathfrak{N}, E, lab)$
    while there exists a leaf N of T with $lab(N) = open$ **do**
      if there is a constraint $c := (a, b, R, \sigma) \in N$ with $|\sigma| > 1$ **then**
        for $d \in \sigma$ **do**
          create new `Rel`-network N_d
          $N_d \leftarrow N \setminus \{c\} \cup \{(a, b, R, \sigma_{=d})\}$
          $lab(N_d) \leftarrow open$
          add N_d to $\mathfrak{N}$
          add (N, N_d) to E
        end for
      else
        $lab(N) \leftarrow closed$
      end if
    end while
    return Set of leaves $\{L \in \mathfrak{N} \mid \forall N \in \mathfrak{N} : \neg\exists (L, N) \in E\}$
  end function

Theorem 3.2.6

For a finitely valued fuzzy constraint system $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ and a $\texttt{Rel}$-network N^* let $\mathfrak{N}^* = \text{EQUALIZE}(N^*)$ be the result set of Algorithm 3.2.1. Then an interpretation I satisfies N^* if and only if it satisfies at least one $N \in \mathfrak{N}^*$.

Proof: Let $T = (\mathfrak{N}, E, lab)$ be the search tree at some point during the application of Algorithm 3.2.1. We will now show, that a network N at some node of T is satisfied by a given interpretation I if and only if at least one of its successors is satisfied by I.

If $(a, b, R, \sigma) \in N$ and $|\sigma| > 1$, then for every $d \in \sigma$ the algorithm creates a set of networks N_d such that $(a, b, R, \sigma_{=d}) \in N_d$ and $(N, N_d) \in E$. For an interpretation I, let w.l.o.g. $I(a, b, R) = s$.

If I satisfies N then $s \in \sigma$. Thus there exists $N_s = N \backslash \{(a, b, R, \sigma)\} \cup \{(a, b, R, \sigma_{=s})\}$. Since I satisfies N it must also satisfy $N \setminus \{(a, b, R, \sigma)\}$. Together with $I(a, b, R) = s$ this results in I satisfying N_s.

If, otherwise, I does not satisfy N then $s \notin \sigma$ and thus none of the successor-networks contains a constraint $(a, b, R, \sigma_{=s})$. Since all of those successors do only contain simple constraints for (a, b, R), none of them is satisfied by I.

We have shown, that any interpretation, that satisfies N^* does also satisfy at least one $N \in \mathfrak{N}^*$ and that any interpretation that does not satisfy N^* does not satisfy any element in $\mathfrak{N}^+$. This resembles the yielded result. ■

Towards the termination of Algorithm 3.2.1 it can be stated, that infinite networks lead to infinite runs of EQUALIZE. In the finite case the algorithm terminates.

Lemma 3.2.7

For finite networks N the search tree T is finite.

Proof: A node $N \in \mathfrak{N}$ has exactly $|\sigma|$ successors for $c = (a, b, R, \sigma)$ being the chosen constraint. Since $\sigma \subseteq \underline{\mathbf{1}}$ and $\underline{\mathbf{1}}$ is finite, σ and consequently the number of successors of a single node is finite.

The maximal path-length is bounded by the number of constraints in N^* since in every step, the chosen constraint is replaced by a simple constraint and thus not in the scope for the succeeding steps.

Since the number of children for arbitrary nodes and the maximal path length are finite, the whole search tree is finite. The maximal number of nodes in T can be computed as follows: Consider an order on the constraints of N^* such that $N^* = \{c_k \mid 0 \leq k < n\}$ with $c_k = (a_k, b_k, R_k, \sigma_k)$. This is always possible for finite networks. Additionally assume the order of constraints being the order in which they are chosen

by the algorithm. Then the number of nodes at each level k is $L_k = \prod_{i=0}^{k-1} |\sigma_i|$. Let $m = max\{|\sigma_k| \mid 0 \leq k < n\}$. The number of all nodes in T then is

$$N_k = \sum_{k=0}^{n} L_k = \sum_{k=0}^{|N^*|} \prod_{i=0}^{k-1} |\sigma_i| \leq \sum_{k=0}^{|N^*|} \prod_{i=0}^{k-1} m = \sum_{k=0}^{|N^*|} m^k \leq \sum_{k=0}^{|N^*|} |\underline{\mathbf{1}}|^k = \frac{1 - |\underline{\mathbf{1}}|^{|N^*|+1}}{1 - |\underline{\mathbf{1}}|}$$

For large sets $\underline{\mathbf{1}}$ this (heavily over-estimated) upper bound approaches $|\underline{\mathbf{1}}|^{|N^*|}$ meaning for fixed $\underline{\mathbf{1}}$ it is exponential in the size of N^*. ■

This small excursion in the field of finetly valued fuzzy constraint systems has shown that this restriction of syntax allows for an easy conversion of those instances of fuzzy constraint systems to crisp constraint systems on the one hand and a simple detection of all interpretations for a fuzzy `Rel`-network on the other hand. Those assumptions cannot be made in case of an infinite set $\underline{\mathbf{1}}$. It would, nevertheless, be favourable to still have a means of integrating them into $\dashv\mathcal{LC}$ yielding decidable reasoning procedures.

As the remainder of the thesis will show, ω-admissibility as inspired by Lutz and Miličić provides this decidability. But first, this inspiration must lead to notions that are appropriate for the altered circumstances that were introduced by the fuzzification of constraint systems.

3.3. ω-Admissibility

To adopt the notion of *ω-admissible* constraint systems as presented in [LM07] it is necessary to find adequate definitions for the *patchwork* and the *compactness* property. The former is characterized by the possibility to patch two satisfiable networks together that agree on their interface. If both contain constraints for the same pair (t, R) then there shall be an interpretation that satisfies both of these constraints. If this is the case, then their union is satisfiable. In the original paper the interface is defined bi-directional via the set of variables. This can be avoided here due to the structure of the networks.

Definition 3.3.1 (*Patchwork Property*)

A fuzzy constraint system $\mathcal{D} = (V, \mathtt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ has the patchwork property *if for two finite satisfiable fuzzy `Rel`-networks N_1, N_2 it holds:*
If $N_1 \cup N_2$ is clash-free, then $N_1 \cup N_2$ is satisfiable in $\mathcal{D}$.

This definition is an adequate transcription from crisp constraint systems since the *interface* of two networks as defined there represents all constraints that are in both networks. In other words if those parts of both networks that use the same variables and relation names is not introducing a `Rel`-clash we assume they are 'patch-able' and thus their union shall be satisfiable.

The *compactness property* as defined by Lutz and Miličić deals with infinite networks and their sub-networks constructed from given sets of variables. This is adopted here, too.

Definition 3.3.2 (*Compactness Property*)

A fuzzy constraint system $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ *has the* compactness property *if any infinite network* N *in normal form is satisfiable if and only if all* $N|_V$ *with finite* V *are satisfiable.*

The above properties are then combined with satisfiability of finite networks to a property that will provide a decision procedure for the satisfiability of concepts w.r.t. general TBoxes in $\mathcal{ALC}(\mathcal{D})$ with a concrete domain $\mathcal{D}$ that has this property.

Definition 3.3.3 (*ω-Admissibility*)

We call a fuzzy constraint system $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ ω-admissible *if:*

1. *the satisfiability of finite fuzzy* `Rel`*-networks in* $\mathcal{D}$ *is decidable*
2. $\mathcal{D}$ *has the patchwork property*
3. $\mathcal{D}$ *has the compactness property*

As stated repeatedly, fuzzy constraint systems that are ω-admissible yield an $\mathcal{ALC}$-extension in which concept satisfiability w.r.t. general TBoxes is decidable. The following section will talk about this extended description logic, presenting formal definitions and a normal form that finds use in the decision procedure.

4. $\mathcal{ALC}(\mathcal{D})$ with Fuzzy Constraint Systems as Concrete Domains

Finally all the preliminary knowledge is introduced. The syntax and semantics of $\mathcal{ALC}(\mathcal{D})$ with D being a fuzzy constraint system can now be presented. Its definition is straightforward from the standard definition, using the sets N_C, N_{sR} of concept names and (standard) roles together with the well-known concept constructors, namely concept negation ($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), existential ($\exists r.C$) and value restriction ($\forall r.C$). We additionally expect some roles and all role-like mappings to the concrete domain to be functional i.e. such *feature* maps each element to at most one other. We therefore require the sets N_{aF} and N_{cF} of abstract and concrete features. Combinations of those are then combined to so-called *feature paths*.

Definition 4.1 (*Feature Path*)

A feature path P *consists of a concrete feature preceded by a sequence of abstract features i.e.*

$$P = a_1\ a_2\ \ldots\ a_n\ f$$

With $a_j \in N_{aF}, j, n \in \mathbb{N}, 0 \leq j \leq n, f \in N_{cF}$. *The* length *of such feature path is* $n+1$ *and must at least be one i.e. the concrete feature is obligatory.*

Those feature paths are important ingredients of *concrete terms* which build our connection to the concrete domain. Such concrete term basically looks like a fuzzy constraint preceded by a quantifier ($\exists$ or $\forall$), in which the concrete variables are substituted by feature paths. They accordingly represent two 'numerical' values of (implicit) domain individuals, standing in a given relation by one of the degrees declared in the last entry of the concrete term. To be a model for a concept or TBox, an interpretation must – besides satisfying the classical $\mathcal{ALC}$-concepts – provide a fuzzy `Rel`-network Net_I, that is satisfiable in the given fuzzy concrete domain while entailing all the requirements derived from concrete terms in the very concept(s). Consider the following example:

The concept $C_\exists = \exists(\boldsymbol{hasShoeSize}, \boldsymbol{hasAge}, Greater, \sigma_{\geq 0.5})$ with concrete features $\boldsymbol{hasShoeSize}, \boldsymbol{hasAge} \in N_{cF}$ and the fuzzy relation $Greater \in$ `Rel` shall contain all domain elements for which, intuitively, the shoe size is greater than the age, with a degree of at least 0.5. Technically spoken for $\delta \in \Delta_I$ we have $\delta \in C_\exists^I$ if

- there is a shoe size representation i.e. $\exists a \in V_{Net_I} : \boldsymbol{hasShoeSize}(\delta) = a$
- there is an age representation i.e. $\exists b \in V_{Net_I} : \boldsymbol{hasAge}(\delta) = b$
- the shoe size is greater than the age with a degree of at least 0.5, given that the fuzzy relation $Greater$ means what its name says

This means that a network Net_I that is part of a model for $C_\exists$ must hold a constraint $(a, b, Greater, \psi)$ such that

1. $\vartheta = \psi \cap \sigma_{\leq 0.5}$ is not empty

2. there exists a model $M \in \mathfrak{M}$, a renaming function $\tau : V_{Net_I} \to V_M$, and an interpretation I that satisfies M while $I(\tau(a), \tau(b), Greater) \in \vartheta$

To simplify this construction we suppose Net_I to be satisfiable and that for the constraint $(a, b, Greater, \psi) \in Net_I$ as introduced above we have $\psi \subseteq \sigma_{\leq 0.5}$. This ensures a non-empty union ϑ: since Net_I is satisfiable it must moreover be `Rel`-clash-free, thus ψ and consequently ϑ are non-empty.

On the other hand the concept $C_\forall = \forall(\boldsymbol{hasShoesize}, \boldsymbol{hasAge}, Greater, \sigma_{\leq 0.5})$ shall contain all those elements for which all shoe sizes and all ages obey the above-described restrictions. This sounds funny here since a person naturally has only one shoe size and age. Actually, by demanding feature paths for the mapping we reduce the number of possible constraints of that shape to at most one. What the concept expresses is: For $\delta \in C_\forall^I$ we have

- if there is a shoe size representation i.e. $\exists a \in V_{Net_I} : \boldsymbol{hasShoeSize}(\delta) = a$
- and if there is an age representation i.e. $\exists b \in V_{Net_I} : \boldsymbol{hasAge}(\delta) = b$
- then for all constraints in Net_I which specify a and b being in relation $Greater$ by some degree, this degree must be greater or equal to 0.5

This example should enlighten the reason of choice of the following syntactic forms and their interpretations. For handling purposes we assume all feature paths in the remainder of the paper to be of length at most two.

Definition 4.2 ($\mathcal{ALC}(\mathcal{D})$, *syntax*)

Let N_C, N_R, N_{cF} be pairwise disjoint countably infinite sets of concept names, role names, and concrete features, respectively. The set N_R of role names is a disjoint union of the set N_{aF} of abstract features and the set N_{sR} of standard roles. Further $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ is a fuzzy constraint system in which all relations are binary i.e. for every $R \in \texttt{Rel} : ar(R) = 2$. Then $\mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ is the set of all $\mathcal{ALC}(\mathcal{D})$-concepts, defined recursively:
Every concept name is a concept i.e. $N_C \subseteq \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$. For $C, D \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}, r \in N_R, R \in \texttt{Rel}, \sigma \subseteq \underline{\mathbf{1}}$ and feature paths P_1, P_2 the following are concepts in $\mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$:
$\top$, $\bot$, $\neg C$, $C \sqcup D$, $C \sqcap D$, $\exists r.C$, $\forall r.C$ as known from $\mathcal{ALC}$ (Definition 2.2.1)

8. $\exists(P_1, P_2, R, \sigma)$

9. $\forall(P_1, P_2, R, \sigma)$

A general concept inclusion (GCI) *is an axiom of the form $C \sqsubseteq D$ for arbitrary concepts $C, D \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$. A finite set of GCIs is called a* general TBox.

We assume the network Net_I to be in normal form i.e. there is exactly one constraint $(a, b, R, \sigma) \in Net_I$ for every triple $(a, b, R) \in V_{Net_I} \times V_{Net_I} \times \mathtt{Rel}_{Net_I}$ with $\sigma \in \underline{\mathbf{1}}$ (cf. Definition 3.1.5 on page 20). The interpretation of concrete terms as given in [LM07] can then be adopted with slight changes. The $\exists$-concrete term was already illustrated in the introductory example. Recall that the $\forall$-term in the original paper subsumed the abstract domain elements for which the existence of concrete domain that are mapped to the former via given feature paths implies the existence of a constraint in the attached `Rel`-network. In the fuzzy case we would like to extend this by the requirement that the constraint is necessarily satisfied by all $\mathcal{D}$-interpretations which satisfy that network.

