A Tableau Algorithm for Description Logics with Concrete Domains and GCIs

Carsten Lutz and Maja Miličić [⋆]

Institute of Theoretical Computer Science
TU Dresden, Germany
{lutz,milicic}@tcs.inf.tu-dresden.de

Abstract. In description logics (DLs), *concrete domains* are used for defining concepts based on concrete qualities of their instances such as the weight, age, duration, and spatial extension. So-called *general concept inclusions (GCIs)* play an important role for capturing background knowledge. It is well-known that, when combining concrete domains with GCIs, reasoning easily becomes undecidable. In this paper, we identify a general property of concrete domains that is sufficient for proving decidability of DLs with both concrete domains and GCIs. We exhibit some useful