Definition 4.3 (*Semantics*)

An interpretation $I = (\Delta_I, \cdot^I, Net_I)$ consists of a nonempty set Δ_I, the domain of I, a fuzzy `Rel`-network in normal form Net_I, and a function $\cdot^I$ that assigns

- *every concept $C \in N_C$ a subset of Δ_I: $C^I \subseteq \Delta_I$*
- *every standard role $r \in N_{sR}$ a binary relation over Δ_I: $r^I \subseteq \Delta_I \times \Delta_I$*
- *every abstract feature $a \in N_{aF}$ a partial function $f^I : \Delta_I \to \Delta_I$*
- *every concrete feature $f \in N_{cF}$ a partial function $c^I : \Delta_I \to V_{Net_I}$*

A feature path $P = a\,f$ with $a \in N_{aF}, f \in N_{cF}$ will be interpreted as the application of the interpretation of a feature on that of its very predecessor in the path i.e. for a domain element $\delta \in \Delta_I$:

$$P^I(\delta) = f^I(a^I(\delta))$$

This assignment is extended to complex concepts as follows: again for the $\mathcal{ALC}$ concept constructors the interpretation is the same (cf. Definition 2.2.2)

8. $(\exists(P_1, P_2, R, \sigma))^I = \left\{\delta \in \Delta_I \mid \exists v, w \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : v = P_1^I(\delta) \wedge w = P_2^I(\delta) \wedge (v, w, R, \sigma') \in Net_I \wedge \sigma' \subseteq \sigma\right\}$
9. $(\forall(P_1, P_2, R, \sigma))^I = \left\{\delta \in \Delta_I \mid \forall v, w \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : (v = P_1^I(\delta) \wedge w = P_2^I(\delta) \wedge (v, w, R, \sigma') \in Net_I) \Rightarrow \sigma' \subseteq \sigma\right\}$

An interpretation I is said to satisfy *a GCI $C \sqsubseteq D$ if $C^I \subseteq D^I$ and a general TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$ while, for both, Net_I is satisfiable in $\mathcal{D}$. We then call I a model for $\mathcal{T}$.*

We call a concept $\mathcal{C}$ satisfiable w.r.t. a general TBox $\mathcal{T}$ if there is an interpretation I that satisfies $\mathcal{T}$ while $\mathcal{C}^I \neq \emptyset$. To decide about concept-satisfiability in the presence of a fuzzy constraint system $\mathcal{D}$ we try to find a canonical model for a given TBox and concept. Obviously this satisfiability directly depends on the satisfiability of the chosen

`Rel`-network Net_I. Thus, a main part of the further approach must be the construction of that network and the decision about its satisfiability.

The presented Tableau algorithm is founded on the idea given in [LM07]. For use in this algorithm we introduce the notions of the sub-concepts of $\mathcal{ALC}(\mathcal{D})$-concepts and that of the *negation normal form* of such.

The structure of complex concept descriptions allows for a finer analysis of concepts regarding the concepts they are constructed of.

Definition 4.4 (*Sub-Concepts*)

Let $\text{sub} : \mathcal{C}_{\mathcal{ALC}(\mathcal{D})} \to 2^{\mathcal{C}_{\mathcal{ALC}(\mathcal{D})}}$ *be a function that assigns a concept* C *its set of sub-concepts as follows:*

- *if* $E = C \in N_C$ *then* $\text{sub}(E) = \{C\}$
- *if* $E = \neg D$ *then* $\text{sub}(E) = \{E\} \cup \text{sub}(D)$
- *if* $E = C \sqcap D$ *or* $E = C \sqcup D$ *then* $\text{sub}(E) = \{E\} \cup \text{sub}(C) \cup \text{sub}(D)$
- *if* $E = \exists r.D$ *or* $E = \forall r.D$ *then* $\text{sub}(E) = \{E\} \cup \text{sub}(D)$
- *if* $E = \exists(P_1, P_2, R, \sigma)$ *or* $E = \forall(P_1, P_2, R, \sigma)$ *then* $\text{sub}(C) = \{E\}$

This definition shall be extended on GCIs and TBoxes and sets of concepts as follows

- $\text{sub}(C \sqsubseteq D) = \text{sub}(C) \cup \text{sub}(D)$
- *for a general TBox* $\mathcal{T}$: $\text{sub}(\mathcal{T}) = \bigcup \{\text{sub}(C \sqsubseteq D) \mid C \sqsubseteq D \in \mathcal{T}\}$
- *for a set of concepts* $\mathcal{S} \subseteq \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$: $\text{sub}(\mathcal{S}) = \bigcup \{\text{sub}(C) \mid C \in \mathcal{S}\}$

In the case of concept sets explicitly denoted by enumerating their elements we omit the braces i.e. $\text{sub}(C, D) = \text{sub}(C) \cup \text{sub}(D)$

For an example on this definition consider $E = \neg(A \sqcup (B \sqcap \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C)))$. Then

$$\text{sub}(E) = \left\{ \begin{array}{l} \neg(A \sqcup (B \sqcap \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C))), \\ A \sqcup (B \sqcap \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C)), \\ A, \\ B \sqcap \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C), \\ B, \\ \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C), \\ \forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C, \\ \forall(P_1, P_2, R, \leq_{0.5}), \\ \neg C, \\ C \end{array} \right\}$$

We will eventually use $|C| := |\mathrm{sub}(C)|$ to denote the *size* of a concept C.

The definition of the *negation normal form* of $\mathcal{ALC}(\mathcal{D})$-concepts will be succeeded by the proof of existence of such normal form for all such concepts.

Definition 4.5 (*Negation Normal Form*)

A concept is in negation normal form (NNF) *if the negation symbol $\neg$ does only appear in front of concept names. A GCI is said to be in NNF if both, the left and the right side concept are in NNF. A TBox is in NNF if all contained GCIs are.*

Lemma 4.6 (*NNF*)

Any $\mathcal{ALC}(\mathcal{D})$-concept can be transformed into a semantically equivalent one in NNF.

Proof: Let $\mathrm{nnf} : \mathcal{C}_{\mathcal{ALC}(\mathcal{D})} \to \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ be a function that assigns every concept another concept, recursively defined as follows:

$$
\begin{aligned}
\mathrm{nnf}(C) &= C \text{ if } C \in N_C \\
\mathrm{nnf}(C \sqcap D) &= \mathrm{nnf}(C) \sqcap \mathrm{nnf}(D) \\
\mathrm{nnf}(C \sqcup D) &= \mathrm{nnf}(C) \sqcup \mathrm{nnf}(D) \\
\mathrm{nnf}(\exists r.C) &= \exists r.\mathrm{nnf}(C) \\
\mathrm{nnf}(\forall r.C) &= \forall r.\mathrm{nnf}(C) \\
\mathrm{nnf}(\exists(P_1, P_2, R, \sigma)) &= \exists(P_1, P_2, R, \sigma) \\
\mathrm{nnf}(\forall(P_1, P_2, R, \sigma)) &= \forall(P_1, P_2, R, \sigma) \\
\mathrm{nnf}(\neg C) &= \neg C \text{ if } C \in N_C \\
\mathrm{nnf}(\neg\neg C) &= \mathrm{nnf}(C) \\
\mathrm{nnf}(\neg(C \sqcap D)) &= \mathrm{nnf}(\neg C) \sqcup \mathrm{nnf}(\neg D) \\
\mathrm{nnf}(\neg(C \sqcup D)) &= \mathrm{nnf}(\neg C) \sqcap \mathrm{nnf}(\neg D) \\
\mathrm{nnf}(\neg(\exists r.C)) &= \forall r.(\mathrm{nnf}(\neg C)) \\
\mathrm{nnf}(\neg(\forall r.C)) &= \exists r.(\mathrm{nnf}(\neg C)) \\
\mathrm{nnf}(\neg(\exists(P_1, P_2, R, \sigma))) &= \forall(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma) \\
\mathrm{nnf}(\neg(\forall(P_1, P_2, R, \sigma))) &= \exists(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma)
\end{aligned}
$$

The last two lines create networks in a way, that necessarily lead to a clash in Net_I if the network would contain a constraint satisfying the negated axiom.

Application of nnf transforms an arbitrary concept into a semantically equal one in NNF:

1. *It holds that* $\mathrm{nnf}(E)^I = E^I$
   for any concept $E \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ and any interpretation I. Consider an arbitrary

interpretation I. For $C \in N_C$ this is obviously true. Further if $E = \neg C$ then

$$\mathrm{nnf}(E) = \mathrm{nnf}(\neg C)^I = (\neg C)^I = E^I$$

For concrete terms the correctness can also be shown directly:

- $E = \exists(P_1, P_2, R, \sigma)$
  from the above definition we directly get $\mathrm{nnf}(E) = E$, which obviously obeys the assumption.
- for $E = \forall(P_1, P_2, R, \sigma)$ the case is similar
- $E = \neg(\exists(P_1, P_2, R, \sigma))$

$$\begin{aligned}
\mathrm{nnf}(E) &= \mathrm{nnf}(\neg(\exists(P_1, P_2, R, \sigma))) \\
&= \forall(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma) \\
\mathrm{nnf}(E)^I &= (\forall(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma))^I \\
&= \big\{\delta \in \Delta_I \mid \forall a, b \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : \\
&\quad (a = P_1^I(\delta) \wedge b = P_2^I(\delta) \wedge (a, b, R, \sigma') \in Net_I) \Rightarrow \sigma' \subseteq \underline{\mathbf{1}} \setminus \sigma\big\} \\
&= \big\{\delta \in \Delta_I \mid \forall a, b \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : \\
&\quad \neg(a = P_1^I(\delta) \wedge b = P_2^I(\delta) \wedge (a, b, R, \sigma') \in Net_I) \vee \sigma' \subseteq \underline{\mathbf{1}} \setminus \sigma\big\} \\
&= \Delta_I \setminus \big\{\delta \in \Delta_I \mid \neg(\forall a, b \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : \\
&\quad \neg(a = P_1^I(\delta) \wedge b = P_2^I(\delta) \wedge (a, b, R, \sigma') \in Net_I) \vee \sigma' \subseteq \underline{\mathbf{1}} \setminus \sigma)\big\} \\
&= \Delta_I \setminus \big\{\delta \in \Delta_I \mid \exists a, b \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : \\
&\quad (a = P_1^I(\delta) \wedge b = P_2^I(\delta) \wedge (a, b, R, \sigma') \in Net_I) \wedge \neg(\sigma' \subseteq \underline{\mathbf{1}} \setminus \sigma)\big\} \\
&= \Delta_I \setminus \big\{\delta \in \Delta_I \mid \exists a, b \in V_{Net_I}, \sigma' \subseteq \underline{\mathbf{1}} : \\
&\quad (a = P_1^I(\delta) \wedge b = P_2^I(\delta) \wedge (a, b, R, \sigma') \in Net_I) \wedge \sigma' \subseteq \sigma\big\} \\
&= \Delta_I \setminus (\exists(P_1, P_2, R, \sigma))^I \\
&= (\neg\exists(P_1, P_2, R, \sigma))^I \\
&= E^I
\end{aligned}$$

- for $E = \neg\forall(P_1, P_2, R, \sigma)$ the proof is similar

Thus we can assume the existence of concepts $C, D \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ such that

$$\begin{aligned}
(\mathrm{nnf}(C))^I = C^I, &(\mathrm{nnf}(\neg C))^I = (\neg C)^I, \\
(\mathrm{nnf}(D))^I = D^I, &(\mathrm{nnf}(\neg D))^I = (\neg D)^I
\end{aligned} \tag{4.6.1}$$

For complex concepts we use this induction hypothesis to prove local correctness according to their structure. Let $E \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ be a complex concept:

- $E = C \sqcap D$

$$\begin{aligned} \mathrm{nnf}(E) &= \mathrm{nnf}(C \sqcap D) \\ &= \mathrm{nnf}(C) \sqcap \mathrm{nnf}(D) \\ \mathrm{nnf}(E)^I &= (\mathrm{nnf}(C) \sqcap \mathrm{nnf}(D))^I \\ &= \mathrm{nnf}(C)^I \cap \mathrm{nnf}(D)^I \\ &= C^I \cap D^I \\ &= (C \sqcap D)^I \\ &= E^I \end{aligned} \tag{4.6.1}$$

- for $E = C \sqcup D$ the proof is similar

- $E = \exists r.C$

$$\begin{aligned} \mathrm{nnf}(E) &= \mathrm{nnf}(\exists r.C) \\ &= \exists r.\mathrm{nnf}(C) \\ \mathrm{nnf}(E)^I &= (\exists r.\mathrm{nnf}(C))^I \\ &= (\exists r.C)^I \\ &= E^I \end{aligned} \tag{4.6.1}$$

- for $E = \forall r.C$ the proof is similar

- $E = \neg(\neg C)$

$$\begin{aligned} \mathrm{nnf}(E) &= \mathrm{nnf}(\neg(\neg C)) \\ &= \mathrm{nnf}(C) \\ E^I &= (\mathrm{nnf}(C))^I \\ &= C^I \\ &= \Delta_I \setminus (\Delta_I \setminus C^I) \\ &= (\neg(\neg C))^I \\ &= E^I \end{aligned} \tag{4.6.1}$$

- $E = \neg(C \sqcap D)$

$$\begin{aligned}
\mathrm{nnf}(E) &= \mathrm{nnf}(\neg(C \sqcap D)) \\
&= \mathrm{nnf}(\neg C) \sqcup \mathrm{nnf}(\neg D) \\
\mathrm{nnf}(E)^I &= (\mathrm{nnf}(\neg(C)) \sqcup \mathrm{nnf}(\neg(D)))^I \\
&= \mathrm{nnf}(\neg C)^I \cup \mathrm{nnf}(\neg D)^I \\
&= (\neg C)^I \cup (\neg D)^I \qquad (4.6.1) \\
&= \Delta_I \setminus C^I \cup \Delta_I \setminus D^I \\
&= \Delta_I \setminus (C^I \cap D^I) \\
&= \Delta_I \setminus (C \sqcap D)^I \\
&= (\neg(C \sqcap D))^I \\
&= E^I
\end{aligned}$$

- for $E = \neg(C \sqcup D)$ the proof is similar
- $E = \neg(\exists r.C)$

$$\begin{aligned}
\mathrm{nnf}(E) &= \mathrm{nnf}(\neg(\exists r.C)) \\
&= \forall r.(\mathrm{nnf}(\neg C)) \\
\mathrm{nnf}(E)^I &= (\forall r.(\mathrm{nnf}(\neg C)))^I \\
&= \left\{x \in \Delta_I \mid \forall y \in \Delta_I : (x,y) \in r^I \Rightarrow y \in \mathrm{nnf}(\neg C)^I\right\} \\
&= \left\{x \in \Delta_I \mid \forall y \in \Delta_I : (x,y) \in r^I \Rightarrow y \in (\neg C)^I\right\} \qquad (4.6.1) \\
&= \left\{x \in \Delta_I \mid \forall y \in \Delta_I : (x,y) \in r^I \Rightarrow y \in \Delta_I \setminus C^I\right\} \\
&= \Delta_I \setminus \left\{x \in \Delta_I \mid \neg(\forall y \in \Delta_I : (x,y) \in r^I \Rightarrow y \in \Delta_I \setminus C^I)\right\} \\
&= \Delta_I \setminus \left\{x \in \Delta_I \mid \exists y \in \Delta_I : \neg((x,y) \in r^I \Rightarrow y \in \Delta_I \setminus C^I)\right\} \\
&= \Delta_I \setminus \left\{x \in \Delta_I \mid \exists y \in \Delta_I : \neg((x,y) \notin r^I \vee y \notin C^I)\right\} \\
&= \Delta_I \setminus \left\{x \in \Delta_I \mid \exists y \in \Delta_I : (x,y) \in r^I \wedge y \in C^I\right\} \\
&= \Delta_I \setminus (\exists r.C)^I \\
&= (\neg(\exists r.C))^I \\
&= E^I
\end{aligned}$$

- for $E = \neg(\forall r.C)$ the proof is similar

It is thus shown $\mathrm{nnf}(E)^I = E^I$ for all concepts $E \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ and $\mathcal{ALC}(\mathcal{D})$-interpretations I

2. *Application of the above equations terminates:*
   During the application of nnf it holds that the concepts which the function is applied on in a subsequent step are strictly smaller (w.r.t. $|C|$, see above) than the currently regarded. This fact and its influence on the termination of the algorithm should be quite obvious.
3. *The construction does indeed provide the desired normal form:*
   Let $E \in \mathcal{C}_{\mathcal{ALC}(\mathcal{D})} \setminus N_C$ be a complex concept on which nnf has been applied. Assume there is a negated sub-concept $\neg D \in sub(E)$. For non-negated concepts the upper seven rules just push the normalization to sub-concepts. At some point during application there must have been a direct application $\text{nnf}(\neg D)$. Since this did not change that concept, none of the lower seven rules was applied. Thus it must necessarily have been the rule for negated basic concepts, hence $D \in N_C$. ■

Reconsidering the above used example concept

$$E = \neg(A \sqcup (B \sqcap \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C)))$$

we get

$$\begin{aligned}
&\text{nnf}(E) \\
&= \text{nnf}(\neg A) \sqcap \text{nnf}(\neg(B \sqcap \exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C))) \\
&= \neg A \sqcap (\text{nnf}(\neg B) \sqcup \text{nnf}(\neg\exists r.(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C))) \\
&= \neg A \sqcap (\neg B \sqcup (\forall r.(\text{nnf}(\neg(\forall(P_1, P_2, R, \leq_{0.5}) \sqcap \neg C))))) \\
&= \neg A \sqcap (\neg B \sqcup (\forall r.(\text{nnf}(\neg\forall(P_1, P_2, R, \leq_{0.5})) \sqcup \text{nnf}(\neg\neg C)))) \\
&= \neg A \sqcap (\neg B \sqcup (\forall r.(\exists(P_1, P_2, R, >_{0.5}) \sqcup C)))
\end{aligned}$$

Further the TBox shall consist of a single axiom of the form $\top \sqsubseteq C$. This is also possible to achieve with any ontology. The following rules are satisfiability preserving:

$$\begin{gathered}
C \sqsubseteq D_1, C \sqsubseteq D_2 \rightsquigarrow C \sqsubseteq D_1 \sqcap D_2 \\
\text{for } C \neq \top : C \sqsubseteq D \rightsquigarrow \top \sqsubseteq \neg C \sqcup D
\end{gathered}$$

Lemma 4.7

Exhaustive application of the above rules on a TBox T leads to a TBox T' that contains one single axiom only and has the same models that T has.

Proof: For the first rule consider

$$C^I \subseteq D_1^I \wedge C^I \subseteq D_2^I \Leftrightarrow C^I \subseteq D_1^I \cap D_2^I$$

For the second

$$\begin{aligned} & C^I \subseteq D^I \\ \Leftrightarrow & (\Delta_I \setminus C^I) \supseteq (\Delta_I \setminus D^I) \\ \Leftrightarrow & (\Delta_I \setminus C^I) \cup D^I \supseteq \Delta_I \\ \Leftrightarrow & \Delta_I \subseteq (\Delta_I \setminus C^I) \cup D^I \end{aligned}$$

As to the termination of rule application it can be seen, that the first rule strictly lessens the number of axioms, while the second leads to axioms of the form, that are used for the first. Let no rule be applicable. Assume T' to contain at least two GCIs $C_1 \sqsubseteq D_1$ and $C_2 \sqsubseteq D_2$. If $C_1 = C_2$ then the first rule is applicable. Else at least for one $i \in \{1, 2\}$: $C_i \neq \top$ and thus the second rule can be applied. ■

We can now assume all used concepts and TBoxes to be in NNF while all TBoxes consist of one single axiom only. With a proper definition of $\mathcal{ALC}(\mathcal{D})$ with fuzzy $\mathcal{D}$ at hand, it is time to finally tackle concept satisfiability in that DL.

5. Tableau Algorithm

The main goal of this section is to show, that ω-admissibility (cf. Definition 3.3.3 on page 33) of the fuzzy constraint system that forms the concrete domain makes concept satisfiability in $\mathcal{ALC}(\mathcal{D})$ decidable. We thus present an algorithm that provides such decision procedure. It is designed as a tableau algorithm that tries to construct a model for the given input i.e. the concept $\mathcal{C}$ and the TBox $\mathcal{T}$ whose satisfiability are to be shown. As a result of the previous section we can assume $\mathcal{T} = \{\top \sqsubseteq \mathcal{C}_\mathcal{T}\}$ to consist of one axiom only and both $\mathcal{C}$ and $\mathcal{C}_\mathcal{T}$ to be in negation normal form.

We assume the interpretation of $\mathcal{C}$ to contain one element. Since every element in the domain must also be in $\top^I = \Delta_I$ and thus in ${\mathcal{C}_\mathcal{T}}^I$ we also adopt this. Starting from this initial *completion system* we stepwise expand the interpretation to arrive at a model of $\mathcal{C}$ and $\mathcal{C}_\mathcal{T}$ or at an un-mistakable sign for non-satisfiability.

Instead of keeping the concepts and adding elements to their respective interpretations, we keep domain elements and store the concepts they are in. This is done with the help of a tree whose nodes represent the elements while edges represent roles. One element being in a special concept implies this same element to be in other concepts, too, or other elements to be connected to the first via roles and being in special concepts, or - this being new for concrete domains - some constraints to be added to a fuzzy `Rel`-network that is also kept. The choice of what to do is based on the structure of the given concept.

Example

Assume Alice $\in \Delta_I$ to be a domain element and a concept $C = \texttt{Woman} \sqcap \exists \mathit{hasChild}.\texttt{Man}$. Now let Alice $\in C^I$. The definition of $\mathcal{ALC}(\mathcal{D})$-interpretations implies

- Alice $\in \texttt{Woman}^I$

and

- Alice $\in (\exists \mathit{hasChild}.\texttt{Man})^I$
- which implies $\exists \delta \in \texttt{Man}^I$ such that $(\text{Alice}, \delta) \in \mathit{hasChild}^I$

The tableau algorithm would then just assume those facts and consequently add `Woman` and $\exists \mathit{hasChild}.\texttt{Man}$ to the set of concepts that Alice is in. It further introduces such δ, say Bob $\in \Delta_I$, makes `Man` a concept Bob is in, and connects Alice and Bob via *hasChild*.

The completion of the interpretation will be relying on rules that explain which concepts imply which extension of I and the sets of concepts the elements of Δ_I are in.

As described by now, the approach would work fine as long as the tree that is created stays finite. This, however, cannot be assumed from scratch. To ensure termination

and thus a finite tree, a *blocking*-mechanism is introduced, that detects tree-nodes which would result in equal (up to isomorphism) sub-trees, could thus be handled equally, and are consequently considered only once. When, afterwards, deriving the interpretation from this tree the domain elements cannot simply be read off the tree nodes anymore. The finite tableau tree must be unravelled. Where we detected isomorphic sub-trees and thus stopped, this isomorphism must be made explicit by taking copies of those isomorphic sub-trees and re-integrating them where they belong. For this purpose we use so-called *chains* which essentially resemble the end-nodes of paths in the infinite unravelled tree being themselves assembled from shorter paths and hooked together at the blocked nodes.

After explaining the completion system we work on, which basically reflects the created model, we continue the proof of decidability of concept satisfiability with the proof of correctness for the presented algorithm. Appendix A provides a detailed example for the application of the algorithm on a (simple) TBox and concept.

5.1. Data Structures

The tableau algorithm will work on a *completion system* $\mathcal{S} = (\mathfrak{T}, \mathcal{N}, \Sigma)$ consisting of a *tableau tree* $\mathfrak{T}$, a finite fuzzy `Rel`-network $\mathcal{N}$, and a the *signature* $\Sigma \subseteq 2^{\underline{\mathbf{1}}}$ as a set of sets of degrees that can be part of constraints in $\mathcal{N}$. During the application of the algorithm such completion system is altered in either element.

Definition 5.1.1 (*Tableau Tree*)

Let $\mathfrak{T} = (V, E, \mathcal{L})$ be a labelled tree with the set of nodes $V = V_A \cup V_C$ consisting of two disjoint subsets $V_A \cap V_C = \emptyset$ representing abstract *and* concrete *nodes, respectively. The set of edges $E \subseteq (V_A \times V_A) \cup (V_A \times V_C)$ shall not contain outgoing edges from concrete nodes. Further the labelling function $\mathcal{L}$ assigns a set of concepts to each abstract node, a role name to each edge that ends in an abstract node, and a concrete feature to each edge ending in a concrete node:*

$$v \in V_A : \mathcal{L}(v) \subseteq sub(\mathcal{C}, \mathcal{C}_\mathcal{T}) \tag{5.1.1.1}$$

$$v, w \in V_A : \mathcal{L}(v, w) \in N_R \tag{5.1.1.2}$$

$$v \in V_A, c \in V_C : \mathcal{L}(v, c) \in N_{cF} \tag{5.1.1.3}$$

In the first line $sub(\mathcal{C}, \mathcal{C}_\mathcal{T})$ denotes the set of sub-concepts of $\mathcal{C}$ and $\mathcal{C}_\mathcal{T}$ (cf. Definition 4.4).

The tableau tree will be initialized with one single node $v_0 \in V$ such that $\mathcal{L}(v_0) = \{C, \mathcal{C}_\mathcal{T}\}$.

In some steps the tree will be extended with new nodes i.e. the algorithm takes a fresh node name $x \notin V$ and an edge name and type. Then, depending on the current state of $\mathfrak{T}$,

it tries to add and connect this new node to the old tree. Since this extension depends on the type of edge that will connect the fresh node to the 'old' tree, the extension procedure is subdivided into three cases:

Definition 5.1.2 (*Tableau Tree Extension*)

Given a tableau tree $\mathfrak{T} = (V, E, \mathcal{L})$, a node $v \in V_A$, a role or concrete feature $r \in N_R \cup N_{cF}$, and a fresh node name $x \notin V$ let $\textsc{Ext}(\mathfrak{T}, v, r, x)$ *extend $\mathfrak{T}$ as specified in Table 5.1.1.*

rule	precondition → action
$\textsc{Ext}_{sR}$	If $r \in N_{sR}$
	→ add x to V_A, (v, x) to E and set $\mathcal{L}(x) = \{\mathcal{C}_\mathcal{T}\}$, $\mathcal{L}(v, x) = r$
$\textsc{Ext}_{aF+}$	If $r \in N_{aF}$ and there is no edge $(v, c) \in E$ with $\mathcal{L}(v, c) = r$ for any $c \in V_A$ → add x to V_A, (v, x) to E and set $\mathcal{L}(x) = \{\mathcal{C}_\mathcal{T}\}$, $\mathcal{L}(v, x) = r$
$\textsc{Ext}_{cF+}$	If $r \in N_{cF}$ and there is no edge $(v, c) \in E$ with $\mathcal{L}(v, c) = r$ for any $c \in V_C$
	→ add x to V_C, (v, x) to E and set $\mathcal{L}(v, x) = r$
$\textsc{Ext}_{aF\circlearrowleft}$	If $r \in N_{aF}$ and there exists an edge $(v, w) \in E$ such that $\mathcal{L}(v, w) = r$
	→ rename w in $\mathfrak{T}$ to x
$\textsc{Ext}_{cF\circlearrowleft}$	If $r \in N_{cF}$ and there exists an edge $(v, w) \in E$ such that $\mathcal{L}(v, w) = r$
	→ rename w in $\mathfrak{T}$ to x

Table 5.1.1: Tableau Tree Extension Function

The extension rules are chosen to obey the functionality of features. When adding a fresh node via a standard role, it is just added and connected as one would expect. In case the connection is triggered by an abstract feature or a concrete feature the extension process ensures that there still is at most one successor of an abstract node for each feature. This is yielded by renaming the existing successor if there is one. Otherwise, i.e. in case no child node exists that is connected via an edge with the desired label, we can just add the node and edge normally. In the algorithm we only use $\textsc{Ext}(\mathfrak{T}, v, r, x)$ and assume that the proper rule from the table is chosen. Also notice, that every newly introduced node's label is initially holding $\mathcal{C}_\mathcal{T}$ and none of the rules ever deletes a concept from a label.

As the second data structure there is a finite fuzzy Rel-network $\mathcal{N}$ that is constructed using the concrete nodes in the tableau tree $\mathfrak{T}$ i.e. $V_{\mathcal{N}} = V_C$. This network will be equipped with constraints during the application of the algorithm. For each abstract node $v \in V_A$ we define a *local network* that contains all constraint for which at least one member is a concrete successor of the node itself: $\mathcal{N}(v) = \{(a, b, R, \sigma) \in \mathcal{N} \mid a, b \in V_C, (v, a) \in E \vee (v, b) \in E\}$. We call the local networks of two abstract nodes $v, w \in V_A$ *isomorphic*, if there exist bijective mappings $\mu_a : V_{\mathcal{N}(a)} \to V$ and $\mu_b : V_{\mathcal{N}(b)} \to V$ such that $\mu_a(\mathcal{N}(a)) = \mu_b(\mathcal{N}(b))$. This is denoted as $\mathcal{N}(a) \sim \mathcal{N}(b)$. We assume $\mathcal{N}$ to be in normal form in every state of $\mathcal{S}$.

The unravelling of $\mathfrak{T}$ may produce and infinite set of copies of $\mathcal{N}$. For the proof of correctness it will thus be shown, that it is possible, to patch copies together, creating Net_I as needed for the interpretation of $\mathcal{T}$.

5.2. Completion Rules

Let the initial completion system be $\mathcal{S}_0 = (\mathfrak{T}_0, \mathcal{N}_0, \Sigma_0)$ with $\mathfrak{T}_0$ initialized as shown above, $\mathcal{N}_0 = \emptyset$, and $\Sigma_0 = \{\underline{\mathbf{1}}\}$. We will go through step-wise refinement and expansion of $\mathcal{S}$. This is yielded by rules that are applied on the current state of the completion system, expanding the labels in the tableau tree, the tableau tree itself, or the Rel-network and the signature. Table 5.2.1 shows those rules. In this table '$C \in \mathcal{L}(a)$' for a concept $C \in sub(\mathcal{C}, \mathcal{C}_\mathcal{T})$ is used as an abbreviation for '$\mathcal{L}(a)$ contains a concept of the form C' where – in the construction of C – $D_1, D_2 \in sub(\mathcal{C}, \mathcal{C}_\mathcal{T})$ are concepts, $r \in N_R$ is a role name, $r_1, r_2 \in N_{aF}$ are abstract features, $f_1, f_2 \in N_{cF}$ are concrete features, $R \in \mathtt{Rel}$ is a concrete domain relation, and $\sigma \subseteq \underline{\mathbf{1}}$ is a set of fuzzy degrees.

When adding a constraint to $\mathcal{N}$ i.e. when one of the rules $R_{\mathfrak{o}}$ or $R_{\mathfrak{k}}$ is used, we must update Σ and add several tautological constraints to $\mathcal{N}$ in order to keep its normal form. The latter is achieved by the normalization presented in the proof of Lemma 3.1.6 on page 20. For the former: when a constraint (x, y, R, σ) is added to $\mathcal{N}$ then set

$$\Sigma \leftarrow \Sigma \cup \{\psi \cap \sigma \mid \psi \in \Sigma\}$$

If during rule application an obvious contradiction arises in the label of an abstract node, or the network $\mathcal{N}$ becomes unsatisfiable, no future rule can withdraw that fact. We thus include a clash-detection in the algorithm that breaks operation at such points.

Definition 5.2.1 ($\mathcal{ALC}(\mathcal{D})$-*clash*)

A clash *in a set $\mathcal{L} \subseteq \mathcal{C}_{\mathcal{ALC}(\mathcal{D})}$ of $\mathcal{ALC}(\mathcal{D})$ concepts is an obvious contradiction of the form*

$$C, \neg C \in \mathcal{L}$$

for a concept $C \in N_C$. We say there is a clash *in a tableau tree $\mathfrak{T}$ if there exists an abstract node $v \in V_A$ such that $\mathcal{L}(v)$ contains a clash.*

rule	precondition $\rightarrow$ action
$R_{\sqcap}$	$D_1 \sqcap D_2 \in \mathcal{L}(v)$ but $\{D_1, D_2\} \not\subseteq \mathcal{L}(v)$
	$\rightarrow$ add D_1 and D_2 to $\mathcal{L}(v)$
$R_{\sqcup}$	$D_1 \sqcup D_2 \in \mathcal{L}(v)$ but $\{D_1, D_2\} \cap \mathcal{L}(v) = \emptyset$
	$\rightarrow$ non-deterministically choose $D \in \{D_1, D_2\}$ and add D to $\mathcal{L}(v)$
$R_{\exists}$	$\exists r.D \in \mathcal{L}(v)$ but there is no abstract node $w \in V_A$ such that $(v, w) \in E$, $\mathcal{L}(v, w) = r$, and $D \in \mathcal{L}(w)$
	$\rightarrow$ choose a fresh abstract node x set $\mathfrak{T} \leftarrow \textsc{Ext}(\mathfrak{T}, v, r, x)$ and add D to $\mathcal{L}(x)$
$R_{\forall}$	$\forall r.D \in \mathcal{L}(v)$ but there is an abstract node $w \in V_A$ such that $(v, w) \in E$ and $\mathcal{L}(v, w) = r$ while $D \notin \mathcal{L}(w)$
	$\rightarrow$ add D to $\mathcal{L}(w)$
$R_{\mathfrak{Ə}}$	$\exists(r_1 f_1, r_2 f_2, R, \sigma) \in \mathcal{L}(v)$ but there are no nodes $w_1, w_2 \in V_A$, $x_1, x_2 \in V_C$ such that for $i \in \{1, 2\}$: $(v, w_i), (w_i, x_i) \in E$, $\mathcal{L}(v, w_i) = r_i, \mathcal{L}(w_i, x_i) = f_i$, and $(x_1, x_2, R, \sigma') \in \mathcal{N}$ with $\sigma' \subseteq \sigma$
	$\rightarrow$ choose fresh abstract nodes w_1, w_2 and fresh concrete nodes x_1, x_2 and set $\mathfrak{T} \leftarrow \textsc{Ext}(\mathfrak{T}, v, r_i, w_i) \leftarrow \textsc{Ext}(\mathfrak{T}, w_i, f_i, x_i)$ for $i \in \{1, 2\}$. Then add (x_1, x_2, R, σ) to $\mathcal{N}$
$R_{\mathfrak{Ɐ}}$	$\forall(r_1 f_1, r_2 f_2, R, \sigma) \in \mathcal{L}(v)$ and there are nodes $w_1, w_2 \in V_A$, $x_1, x_2 \in V_C$ such that for $i \in \{1, 2\}$: $(v, w_i), (w_i, x_i) \in E$, $\mathcal{L}(v, w_i) = r_i, \mathcal{L}(w_i, x_i) = f_i$, but there is no $(x_1, x_2, R, \sigma') \in \mathcal{N}$ with $\sigma' \subseteq \sigma$
	$\rightarrow$ add (x_1, x_2, R, σ) to $\mathcal{N}$
rules are applied on the completion system $\mathcal{S} = (\mathfrak{T}, \mathcal{N}, \Sigma)$ and an abstract node $v \in V_A$ and extend $\mathfrak{T}$ or $\mathcal{N}$, the instance of Ext is chosen in regard to the types of nodes and edges	

Table 5.2.1: Tableau Rules for use in Algorithm 5.2.1

To decide about satisfiability of $\mathcal{C}$ w.r.t. $\mathcal{T} = \{\top \sqsubseteq \mathcal{C}_\mathcal{T}\}$ we invoke Algorithm 5.2.1 with the initial configuration of the completion system i.e. $\mathfrak{T} = (\{v_0\}, \emptyset, \{v_0 \mapsto \{\mathcal{C}\}\}), \mathcal{N}_0 = \emptyset, \Sigma_0 = \{\underline{\mathbf{1}}\}, \mathcal{S}_0 = (\mathfrak{T}_0, \mathcal{N}_0, \Sigma_0)$. So, the concept should be satisfiable w.r.t. the TBox if and only if $\textsc{Sat}(\mathcal{S}_0, \mathcal{C}_\mathcal{T})$ returns '*satisfiable*'.

Algorithm 5.2.1 Satisfiability of $\mathcal{C}$ w.r.t. $\mathcal{T}$

Require: $\mathcal{ALC}(\mathcal{D})$ concepts $\mathcal{C}_\mathcal{T}$, completion system $\mathcal{S} = (\mathfrak{T}, \mathcal{N}, \Sigma)$, rules from Table 5.2.1

```
function SAT(S, C_T)
    if T contains a clash OR N is not satisfiable then
        return unsatisfiable
    else
        if any rule R_x is applicable for some node v ∈ V then
            let S⁺ result from application of R_x on v and S
            return SAT(S⁺, C_T)
        else
            return satisfiable
        end if
    end if
end function
```

It should be obvious that by now the algorithm may run infinitely often without arriving at a clash or complete network. This is because any newly introduced node can be chosen for rule application. To avoid this infinity we introduce a means of cycle detection, commonly named *blocking* to find isomorphic nodes in $\mathcal{S}$. Isomorphism must here be seen relative to possible interpretations. We call two nodes isomorphic if their respective sub-trees are equal up to the connected local networks.

Definition 5.2.2 (*Blocking*)

We say a node $v \in V_A$ directly blocks *a node $w \in V_A$ if*

- *there exists a set of nodes $v_1, \cdots, v_n \in V_A$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$ with $1 \leq i < n$ and $v = v_1$, $w = v_n$ i.e. there is a path from v to w in $\mathfrak{T}$*
- $\mathcal{L}(a) = \mathcal{L}(b)$
- $\mathcal{N}(a) \sim \mathcal{N}(b)$

A node $w \in V_A$ is indirectly blocked *by a node $v \in V_A$ if v is directly blocked and there exists a path from v to w in $\mathfrak{T}$.*

A node is blocked *if it is directly or indirectly blocked.*

5.3. Proof of Correctness

Let the algorithm apply rules only on unblocked nodes. This necessarily leads to finiteness of the tree.

5.3.1. Termination

A closer look at the completion rules unveils some first results:

Corollary 5.3.1

1. *every rule does expand*
   a) *the set of nodes and edges each by one element, or*
   b) *the label of a node by at most two concepts, or*
   c) *the fuzzy* `Rel`*-network by one constraint*
2. *no rule ever erases a formerly added element of $\mathcal{S}$, neither a node nor an edge nor a concept from a node's label. No edge's label is ever changed.*
3. *a constraint that is once added to $\mathcal{N}$ is only altered in the set of membership degrees. Those can only become smaller i.e. if for a given constraint $(a, b, R, \sigma') \in \mathcal{N}$ at some point during rule application we have $\sigma' \subseteq \sigma$ for some σ, every instance $(a, b, R, \sigma'') \in \mathcal{N}$ with the same a, b, R at a later point satisfies $\sigma'' \subseteq \sigma' \subseteq \sigma$*
4. *every constraint in $\mathcal{N}$ holds a set of possible degrees that is in Σ*

We will now show, that the tree always stays finite because:

1. the number of successors of a single node is finite
2. the maximal path length is finite

For (1) observe that successor nodes are only introduced by the $R_\exists$ and the $R_\ominus$-rules, both adding a finite number (1 or at most 2, respectively) of abstract nodes. The number of $\exists$-symbols in the label of a node thus bounds the number of created abstract successors. Meanwhile the number of concrete successors is bounded by the number of concrete $\exists$-terms in the labels of the node in focus and its direct predecessor, which also are finite.

Towards (2): since new abstract nodes are only introduced after unblocked nodes, it is sufficient to show that in every path of $\mathfrak{T}$ there must necessarily arise a blocking situation. Since $\mathcal{L}(a) \subseteq sub(\mathcal{C}, \mathcal{C}_\mathcal{T})$ for any abstract node $a \in V_A$ and $sub(\mathcal{C}, \mathcal{C}_\mathcal{T})$ is finite, no path can be infinitely long without holding two nodes with the same label. Further the local

networks are consisting of only such constraints that are built from the label at that very node or its direct predecessor which leads to finitely many different local networks. A path of abstract nodes is succeeded by paths of concrete nodes with a length of at most one. Every path in a tableau tree is finite.

The fuzzy `Rel`-network $\mathcal{N}$ is also necessarily finite: Since $V_C = V_\mathcal{N}$, $\texttt{Rel}_\mathcal{N}$ and Σ as the set of possible degree-sets are finite, there can only be finitely many different constraints in $\mathcal{N}$. Consequently all of the local networks must also be finite, and there cannot be infinitely many different local networks.

Reconsidering the results in Corollary 5.3.1, we can now state:

1. due to the size of labels and the number of successors of nodes being bounded by $|sub(\mathcal{C}, \mathcal{C}_\mathcal{T})|$ and the finiteness of the tableau tree, there can only be a finite number of non-concrete-term-rules applied

2. Furthermore, the $R_\ominus$ and the $R_\mathfrak{C}$-rule can only be applied once per node and concept: both rules eventually add a constraint (x, y, R, σ') to the network such that $\sigma' \subseteq \sigma$ for σ being the set used in the concrete term triggering rule application. Any further application of any concrete-term-rule can only decrease the size of the degree-set held in the regarded constraint i.e. after SAT has finished there still is a constraint $(x, y, R, \psi) \in \mathcal{N}$ and $\psi \subseteq \sigma' \subseteq \sigma$. Thus the rule is never triggered again by the same node and concept.

It is thus true, that there is only a finite number of possible rule applications. The algorithm terminates.

Lemma 5.3.2

The algorithm SAT *with blocking as defined above always terminates on finite input.*

5.3.2. Soundness

To show the soundness of the algorithm it will be necessary to create an interpretation for $\mathcal{C}$ and $\mathcal{C}_\mathcal{T}$ from the completion system $\mathcal{S}$ in its state in the last application of $\text{SAT}(\mathcal{S}, \mathcal{C}_\mathcal{T})$. The construction of this interpretation is – caused by the great similarities between the approaches – in great parts taken from [LM07]. Some renaming and annotations were made to explain the approach. Where the fuzziness is an issue, however, some interventions had to be undertaken. We first present this interpretation and then prove it to be a model for the given concept and TBox.

Let the *derived interpretation* be $I_\mathcal{S} = (\Delta_{I_\mathcal{S}}, \cdot^{I_\mathcal{S}}, Net_{I_\mathcal{S}})$. For any node $v \in V_A$ we would like to have a $\hat{v} \in \Delta_{I_\mathcal{S}}$ such that $\hat{v} \in D^{I_\mathcal{S}}$ for all concepts $D \in \mathcal{L}(v)$. This can be easily

achieved but does not respect the cycles that were cut by blocking. Let a $\mathfrak{T}$-*chain* [9] be a sequence of pairs $\chi = \frac{v_1}{w_1}, \frac{v_2}{w_2}, \cdots, \frac{v_n}{w_n}$ such that for $1 \leq i \leq n : v_i, w_i \in V_A$ and

1. $(v_i, w_{i+1}) \in E$, v_{i+1} is not blocked and $w_{i+1} = v_{i+1}$

or

2. $(v_i, w_{i+1}) \in E$, and w_{i+1} is blocked by v_{i+1}

With this definition we ensure that for every $\frac{v_i}{w_i}$ in such chain $\mathcal{L}(v_i) = \mathcal{L}(w_i)$ and $\mathcal{N}(v_i) \sim \mathcal{N}(w_i)$. Further let **chains** be the set of all such $\mathfrak{T}$-chains, v_0 be the root-node of $\mathfrak{T}$ and $\text{tail}(\chi) = \frac{v_n}{w_n}$ be the last pair of nodes in a chain χ. A probably not obvious result can be derived directly from the definition. Here we use $\chi_1 \cdot \chi_2$ to denote the *concatenation* of two chains i.e.

$$\begin{aligned} &\text{if} && \chi_1 = \frac{v_{11}}{w_{11}}, \cdots, \frac{v_{1n}}{w_{1n}} \\ &\text{and} && \chi_2 = \frac{v_{21}}{w_{21}}, \cdots, \frac{v_{2m}}{w_{2m}} \\ &\text{we get} && \chi_1 \cdot \chi_2 = \frac{v_{11}}{w_{11}}, \cdots, \frac{v_{1n}}{w_{1n}}, \frac{v_{21}}{w_{21}}, \cdots, \frac{v_{2m}}{w_{2m}} \end{aligned}$$

Let the set of prefixes of a $\mathfrak{T}$-chain $\chi \in$ **chains** be defined as

$$P(\chi) = \{\chi_1 \in \textbf{chains} \mid \exists \chi_2 \in \textbf{chains} : \chi = \chi_1 \cdot \chi_2\}$$

Corollary 5.3.3

If in a completion system $\mathcal{S} = (\mathfrak{T}, \mathcal{N})$ there is at least one blocked node, then the set ***chains*** *of $\mathfrak{T}$-chains is infinite.*

Proof: For $a, b, v_1, \cdots, v_n \in V_A$ let $(v_i, v_{i+1}) \in E$ for all $n \in \mathbb{N}$ with $1 \leq i < n$ and $(a, v_1), (v_n, b) \in E$ such that a blocks b, while no v_i is blocked. Then $\chi_1 = \frac{a}{a}, \frac{v_1}{v_1}, \cdots, \frac{v_n}{v_n}, \frac{a}{b}$ is a $\mathfrak{T}$-chain. This chain can, obeying the definition, be continued by $\frac{v_1}{v_1}$ since $(a, v_1) \in E$ and v_1 is not blocked. It should now be easy to see that $\chi_2 = \chi_1 \cdot \left(\frac{v_1}{v_1}, \cdots, \frac{v_n}{v_n}, \frac{a}{b}\right)$ is also a $\mathfrak{T}$-chain. This holds for all $\chi_{i+1} = \chi_i \cdot \left(\frac{v_1}{v_1}, \cdots, \frac{v_n}{v_n}, \frac{a}{b}\right)$ ■

Now define

$$\Delta_{I_\mathcal{S}} = \left\{\chi \in \textbf{chains} \mid \chi \text{ starts with } \frac{v_0}{v_0}\right\}$$

as the domain of interest. These elements do reflect infinite paths in the infinite unravelled tree, i.e. starting in the root node we go down a path in the tree until a blocked node

[9] [LM07] called those 'paths', but to avoid mistaking them for normal paths in a tree or graph they are renamed here

is reached. This node is by definition blocked because of being isomorphic to another node earlier in the chain/path. Hence we can 'hop' to the node that blocks the regarded one and continue our path from there. This results in infinite $\mathfrak{T}$-chains of the described pattern, all starting in v_0 and ending in a tree node, whose label is then used to clarify the concepts the chain – now as a domain element – is in.

For basic concepts $D \in N_C$ and roles $r \in N_R$ we define

$$D^{I_\mathcal{S}} = \left\{ \chi \in \Delta_{I_\mathcal{S}} \mid \text{tail}(\chi) = \frac{v}{w}, D \in \mathcal{L}(v) \right\}$$
$$r^{I_\mathcal{S}} = \left\{ (\chi \cdot \frac{v_1}{w_1}, \chi \cdot \frac{v_1}{w_1} \cdot \frac{v_2}{w_2}) \in \Delta^2_{I_\mathcal{S}} \mid (v_1, w_2) \in E, \mathcal{L}(v_1, w_2) = r \right\}$$

The interpretation of the abstract part of the TBox is thus ready. It works with reusing isomorphic sub-trees at places where blocked nodes were. This unravelling process must be applied on the concrete part, too.

First we define a partial function $\pi : N_{cF} \times V_A \to V_C$ such that $\pi(f, v) = x$ if $(v, x) \in E$ and $\mathcal{L}(v, x) = f$. Due to the construction of Ext (cf. Definition 5.1.2 on page 46) this is well-defined for those abstract nodes having an f-successor in $\mathfrak{T}$. Further an equivalent mapping shall be defined for $\mathfrak{T}$-chains, which is used as an indirect definition of the set of $Net_{I_\mathcal{S}}$-variables. Let $\pi : N_{cF} \times \mathbf{chains} \to V_{Net_{I_\mathcal{S}}}$ be an injective mapping such that for a concrete feature $f \in N_{cF}$ and a chain $\chi \in \mathbf{chains}$ with $\text{tail}(\chi) = \frac{v}{w}$, it holds that $\pi(f, \chi)$ exists if and only if $\exists x \in V_C : (w, x) \in E, \mathcal{L}(w, x) = f$. For shortness we write $f(\chi)$ instead of $\pi(f, \chi)$. Then let the set of $Net_{I_\mathcal{S}}$-variables be

$$V_{Net_{I_\mathcal{S}}} = \{ f(\chi) \mid \chi \in \Delta_{I_\mathcal{S}} \}$$

We can now define

$$Net_{I_\mathcal{S}} = \left\{ (f_1(\chi_1), f_2(\chi_2), R, \sigma) \mid \text{tail}(\chi_1) = \frac{v_1}{w_1}, \text{tail}(\chi_2) = \frac{v_2}{w_2}, \right.$$
$$\left. (\pi(f_1, w_1), \pi(f_2, w_2), R, \sigma) \in \mathcal{N} \right\}$$

We would now like to show, that this network is patched together from finite satisfiable networks and thus is satisfiable due to the ω-admissibility of $\mathcal{D}$ (cf. Definition 3.3.3 on page 33). From Sat returning *satisfiable* we know that $\mathcal{N}$ is satisfiable in $\mathcal{D}$. The finiteness of $\mathcal{N}$ has been shown above. Further we define finite sub-networks of $Net_{I_\mathcal{S}}$ as a restriction of that set to the renamed prefixes of one chain:

$$Net_{I_\mathcal{S}}(\chi) = Net_{I_\mathcal{S}}|_{\left\{ f(\chi') \in V_{Net_{I_\mathcal{S}}} \mid \chi' \in P(\chi) \right\}}$$

With this definition we can show some results that lead to the satisfiability of $Net_{I_\mathcal{S}}$. First it should be clear that $Net_{I_\mathcal{S}}$ can be mapped back to $\mathcal{N}$ by using $\tau : V_{Net_{I_\mathcal{S}}} \to V_\mathcal{N}, \pi(f, \chi) \mapsto \pi(f, w)$ with $\text{tail}(\chi) = \frac{v}{w}$. With this mapping we get $\tau(Net_{I_\mathcal{S}}(\chi)) \subseteq \mathcal{N}$ for every chain χ. Which necessarily makes $Net_{I_\mathcal{S}}(\chi)$ satisfiable.

Claim 5.3.4

For two domain elements $\chi_1, \chi_2 \in \Delta_{I_\mathcal{S}}$ we have:

1. *if $\chi_2 \in P(\chi_1)$ then $Net_{I_\mathcal{S}}(\chi_2) \subseteq Net_{I_\mathcal{S}}(\chi_1)$*
2. *if $\chi_1 \notin P(\chi_2)$ and $\chi_2 \notin P(\chi_1)$ then $Net_{I_\mathcal{S}}(\chi_1) \cup Net_{I_\mathcal{S}}(\chi_2)$ is `Rel`-clash-free (cf. Definition 3.1.7 on page 21)*

Proof: For (1): From the definition of $P(\chi)$: since for every $\chi' \in P(\chi)$ we have $P(\chi') \subseteq P(\chi)$ the assumption follows directly.

For (2): Let $\chi_3 \in V_{Net_{I_\mathcal{S}}}$ be the maximal $\mathfrak{T}$-chain such that $\chi_3 \in P(\chi_1)$ and $\chi_3 \in P(\chi_2)$. Such element necessarily exists since $\frac{v_0}{v_0} \in P(\chi)$ for every chain χ. As shown above $Net_{I_\mathcal{S}}(\chi_3) \subseteq Net_{I_\mathcal{S}}(\chi_1)$ and $Net_{I_\mathcal{S}}(\chi_3) \subseteq Net_{I_\mathcal{S}}(\chi_2)$ and thus $Net_{I_\mathcal{S}}(\chi_3) \subseteq Net_{I_\mathcal{S}}(\chi_1) \cap Net_{I_\mathcal{S}}(\chi_2)$. Any constraint in $Net_{I_\mathcal{S}}(\chi_1) \setminus Net_{I_\mathcal{S}}(\chi_2)$ contains at least one variable $f(\chi_1')$ for a chain χ_1' such that $\chi_1' \notin P(\chi_2)$ due to the injectivity of π this variable cannot be in $V_{Net_{I_\mathcal{S}}(\chi_2)}$. The same holds vice versa and thus there can be no clash outside $Net_{I_\mathcal{S}}(\chi_1) \cap Net_{I_\mathcal{S}}(\chi_2)$. Since the latter is satisfiable it can contain no clash. So, the whole network $Net_{I_\mathcal{S}}(\chi_1) \cup Net_{I_\mathcal{S}}(\chi_2)$ must be clash-free. The same argument holds for a union of any number of $Net_{I_\mathcal{S}}(\chi)$-networks. ■

By now it is clear that every $Net_{I_\mathcal{S}}(\chi)$ is finite and satisfiable and that the union of any two of these networks is clash-free which makes the patchwork property applicable. It is yet left to show that we can apply the compactness property.

Claim 5.3.5

Every network $Net_{I_\mathcal{S}}|_V$ with $V \subseteq V_{Net_{I_\mathcal{S}}}$ is satisfiable.

Proof: It is sufficient to show, that for any such $V \subseteq V_{Net_{I_\mathcal{S}}}$ we can find a set $X \subseteq \{Net_{I_\mathcal{S}}(\chi) \mid \chi \in \Delta_{I_\mathcal{S}}\}$ such that $Net_{I_\mathcal{S}}|_V \subseteq \bigcup X$. Since $\bigcup X$ is satisfiable (see proof of Claim 5.3.4) then $Net_{I_\mathcal{S}}|_V$ would also be satisfiable. It should be clear that $X = \{Net_{I_\mathcal{S}}(\chi) \mid f(\chi) \in V\}$ does the job. ■

Due to ω-admissibility of $\mathcal{D}$ the network $Net_{I_\mathcal{S}}$ as defined above is satisfiable. Subsuming all gathered results including the interpretation of concrete features:

Definition 5.3.6 (*Derived Interpretation*)

From the final state of $\mathcal{S}$ in the application of SAT *we derive the interpretation $I_\mathcal{S} = (\Delta_{I_\mathcal{S}}, \cdot^{I_\mathcal{S}}, Net_{I_\mathcal{S}})$ with, for a concept name $D \in N_C$, a role name $r \in N_R$ and a concrete feature $f \in N_{cF}$:*

$$
\begin{aligned}
Net_{I_\mathcal{S}} &= \Big\{ (f_1(\chi_1), f_2(\chi_2), R, \sigma) \mid \text{tail}(\chi_1) = \frac{v_1}{w_1}, \text{tail}(\chi_2) = \frac{v_2}{w_2}, \\
&\qquad (\pi(f_1, w_1), \pi(f_2, w_2), R, \sigma) \in \mathcal{N} \Big\} \\
\Delta_{I_\mathcal{S}} &= \Big\{ \chi \in \textbf{\textit{chains}} \mid \chi \textit{ starts with } \frac{v_0}{v_0} \Big\} \\
D^{I_\mathcal{S}} &= \Big\{ \chi \in \Delta_{I_\mathcal{S}} \mid \text{tail}(\chi) = \frac{v}{w}, D \in \mathcal{L}(v) \Big\} \\
r^{I_\mathcal{S}} &= \Big\{ (\chi \cdot \frac{v_1}{w_1}, \chi \cdot \frac{v_1}{w_1} \cdot \frac{v_2}{w_2}) \in \Delta^2_{I_\mathcal{S}} \mid (v_1, w_2) \in E, \mathcal{L}(v_1, w_2) = r \Big\} \\
f^{I_\mathcal{S}} &= \Big\{ (\chi, f(\chi)) \mid \text{tail}(\chi) = \frac{v}{w}, \exists x \in V_C : (w, x) \in E, \mathcal{L}(w, x) = f \Big\}
\end{aligned}
$$

It remains to show that this interpretation is obeying complex concepts.

Claim 5.3.7

For all $\chi \in \Delta_{I_\mathcal{S}}$ with $\text{tail}(\chi) = \frac{v}{w}$ and concept $E \in sub(\mathcal{C}, \mathcal{C}_\mathcal{T})$ we have: if $E \in \mathcal{L}(v)$ then $\chi \in E^I$.

Proof: Let $E \in \mathcal{L}(v)$

- $E = D \in N_C$ true by definition
- $E = \neg D$
  since E is in NNF $D \in N_C$. From SAT returning 'satisfiable' we know $\mathcal{L}(v)$ is clash-free, thus $\chi \notin D^I$ and $\chi \in (\neg D)^I$
- $E = D_1 \sqcap D_2$
  rule $R_\sqcap$ lead to $D_1, D_2 \in \mathcal{L}(v)$ and thus by induction $\chi \in D_1^I \cap D_2^I \Rightarrow \chi \in (D_1 \sqcup D_2)^I$
- $E = D_1 \sqcup D_2$
  rule $R_\sqcup$ lead to $D_1 \in \mathcal{L}(v)$ or $D_2 \in \mathcal{L}(v)$ and thus by induction $\chi \in D_1^I \cup D_2^I \Rightarrow \chi \in (D_1 \sqcup D_2)^I$
- $E = \exists r.D$
  rule $R_\exists$ lead to the existence of $w' \in V_A$ such that $(v, w') \in E, \mathcal{L}(v, w') = r, D \in \mathcal{L}(w')$. Due to the definition of $r^{I_\mathcal{S}}$ there exists a $\chi' = \chi \cdot \frac{v'}{w'}$ such that $(\chi, \chi') \in r^{I_\mathcal{S}}$. Here either $v' = w'$ or w' is blocked by v' in both cases we have

$\mathcal{L}(v') = \mathcal{L}(w')$ and thus $D \in \mathcal{L}(v')$. By induction we get $\chi' \in D^{I_\mathcal{S}}$ and thus $\chi \in (\exists r.D)^I$

- $E = \forall r.D$
  let $(\chi, \chi') \in r^{I_\mathcal{S}}$, by definition of $r^{I_\mathcal{S}}$ we have $\chi' = \chi \cdot \frac{v'}{w'}$ with $(v, w') \in E, \mathcal{L}(v, w') = r$. Rule $R_\forall$ then lead to $D \in \mathcal{L}(w')$ with the same argument as before $D \in \mathcal{L}(v') \Rightarrow \chi' \in D^{I_\mathcal{S}} \Rightarrow \chi \in (\forall r.D)^{I_\mathcal{S}}$

- $E = \exists(P_1, P_2, R, \sigma)$
  we only consider the case where $P_i = r_i f_i$ with $r_i \in N_{aF}, f_i \in N_{cF}$ for $i \in \{1, 2\}$. The cases with shorter feature paths are simpler to show. Rule $R_\ni$ lead to the existence of $w_1, w_2 \in V_A$ with $(v, w_i) \in E, \mathcal{L}(v, w_i) = r_i$, $x_1, x_2 \in V_C$ with $(w_i, x_i) \in E, \mathcal{L}(w_i, x_i) = f_i$, and $(x_1, x_2, R, \sigma') \in \mathcal{N}$ with $\sigma' \subseteq \sigma$. Due to the definition of $r^{I_\mathcal{S}}$ we have $\chi_1, \chi_2 \in \Delta_{I_\mathcal{S}}$ with $\chi_i = \chi \cdot \frac{v_i}{w_i}$ such that $(\chi, \chi_i) \in r^{I_\mathcal{S}}$. Further the definition of $Net_{I_\mathcal{S}}$ leads to $(f_1(\chi_1), f_2(\chi_2), R, \sigma') \in Net_{I_\mathcal{S}}$ since $(\pi(f_1, w_1), \pi(f_2, w_2), R, \sigma') \in \mathcal{N}$, with $\pi(f_i, w_i) = x_i$ as defined above. The definition of $f^{I_\mathcal{S}}$ additionally leads to $(\chi_i, f_i(\chi_i)) \in f_i^{I_\mathcal{S}}$, thus we have $(\chi, \chi_i) \in r^{I_\mathcal{S}}$, $(\chi_i, f_i(\chi_i)) \in f_i^{I_\mathcal{S}}$ and $(f_1(\chi_1), f_2(\chi_2), R, \sigma') \in Net_{I_\mathcal{S}}$ which together imply $\chi \in (\exists(P_1, P_2, R, \sigma))^{I_\mathcal{S}}$

- $E = \forall(P_1, P_2, R, \sigma)$
  as before $P_i = r_i f_i$. For $\chi_1, \chi_2 \in \Delta_{I_\mathcal{S}}$ let $(\chi, \chi_i) \in r_i^{I_\mathcal{S}}, (\chi_i, f_i(\chi_i)) \in f_i^{I_\mathcal{S}}$. By definition of r^I we have $w_1, w_2 \in V_A, x_1, x_2 \in V_C$ such that $\text{tail}(\chi_i) = \frac{v_i}{w_i}, (v, w_i) \in E, \mathcal{L}(v, w_i) = r_i, (w_i, x_i) \in E, \mathcal{L}(w_i, x_i) = f_i$. Rule $R_\mathfrak{P}$ then lead to $(x_1, x_2, R, \sigma') \in \mathcal{N}$ while $\sigma' \subseteq \sigma$. With $\pi(f_i, w_i) = x_i$ and according the definition of $Net_{I_\mathcal{S}}$ this implies $(f_1(\chi_1), f_2(\chi_2), R, \sigma') \in Net_{I_\mathcal{S}}$ and thus $\chi \in (\forall(P_1, P_2, R, \sigma))^{I_\mathcal{S}}$ ■

As the last easy task it remains to show that $I_\mathcal{S}$ is an interpretation for both $\mathcal{C}$ and $\mathcal{C}_\mathcal{T}$. The former is provided by $\mathcal{C} \in \mathcal{L}(v_0)$ which implies $\frac{v_0}{v_0} \in \mathcal{C}^I$, while the latter is ensured by the addition of $\mathcal{C}_\mathcal{T}$ to each and every label of an abstract node.

Lemma 5.3.8

If SAT *returns 'satisfiable' then the concept $\mathcal{C}$ is satisfiable with respect to the TBox $\mathcal{T} = \{\top \sqsubseteq \mathcal{C}_\mathcal{T}\}$*

5.3.3. Completeness

To show the completeness of our algorithm, assume $\mathcal{C}$ to be satisfiable w.r.t. $\mathcal{T} = \{\top \sqsubseteq \mathcal{C}_\mathcal{T}\}$. Then there exists an interpretation $I = (\Delta_I, \cdot^I, Net_I)$ including a satisfiable fuzzy `Rel`-network Net_I such that I satisfies both $\mathcal{C}$ and $\mathcal{C}_\mathcal{T}$. We will use this interpretation to 'guide' the rules through the algorithm. This means where a rule is non-deterministic,

the guidance tells the decision. At the end of rule application the guided algorithm will return '$satisfiable$'. This approach is again similar to that presented in [LM07].

Definition 5.3.9 (*I-Compatibility*)

Let the completion system $\mathcal{S} = (\mathfrak{T}, \mathcal{N})$ be called I-compatible, *if there exist two mappings $\hat{\cdot} : V_A \to \Delta_I$ and $\bar{\cdot} : V_C \to V_{Net_I}$ such that for all $a, b \in V_A, x, y \in V_C, C \in sub(\mathcal{C}, \mathcal{C}_\mathcal{T}), r \in N_R, f \in N_{cF}$:*

1. $C \in \mathcal{L}(a) \Rightarrow \hat{a} \in C^I$
2. $\mathcal{L}(a, b) = r \Rightarrow (\hat{a}, \hat{b}) \in r^I$
3. $\mathcal{L}(a, x) = f \Rightarrow (\hat{a}, \overline{c}) \in f^I$
4. $(x, y, R, \sigma) \in \mathcal{N} \Rightarrow (\overline{x}, \overline{y}, R, \sigma') \in Net_I$ *with* $\sigma' \subseteq \sigma$

The initial tableau tree $\mathfrak{T}_0 = (\{v_0\}, \emptyset, \{v_0 \mapsto \{\mathcal{C}\}\})$ is obviously I-compatible. The guidance, that leads through the algorithm will consist of the choice of rule instances in the way, that I-compatibility is preserved through rule application. We will now show, that this is possible for each rule.

- $R_\sqcap$
  Assume $C_1 \sqcap C_2 \in \mathcal{L}(a)$. The rule adds C_1 and C_2 to $\mathcal{L}(a)$, which is I-compatible due to: $C_1 \sqcap C_2 \in \mathcal{L}(a) \Rightarrow \hat{a} \in (C_1 \sqcap C_2)^I \Rightarrow \hat{a} \in C_1^I \wedge \hat{a} \in C_2^I$

- $R_\sqcup$
  Assume $C_1 \sqcup C_2 \in \mathcal{L}(a)$ then $\hat{a} \in (C_1 \sqcup C_2)^I$ and $\hat{a} \in C_1^I \vee \hat{a} \in C_2^I$. Then the rule can add C_i to $\mathcal{L}(a)$ for $i \in \{1, 2\}$ and $\hat{a} \in C_i$. Then $\mathcal{S}$ is I-compatible.

- $R_\exists$
  Assume $\exists r.C \in \mathcal{L}(a)$, then $\hat{a} \in (\exists r.C)^I$ thus $\exists c \in C^I, (\hat{a}, c) \in r^I$. Choose b as the introduced variable, such that $\hat{b} = c$. Then $\mathcal{L}(a, b) = r$ and $C \in \mathcal{L}(b)$ and $\mathcal{S}$ is I-compatible.

- $R_\forall$
  Assume $\forall r.C \in \mathcal{L}(a)$. The rule adds C to the label of every $b \in V$ for which $\mathcal{L}(a, b) = r$. This keeps I-compatibility because: $\forall r.C \in \mathcal{L}(a) \Rightarrow \hat{a} \in (\forall r.C)^I$ and thus for all $c \in \Delta_I$: if $(\hat{a}, c) \in r^I$ then $c \in C$ thus $\forall b \in V : \hat{b} = c \Rightarrow C \in \mathcal{L}(b)$

- $R_\ominus$
  As before we consider the most difficult case only where $P_i = r_i f_i$ for $i \in \{1, 2\}$. Assume $\exists (P_1, P_2, R, \sigma) \in \mathcal{L}(a)$. Then $\hat{a} \in (\exists (P_1, P_2, R, \sigma))^I$ and thus $\exists v_1, v_2 \in \Delta_I, x_1, x_2 \in V_{Net_I} : (\hat{a}, v_i) \in r_i^I \wedge (v_i, x_i) \in f_i^I \wedge (x_1, x_2, R, \sigma') \in Net_I$ with $\sigma' \subseteq \sigma$.

Choose $b_1, b_2 \in V_A, y_1, y_2 \in V_C$ as the introduced variables such that $\hat{b_i} = v_i, \overline{y_i} = x_i$. Then $\mathcal{L}(a, b_i) = r_i, \mathcal{L}(b_i, y_i) = f_i, (y_1, y_2, R, \sigma) \in \mathcal{N}$, which keeps $\mathcal{S}$ I-compatible.

- $R_{\mathfrak{P}}$
  Again $P_i = r_i f_i$ for $i \in \{1, 2\}$. Assume $\forall(P_1, P_2, R, \sigma) \in \mathcal{L}(a)$. If there exist $b_1, b_2 \in V_A, y_1, y_2 \in V_C$ such that $(a, b_i) \in E, \mathcal{L}(a, b_i) = r_i, (b_i, y_i) \in E, \mathcal{L}(b_i, y_i) = f_i$ then the rule adds (y_1, y_2, R, σ) to $\mathcal{N}$. This is I-compatibility-preserving since: $\forall(P_1, P_2, R, \sigma) \in \mathcal{L}(a) \Rightarrow \hat{a} \in (\forall(P_1, P_2, R, \sigma))^I$ and thus $\forall v_1, v_2 \in \Delta_I, x_1, x_2 \in V_{Net_I}$: if $(\hat{a}, v_i) \in r_i^I \wedge (v_i, x_i) \in f_i^I$ then $(x_1, x_2, R, \sigma') \in Net_I$ with $\sigma' \subseteq \sigma$. Thus for all $b_1, b_2 \in V_A, y_1, y_2 \in V_C$: if $\hat{b_i} = v_i, \overline{y_i} = x_i$ then $(\overline{y_1}, \overline{y_2}, R, \sigma') \in Net_I$

We have now shown, that it is always possible to choose the newly added nodes and constraints in a way that preserves I-compatibility. The advantage of I-compatible guidance lies in the following insight.

Claim 5.3.10

In an I-compatible completion system there can be no $\mathcal{ALC}(\mathcal{D})$-clash in any node's label, nor can $\mathcal{N}$ ever become unsatisfiable.

Proof: For the first part assume $\{C, \neg C\} \subseteq \mathcal{L}(v)$ for a concept name $C \in N_C$ and an abstract node $v \in V_A$. The I-compatibility of $\mathcal{S}$ leads to $\hat{v} \in C^I$ and $\hat{v} \in (\neg C)^I = \Delta_I \setminus C^I$ which would lead to the obvious contradiction $\hat{v} \in C^I \cap (\Delta_I \setminus C^I) = \emptyset$.

For the second consider $\overline{\mathcal{N}} = \{(\overline{x}, \overline{y}, R, \sigma) \mid (x, y, R, \sigma) \in \mathcal{N}\}$. From $(x, y, R, \sigma) \in \mathcal{N} \Rightarrow (\overline{x}, \overline{y}, R, \sigma')$ with $\sigma' \subseteq \sigma$ for all x, y, R it directly follows, that the satisfiability of Net_I implies that of $\overline{\mathcal{N}}$ and consequently that of $\mathcal{N}$. ■

Since SAT always terminates and due to the above Claim never arrives at an obvious inconsistency, it must finally return '$satisfiable$'

Lemma 5.3.11

If $\mathcal{C}$ is satisfiable w.r.t. $\mathcal{T} = \{\top \sqsubseteq \mathcal{C}_\mathcal{T}\}$ then SAT *returns '$satisfiable$'*

5.3.4. Results

The Lemmas 5.3.2, 5.3.8, and 5.3.11 have – in respective order – shown that the function SAT as defined in Algorithm 5.2.1 on page 49 is terminating, sound and complete. That shows, that general $\mathcal{ALC}(\mathcal{D})$-TBoxes with an ω-admissible fuzzy constraint system $\mathcal{D}$ as a concrete domain provide a means of deciding the satisfiability of concepts w.r.t. such TBoxes.

Theorem 5.3.12

If $\mathcal{D} = (V, \texttt{Rel}, \underline{\mathbf{1}}, \mathfrak{M})$ is a fuzzy constraint system, that is ω-admissible, then in $\mathcal{ALC}(\mathcal{D})$ the satisfiability of concepts with respect to general TBoxes is decidable.

It has been the goal of this paper to present an algorithm that decides concept satisfiability in $\mathcal{ALC}(\mathcal{D})$. This algorithm does, however, seem to be very complicated and thus not applicable for real-life purposes. To optimize the run-time of the program several modifications are imaginable.

1. the way in which *blocking* is defined is much more strict and thus triggering later than in earlier papers
   a) local networks could be defined with holding only the direct concrete successors of the abstract node
   b) $\mathcal{L}(w) \subseteq \mathcal{L}(v)$ may be sufficient for v to block w
   c) one could prefer *anywhere blocking* that accepts a node blocking an isomorphic other node that is not in the same tree-path but in a later generation
2. an early `Rel`-clash-detection could be added, handling concepts like

   $$\exists(P_1, P_2, R, \sigma) \sqcap \exists(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma)$$

   or

   $$\exists(P_1, P_2, R, \sigma) \sqcap \forall(P_1, P_2, R, \underline{\mathbf{1}} \setminus \sigma)$$

   which both necessarily result in a `Rel`-clash in Net_I and hence make the latter unsatisfiable
3. instead of setting a breakpoint if $\mathcal{N}$ is unsatisfiable this could be done for local networks $\mathcal{N}(v)$ (with the yet mentioned possible restricions)

These seem easy and provable optimizations. In addition the assumption of its completion could probably enhance earlier unsatisfiability of $\mathcal{N}$. This completion, however, results in the necessity of a re-definition of *blocking*: due to the possible existence of overcountably many interpretations and thus completions of one network we cannot assume isomorphic networks to gain isomorphic completions. It might nevertheless be possible to achieve such by further restriction of $\mathcal{D}$ or $\mathcal{ALC}(\mathcal{D})$.

As reasoning over fuzzy `Rel`-networks is provided by an oracle-like black box, we cannot even fix the overall complexity of the algorithm. In [Lut04] Lutz has shown an NExpTime upper bound for concept satisfiability in $\mathcal{ALCRPI}(\mathcal{D})$ with admissible concrete domains, acyclic TBoxes, and $\mathcal{D}$-satisfiability being in NP. Further modifications of requirements could even result in an ExpTime upper bound for reasoning as it exists in $\mathcal{ALC}$ ([DM00],[BCM+03]).

6. Conclusion

We have presented the notion of *fuzzy constraint systems*. These have then been integrated as a concrete domain with the description logic $\mathcal{ALC}$ forming $\mathcal{ALC}(\mathcal{D})$.

Fuzzy constraint systems evolve from crisp constraint systems by simple addition of sets of possible *membership degrees* to `Rel`-constraints. The notion of *interpretations* in such fuzzy constraint system was introduced, which intuitively map relations from that system to *fuzzy membership functions*. The *satisfiability* of fuzzy `Rel`-networks was based on a renaming of such network to fit the variable set of a $\mathcal{D}$-*model* and a common interpretation of both, the model and the renamed network.

The connection of $\mathcal{ALC}$ to such fuzzy constraint system was performed via *feature paths* and *concrete terms*, requiring special constraints in a fuzzy `Rel`-network that must be provided by an $\mathcal{ALC}(\mathcal{D})$-interpretation.

As the main result of this thesis we could prove the decidability of concept satisfiability with respect to general TBoxes in that language. For this purpose, a *tableau algorithm* was presented that correctly decides this satisfiability. Besides well-known tableau rules, two new rules were added to fulfill the requirements initiated by the concrete terms. These rules eventually add special constraints to the attached fuzzy `Rel`-network. A *blocking* mechanism was introduced that, based on the changed conditions, ensured termination of the procedure. Provided the algorithm returned 'satisfiable' we could construct a model for the given TBox that also was a model for the input concept. The completeness of the algorithm was shown via a guidance of the non-deterministic tableau rules, that was based on a model for input concept and TBox. The guidance necessarily lead to the returning of 'satisfiable' as long such a model existed. The algorithm section also provided a list of possible optimization techniques based on modified blocking or further restrictions on the fuzzy constraint system.

To reach the goal of decidable concept satisfiability several restrictions and reductions on concrete domains and description logics had to be adopted. We could prove that the requirement of only binary relations in a fuzzy constraint system does not restrict their expressive power. The existence of an equivalence preserving normal form could also be shown for every `Rel`-network. The use of functional roles i.e. abstract features and concrete features in $\mathcal{ALC}$ as well as the restriction of feature path length are quite common although normal binary relations and arbitrary (finite) length paths seem as well applicable for the concrete domain interaction. A (possibly) severe restriction came into account by founding the algorithm on ω-admissible constraint systems.

This notion requires the decidability of satisfiability of finite fuzzy `Rel`-networks as well as the satisfiability of unions of such finite networks that agree on their interface part (patchwork property) and the satisfiability of an infinite network being deducible from the satisfiability of all of its sub-networks bearing a finite variable set (compactness property). Reasoning in presence of an ω-admissible fuzzy constraint system presupposes

the existence of such a system. It is still open whether this can be assumed and, if yes, whether fuzzy constraint systems are employable in real-life applications.

Further examinations are yet imaginable of the integration of fuzzy concrete domains with fuzzy $\mathcal{ALC}$, as suggested by [Str05]. Other description logics may also be equipped with fuzzy concrete domains. This was already suggested for less expressive DLs like $\mathcal{EL}^{++}$ (see [MSS+12]). Mightier description logics like $\mathcal{SHIQ}$ may, nevertheless, also gain further expressive power by such extension. The presented syntax and semantics and the decidability result in this present thesis can be seen as first steps that encourage further research in those directions.

References

[Baa09] Franz Baader. Description logics. In *Reasoning Web: Semantic Technologies for Information Systems, 5th International Summer School 2009*, volume 5689 of *Lecture Notes in Computer Science*, pages 1–39. Springer–Verlag, 2009.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logics Handbook: Theory, Implementations, and Applications*. Cambridge University Press, 2003.

[BH91] Franz Baader and Philipp Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *IJCAI*, 1991.

[BP12] Stefan Borgwardt and Rafael Peñaloza. Undecidability of Fuzzy Description Logics. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 232–242, Rome, Italy, 2012. AAAI Press.

[BS08] Fernando Bobillo and Umberto Straccia. fuzzyDL: An expressive fuzzy description logic reasoner. In *Fuzzy Systems, 2008. FUZZ-IEEE 2008.(IEEE World Congress on Computational Intelligence). IEEE International Conference on*, pages 923–930. IEEE, 2008.

[DM00] Francesco M. Donini and Fabio Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *ARTIFICIAL INTELLIGENCE*, 124(1):87–138, 2000.

[LM04] Carsten Lutz and Maja Miličić. Description Logics with Concrete Domains and Functional Dependencies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, 2004.

[LM07] Carsten Lutz and Maja Miličić. A Tableau Algorithm for Description Logics with Concrete Domains and General TBoxes. *Journal of Automated Reasoning*, 38(1–3):227–259, 2007.

[Lut04] Carsten Lutz. NExpTime-complete Description Logics with Concrete Domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[MS57] P. S. Mostert and A. L. Shields. On the structure of semigroups on a compact manifold with boundary. In *Annals of Mathematics*, volume 65, pages 117–143, 1957.

[MSS+12] Theofilos Mailis, Giorgos Stoilos, Nikolaos Simou, Giorgos Stamou, and Stefanos Kollias. Tractable reasoning with vague knowledge using fuzzy $\mathcal{EL}^{++}$. *Journal of Intelligent Information Systems*, 39(2):399–440, 2012.

[SB08] Umberto Straccia and Fernando Bobillo. Mixed integer programming, general concept inclusions and fuzzy description logics. *Mathware & Soft Computing*, 14(3):247–259, 2008.

[Str05] Umberto Straccia. Description Logics with Fuzzy Concrete Domains. In Fahiem Bachus and Tommi Jaakkola, editors, *21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, Edinburgh, Scotland, 2005. AUAI Press.

[Yen91] John Yen. Generalizing term subsumption languages to fuzzy logic. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 1*, IJCAI'91, pages 472–477, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[Zad65] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

Appendix A Tableau Algorithm Example

In this appendix we want to illustrate the procedure from our algorithm and the idea behind the interpretation presented in its proof of correctness. We will first provide a full application of SAT on a given TBox and concept. The tableau tree in several states will be shown together with tables holding the labels of single nodes and the network $\mathcal{N}$.

Consider the following TBox:

$$\mathcal{T} = \{D \sqsubseteq \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)\}$$

and the concept

$$\mathcal{C} = D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$$

First we have to detect the concept $\mathcal{C}_\mathcal{T}$ i.e. to bring $\mathcal{T}$ in normal form with a single axiom. This normal form is:

$$\top \sqsubseteq \mathcal{C}_\mathcal{T}$$

with

$$\mathcal{C}_\mathcal{T} = \neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$$

Now we can create the initial completion system $\mathcal{S} = (\mathfrak{T}, \mathcal{N}, \Sigma)$ as presented in the proof and invoke $\text{SAT}(\mathcal{S}, \mathcal{C}_\mathcal{T})$. We consider the root node of $\mathfrak{T}$ to be v_0. The next pages will contain the current state of the completion system including a drawing of the tableau tree, the mapping from nodes to their labels, the fuzzy `Rel`-network $\mathcal{N}$, and, most important, the sequence of rules that lead from the last state to the current with the node and concept that triggered their application. We skip depicting intermediate states where only the label of a single nodes was extended.

Initial State

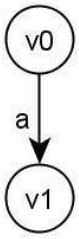
$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$
$\mathcal{N}$	$\emptyset$

Rule	Node	Triggering Concept
$R_{\sqcap}$	v_0	$D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$
$R_{\sqcup}$	v_0	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$

$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{N}$	$\emptyset$

Rule	Node	Triggering Concept
$R_{\exists}$	v_0	$\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}))$

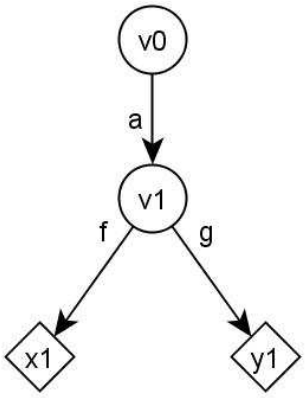
Added abstract node is v_1



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$\mathcal{N}$	$\emptyset$

Rule	Node	Triggering Concept
$R_{\sqcap}$	v_1	$\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$R_{\sqcup}$	v_1	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$R_{\ni}$	v_1	$\exists(f, g, R, \sigma_{\leq 0.5})$

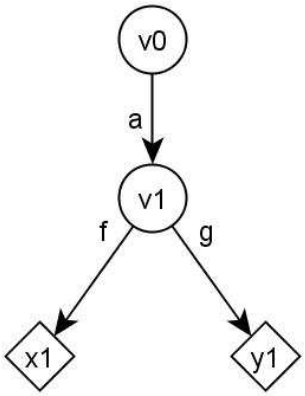
Added concrete nodes are x_1, y_1
Added constraint is $(x_1, y_1, R, \sigma_{\leq 0.5})$



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$, $\exists(f, g, R, \sigma_{\leq 0.5})$, D
$\mathcal{N}$	$(x_1, y_1, R, \sigma_{\leq 0.5})$, $(x_1, x_1, R, \underline{\mathbf{1}})$, $(y_1, y_1, R, \underline{\mathbf{1}})$, $(y_1, x_1, R, \underline{\mathbf{1}})$

Rule	Node	Triggering Concept
$R_{\forall}$	v_0	$\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$

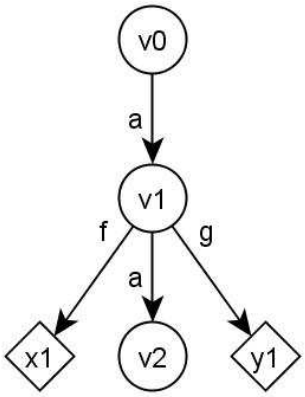
Added constraint is $(x_1, y_1, R, \sigma_{\geq 0.5})$



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$, $\exists(f, g, R, \sigma_{\leq 0.5})$, D
$\mathcal{N}$	$(x_1, y_1, R, \sigma_{=0.5})$, $(x_1, x_1, R, \underline{\mathbf{1}})$, $(y_1, y_1, R, \underline{\mathbf{1}})$, $(y_1, x_1, R, \underline{\mathbf{1}})$

Rule	Node	Triggering Concept
$R_{\exists}$	v_1	$\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$

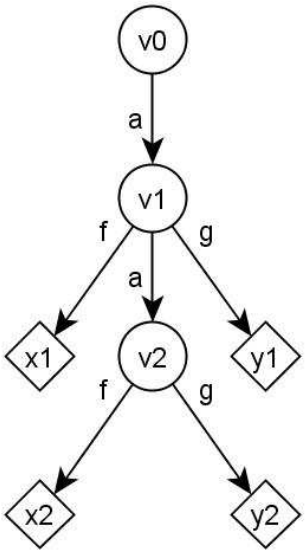
Added abstract node is v_2



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$, $\exists(f, g, R, \sigma_{\leq 0.5})$, D
$\mathcal{L}(v_2)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$\mathcal{N}$	$(x_1, y_1, R, \sigma_{=0.5})$, $(x_1, x_1, R, \underline{\mathbf{1}})$, $(y_1, y_1, R, \underline{\mathbf{1}})$, $(y_1, x_1, R, \underline{\mathbf{1}})$

Rule	Node	Triggering Concept
$R_{\sqcap}$	v_2	$\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$R_{\sqcup}$	v_2	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$R_{\ni}$	v_2	$\exists(f, g, R, \sigma_{\leq 0.5})$

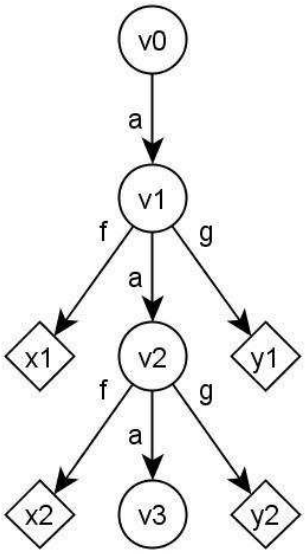
Added concrete nodes are x_2, y_2
Added constraint is $(x_2, y_2, R, \sigma_{\leq 0.5})$



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$, $\exists(f, g, R, \sigma_{\leq 0.5})$, D
$\mathcal{L}(v_2)$	$= \mathcal{L}(v_1)$
$\mathcal{N}$	$(x_1, y_1, R, \sigma_{=0.5})$, $(x_2, y_2, R, \sigma_{\leq 0.5})$, $(x_1, x_1, R, \underline{\mathbf{1}})$, $(y_1, y_1, R, \underline{\mathbf{1}})$, $(y_1, x_1, R, \underline{\mathbf{1}})$, $(x_1, x_2, R, \underline{\mathbf{1}})$, $(x_1, y_2, R, \underline{\mathbf{1}})$, $(y_1, x_2, R, \underline{\mathbf{1}})$, $(y_1, y_2, R, \underline{\mathbf{1}})$, $(x_2, x_1, R, \underline{\mathbf{1}})$, $(y_2, x_1, R, \underline{\mathbf{1}})$, $(x_2, y_1, R, \underline{\mathbf{1}})$, $(y_2, y_1, R, \underline{\mathbf{1}})$, $(x_2, x_2, R, \underline{\mathbf{1}})$, $(y_2, y_2, R, \underline{\mathbf{1}})$, $(y_2, x_2, R, \underline{\mathbf{1}})$

Rule	Node	Triggering Concept
$R_{\sqcap}$	v_2	$\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$R_{\sqcup}$	v_2	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$R_{\exists}$	v_2	$\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$

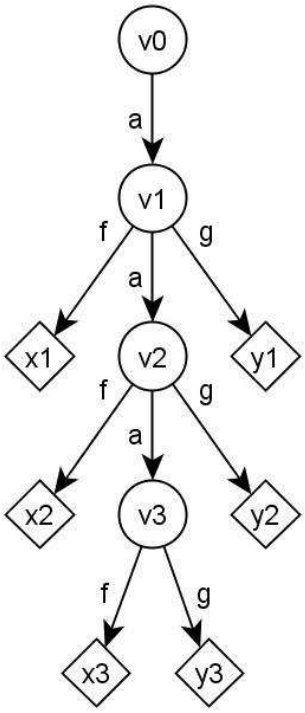
Added abstract node is v_3



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$, $\exists(f, g, R, \sigma_{\leq 0.5})$, D
$\mathcal{L}(v_2)$	$= \mathcal{L}(v_1)$
$\mathcal{L}(v_3)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$\mathcal{N}$	$(x_1, y_1, R, \sigma_{=0.5})$, $(x_2, y_2, R, \sigma_{\leq 0.5})$, $(x_1, x_1, R, \underline{\mathbf{1}})$, $(y_1, y_1, R, \underline{\mathbf{1}})$, $(y_1, x_1, R, \underline{\mathbf{1}})$, $(x_1, x_2, R, \underline{\mathbf{1}})$, $(x_1, y_2, R, \underline{\mathbf{1}})$, $(y_1, x_2, R, \underline{\mathbf{1}})$, $(y_1, y_2, R, \underline{\mathbf{1}})$, $(x_2, x_1, R, \underline{\mathbf{1}})$, $(y_2, x_1, R, \underline{\mathbf{1}})$, $(x_2, y_1, R, \underline{\mathbf{1}})$, $(y_2, y_1, R, \underline{\mathbf{1}})$, $(x_2, x_2, R, \underline{\mathbf{1}})$, $(y_2, y_2, R, \underline{\mathbf{1}})$, $(y_2, x_2, R, \underline{\mathbf{1}})$

Rule	Node	Triggering Concept
$R_{\sqcap}$	v_2	$\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$R_{\sqcup}$	v_2	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$R_{\sqcap}$	v_2	$\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$
$R_{\sqcup}$	v_2	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$R_{\ominus}$	v_2	$\exists(f, g, R, \sigma_{\leq 0.5})$

Added concrete nodes are x_3, y_3
Added constraint is $(x_3, y_3, R, \sigma_{\leq 0.5})$



$\mathcal{L}(v_0)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, D, $\forall(a\ f, a\ g, R, \sigma_{\geq 0.5})$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$
$\mathcal{L}(v_1)$	$\neg D \sqcup \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)$, $\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D$, $\exists(f, g, R, \sigma_{\leq 0.5})$, D
$\mathcal{L}(v_2)$	$= \mathcal{L}(v_1)$
$\mathcal{L}(v_3)$	$= \mathcal{L}(v_2)$
$\mathcal{N}$	$(x_1, y_1, R, \sigma_{=0.5})$, $(x_2, y_2, R, \sigma_{\leq 0.5})$, $(x_3, y_3, R, \sigma_{\leq 0.5})$, $(x_1, x_1, R, \underline{\mathbf{1}})$, $(y_1, y_1, R, \underline{\mathbf{1}})$, $(y_1, x_1, R, \underline{\mathbf{1}})$, $(x_1, x_2, R, \underline{\mathbf{1}})$, $(x_1, y_2, R, \underline{\mathbf{1}})$, $(y_1, x_2, R, \underline{\mathbf{1}})$, $(y_1, y_2, R, \underline{\mathbf{1}})$, $(x_2, x_1, R, \underline{\mathbf{1}})$, $(y_2, x_1, R, \underline{\mathbf{1}})$, $(x_2, y_1, R, \underline{\mathbf{1}})$, $(y_2, y_1, R, \underline{\mathbf{1}})$, $(x_2, x_2, R, \underline{\mathbf{1}})$, $(y_2, y_2, R, \underline{\mathbf{1}})$, $(y_2, x_2, R, \underline{\mathbf{1}})$, $(x_1, x_3, R, \underline{\mathbf{1}})$, $(x_1, y_3, R, \underline{\mathbf{1}})$, $(y_1, x_3, R, \underline{\mathbf{1}})$, $(y_1, y_3, R, \underline{\mathbf{1}})$, $(x_2, x_3, R, \underline{\mathbf{1}})$, $(x_2, y_3, R, \underline{\mathbf{1}})$, $(y_2, x_3, R, \underline{\mathbf{1}})$, $(y_2, y_3, R, \underline{\mathbf{1}})$, $(x_3, x_1, R, \underline{\mathbf{1}})$, $(y_3, x_1, R, \underline{\mathbf{1}})$, $(x_3, y_1, R, \underline{\mathbf{1}})$, $(y_3, y_1, R, \underline{\mathbf{1}})$, $(x_3, x_2, R, \underline{\mathbf{1}})$, $(y_3, x_2, R, \underline{\mathbf{1}})$, $(x_3, y_2, R, \underline{\mathbf{1}})$, $(y_3, y_2, R, \underline{\mathbf{1}})$, $(x_3, x_3, R, \underline{\mathbf{1}})$, $(y_3, y_3, R, \underline{\mathbf{1}})$

This is the first situation in which blocking occurs, due to $\mathcal{L}(v_2) = \mathcal{L}(v_3)$ and

$$\begin{aligned}
\mathcal{N}(v_2) = \big\{&(x_2, y_2, R, \sigma_{\leq 0.5}), (x_1, x_2, R, \underline{\mathbf{1}}), (x_1, y_2, R, \underline{\mathbf{1}}), (y_1, x_2, R, \underline{\mathbf{1}}),\\
&(y_1, y_2, R, \underline{\mathbf{1}}), (x_2, x_1, R, \underline{\mathbf{1}}), (y_2, x_1, R, \underline{\mathbf{1}}), (x_2, y_1, R, \underline{\mathbf{1}}),\\
&(y_2, y_1, R, \underline{\mathbf{1}}), (x_2, x_2, R, \underline{\mathbf{1}}), (y_2, y_2, R, \underline{\mathbf{1}}), (y_2, x_2, R, \underline{\mathbf{1}}),\\
&(x_2, x_3, R, \underline{\mathbf{1}}), (x_2, y_3, R, \underline{\mathbf{1}}), (y_2, x_3, R, \underline{\mathbf{1}}), (y_2, y_3, R, \underline{\mathbf{1}}),\\
&(x_3, x_2, R, \underline{\mathbf{1}}), (y_3, x_2, R, \underline{\mathbf{1}}), (x_3, y_2, R, \underline{\mathbf{1}}), (y_3, y_2, R, \underline{\mathbf{1}})\big\}\\
\mathcal{N}(v_3) = \big\{&(x_3, y_3, R, \sigma_{\leq 0.5}), (x_1, x_3, R, \underline{\mathbf{1}}), (x_1, y_3, R, \underline{\mathbf{1}}), (y_1, x_3, R, \underline{\mathbf{1}}),\\
&(y_1, y_3, R, \underline{\mathbf{1}}), (x_2, x_3, R, \underline{\mathbf{1}}), (x_2, y_3, R, \underline{\mathbf{1}}), (y_2, x_3, R, \underline{\mathbf{1}}),\\
&(y_2, y_3, R, \underline{\mathbf{1}}), (x_3, x_1, R, \underline{\mathbf{1}}), (y_3, x_1, R, \underline{\mathbf{1}}), (x_3, y_1, R, \underline{\mathbf{1}}),\\
&(y_3, y_1, R, \underline{\mathbf{1}}), (x_3, x_2, R, \underline{\mathbf{1}}), (y_3, x_2, R, \underline{\mathbf{1}}), (x_3, y_2, R, \underline{\mathbf{1}}),\\
&(y_3, y_2, R, \underline{\mathbf{1}}), (y_3, x_3, R, \underline{\mathbf{1}}), (x_3, x_3, R, \underline{\mathbf{1}}), (y_3, y_3, R, \underline{\mathbf{1}})\big\}
\end{aligned}$$

being isomorhpic. This is witnessed by the mappings in Table A.0.1 which yield the desired equality

$$\mu_{v_2}(\mathcal{N}(v_2)) = \mu_{v_3}(\mathcal{N}(v_3))$$

	μ_{v_2}	μ_{v_3}
x_1	a	a
x_2	b	c
x_3	c	b
y_1	d	d
y_2	e	f
y_3	f	e

Table A.0.1: Bijective mappings witnessing $\mathcal{N}(v_2) \sim \mathcal{N}(v_3)$

Since no further tableau rules are applicable, the algorithm stops and, given $\mathcal{N}$ is satisfiable, which we assume for now, returns '$satisfiable$'. Hence, a model for $\mathcal{C}$ and $\mathcal{T}$ can now be built.

At first let us construct the domain Δ_I. This consists of all $\mathfrak{T}$-chains from $\mathcal{S}$ (cf. page 52). These are those paths of abstract nodes in $\mathfrak{T}$, which start at the root, extended by those made by patching of blocked nodes to their blockers. This means besides the finite paths in $\mathfrak{T}$ which are those ending in v_0, v_1, v_2, we get an infinite set of chains 'cycling' through v_2, v_3. This is illustrated in Figure A.0.1.

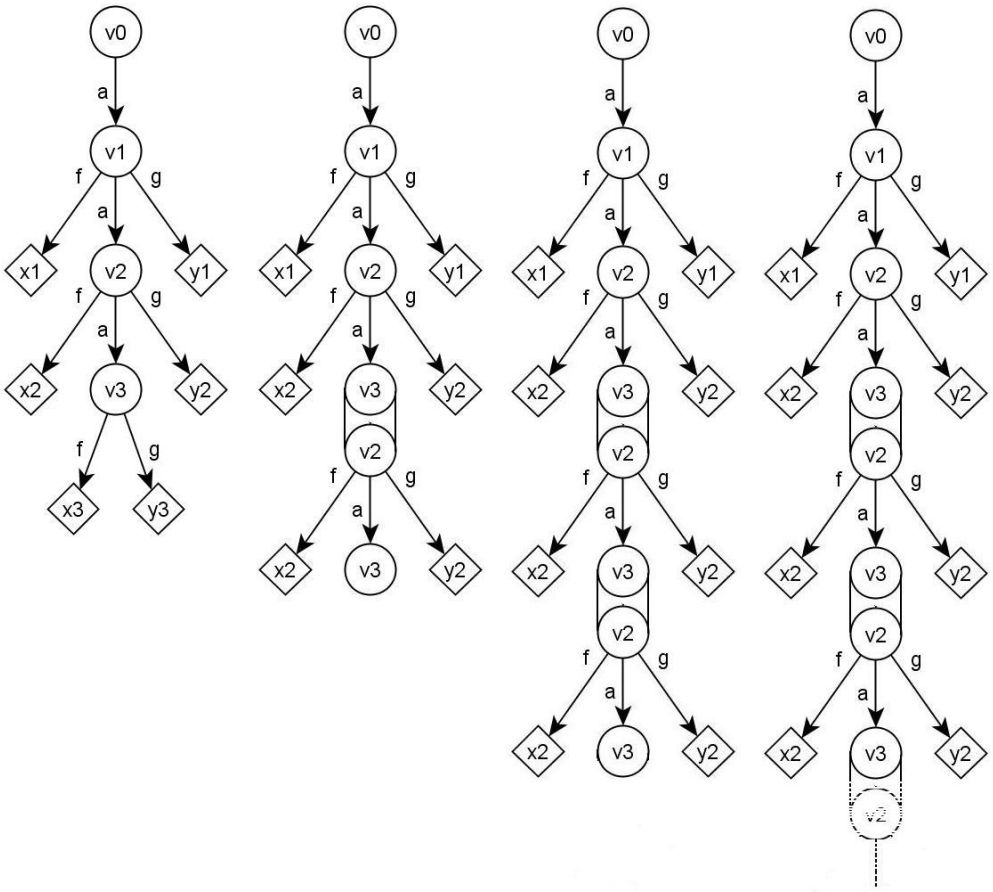


Figure A.0.1: $\mathfrak{T}$-chains in Δ_I

$\alpha :=$	$\frac{v_0}{v_0}$
$\beta :=$	$\frac{v_0\ v_1}{v_0\ v_1}$
$\gamma :=$	$\frac{v_0\ v_1\ v_2}{v_0\ v_1\ v_2}$
$\delta_0 :=$	$\frac{v_0\ v_1\ v_2\ v_2}{v_0\ v_1\ v_2\ v_3}$
$\delta_{i+1} :=$	$\delta_i \cdot \frac{v_2}{v_3}$, for $i \in \mathbb{N}$

Table A.0.2: Elements in Δ_I

χ	tail(χ)	$\pi(f, \chi)$	$\pi(g, \chi)$
β	$\frac{v_1}{v_1}$	x_β	y_β
γ	$\frac{v_2}{v_2}$	x_γ	y_γ
δ_i	$\frac{v_2}{v_3}$	$x_{\delta\, i}$	$y_{\delta\, i}$

Table A.0.3: Mappings to the concrete domain

Here you can see that the isomorphic nodes v_2 and v_3 are understood as one node. So, all outgoing edges from v_2 are made outgoing edges from v_3, which results in a new instance of v_3. This can then, again, be chosen as the starting point for the next iteration, repeat ad infinitum. For the handling of domain elements we use the naming scheme from Table A.0.2. All of these are $\mathfrak{T}$-chains and no other $\mathfrak{T}$-chains exists that start in the root node. Thus Δ_I contains exactly those elements. Further, the variables to be used in Net_I shall be named $x_\beta, x_\gamma, \ldots$ and $y_\beta, y_\gamma, \ldots$. To witness their presence in V_{Net_I} consider the mappings from Table A.0.3. Now, applying Definition 5.3.6 on page 55 we get:

$$\begin{aligned}
D^I &= \Delta_I \\
a^I &= \{\alpha \mapsto \beta, \beta \mapsto \gamma, \gamma \mapsto \delta_0\} \cup \{\delta_i \mapsto \delta_{i+1} \mid i \in \mathbb{N}\} \\
f^I(\chi) &= \pi(f, \chi) \\
g^I(\chi) &= \pi(g, \chi) \\
Net_I &= \{(x_\beta, y_\beta, R, \sigma_{=0.5}), (x_\gamma, y_\gamma, R, \sigma_{\leq 0.5})\} \\
&\quad \cup \{(x_{\delta\, i}, x_{\delta\, i}, R, \sigma_{\leq 0.5}) \mid i \in \mathbb{N}\}
\end{aligned}$$

For f^I and g^I refer to Table A.0.3. The `Rel`-network Net_I must, of course, be extended to its normal form. Then it nevertheless does indeed obey the requirement of tail(χ) = $\frac{v}{w}$ and $(\pi(f, w), \pi(g, w), R, \sigma) \in \mathcal{N}$ implying $(\pi(f, \chi), \pi(g, \chi), R, \sigma) \in Net_I$. That this interpretation is a model for the TBox

$$\mathcal{T} = \{D \sqsubseteq \exists a.(\exists(f, g, R, \sigma_{\leq 0.5}) \sqcap D)\}$$

is proved by:

$$
\begin{aligned}
D^I =& \Delta_I \\
E :=& \exists(f, g, R, \sigma_{\leq 0.5}) \\
E^I =& \{\delta \in \Delta_I \mid \exists x, y \in VNet_I, (x, y, R, \sigma') \in Net_I : f^I(\delta) = x, g^I(\delta) = y, \sigma' \subseteq \sigma_{\leq 0.5}\} \\
=& \Delta_I \\
F :=& (\exists a.(E \sqcap D)) \\
F^I =& (\exists a.(E \sqcap D))^I \\
=& (\exists a.(D \sqcap D))^I \\
=& (\exists a.(D))^I \\
=& D^I
\end{aligned}
$$

We further want to show that

$$\mathcal{C}^I = (D \sqcap \forall(a\ f, a\ g, R, \sigma_{\geq 0.5}))^I$$

is not empty. For this purpose we will show $\alpha \in \mathcal{C}^I$. It is clear that $\alpha \in D^I$. To be in $(\forall(a\ f, a\ g, R, \sigma_{\geq 0.5}))^I$ for all $v, w \in \Delta_I$, $x, y \in V_{Net_I}$ and $(x, y, R, \sigma') \in Net_I$ it holds that if $a^I(\alpha) = v, a^I(\alpha) = w, f^I(v) = x$ and $g^I(w) = y$ then $\sigma' \subseteq \sigma_{\leq 0.5}$. Since $a^I(\alpha) = \beta$, $f^I(\beta) = x_\beta$, $g^I(\beta) = y_\beta$, and $(x_\beta, y_\beta, R, \sigma_{=0.5}) \in Net_I$ we must have $\sigma_{=0.5} \subseteq \sigma_{\leq 0.5}$ i.e. $\{0.5\} \subseteq \{r \in \mathbb{R} \mid r \leq 0.5\}$ which is indeed true. We have shown that $\mathcal{C}$ is satisfiable with respect to $\mathcal{T}$.

Appendix B Defining Concrete Domains

Before being able to talk about *conrete domains* and *constraint systems* it must be clarified what these are. In [BH91] and [LM04] in opposition to [LM07] these notions are defined using quite different syntax for both. Thus we present a set of definitions first showing the original notions. Then a syntax and semantics is defined that can express both definitions in a common structure.

Definition B.1 (*Concrete Domain [BH91]*)

A (general) concrete domain *is a pair $\mathcal{D} = (\Delta, \Phi)$, where Δ is a set and Φ is a set of predicate names. Each predicate name is associated with an arity n and an n-ary predicate $P^{\mathcal{D}} \subseteq \Delta^n$. A $\mathcal{D}$-conjunction is a finite predicate conjunction of the form*

$$c = \bigwedge_{i<k} (x_0^{(i)}, \cdots, x_{n_i}^{(i)}) : P_i$$

where P_i is an n_i-ary predicate for $i < k$ and the $x_j^{(i)}$ are variables. A $\mathcal{D}$-conjunction c is satisfiable if there exists a function δ mapping the variables in c to elements of Δ such that $(\delta(x_0^{(i)}), \cdots, \delta(x_{n_i}^{(i)})) \in \mathcal{P}_i^{\mathcal{D}}$ for each $i < k$. Such a function is called a solution *for c.*

Definition B.2 (`Rel`*-Network, Complete Network, [LM07]*)

Let V be a countably infinite set of variables and `Rel` *a finite set of binary relation symbols. A* `Rel`-constraint *is an expression $c = (a, b, R)$ with $a, b \in V$, $R \in$* `Rel`*. A* `Rel`-network *is a (finite or infinite) set of* `Rel`*-constraints. For N a* `Rel`*-network we, use V_N to denote the variables used in N. We say that N is complete if for all pairs $(a, b) \in V_N \times V_N$ there is exactly one constraint $(a, b, R) \in N$.*

Definition B.3 (*Model, Constraint System, [LM07]*)

Let N be a `Rel`*-network and N' a complete* `Rel`*-network. We say that N' is a model of N if there is a mapping $\tau : V_N \to V_{N'}$ such that $(a, b, R) \in N$ implies $(\tau(a), \tau(b), R) \in N'$.*
A Constraint System *$C = (\texttt{Rel}, \mathfrak{M})$ consists of a finite set of binary relation symbols* `Rel` *and a set $\mathfrak{M}$ of complete* `Rel`*-networks (the* models *of C). A* `Rel`*-network N is satisfiable in C if $\mathfrak{M}$ contains a model of N.*

A reformulation of the above definitions for the sake of uniform syntax leads to:

Definition B.4 (*Rel-Constraints, Rel-Networks, Models*)

Let V be a countably infinite set of variables and `Rel` *a set of relation names, each associated with an arity n. For an n-ary relation with name $R \in$* `Rel` *an R-constraint is a $(n+1)$-tuple $(v_1, v_2, \cdots, v_n, R)$ with $v_i \in V$ for $i \in \mathbb{N}, 1 \leq i \leq n$. A* `Rel`*-constraint is an R-constraint for an arbitrary relation name $R \in$* `Rel`
A `Rel`*-network is a set of* `Rel`*-constraints. For a* `Rel`*-network N we use V_N to denote the variables used in N.*

Let N and M be `Rel`*-networks, with M being complete. We say M is a model for N if there exists a mapping $\delta : V_N \to V_M$ such that for every $R \in$* `Rel`*: $(v_1, v_2, \cdots, v_n, R) \in N$ implies $(\delta(v_1), \delta(v_2), \cdots, \delta(v_n), R) \in M$, where n is the arity of R.*

Definitions B.5 and B.6 are re-written versions of B.1 and B.3, respectively, obeying that new syntax.

Definition B.5 (*re-formulation of Def. B.1*)

A concrete domain $\mathcal{D} = (\texttt{Rel}, \mathfrak{M})$ *with* `Rel` *a (finite or infinite) set of relation symbols as defined above and $\mathfrak{M} = \{M\}$ a set containing one single* `Rel`*-network M then represents a set of* `Rel` *networks, which all have M as a model i.e. a* `Rel`*-network N is satisfiable in $\mathcal{D}$ if and only if M is a model of N. In a concrete domain all* `Rel`*-networks but the model have to be finite.*

Definition B.6 (*re-formulation of Def. B.3*)

In a constraint system $C = (\texttt{Rel}_2, \mathfrak{M})$ *it holds that* $\texttt{Rel}_2$ *is finite and all relation names are associated with arity 2. A model in a constraint system must be a complete* $\texttt{Rel}_2$*-network i.e. for very pair $(a, b) \in V \times V$ the network contains exactly one constraint (a, b, R). The set of models is not restricted to size 1 i.e. a* $\texttt{Rel}_2$*-network is satisfiable in C if and only if $\mathfrak{M}$ contains a model for N. In a constraint system* $\texttt{Rel}_2$*-networks may well be infinite.*

To show, that the definitions from [BH91, LM04, LM07] are adequately transcribed into the new syntax, it should suffice to say that the model M from Definition B.5 represents the set of predicates and that the `Rel`-networks in such concrete domains can be read as $\mathcal{D}$-conjunctions from Definition B.1.

The main differences between constraint systems and concrete domains as defined here lie in the size of the different sets used in their definition, see Table B.0.4. This makes obvious that neither all constraint systems are concrete domains nor vice versa. However, a constraint system would necessarily be a concrete domain, if the networks were restricted to be finite and there was a unique greatest model i.e. there exists a $M^* \in \mathfrak{M}$ such that M is a model for all $M \in \mathfrak{M}$.

	concrete domain	constraint system
`Rel`	probably infinite	finite
$\mathfrak{M}$	1	probably infinite
networks	finite	probably infinite
V	countably infinite	countably infinite
arities	$\in \mathbb{N}$	$= 2$

Table B.0.4: Sizes in Concrete Domains / Constraint Systems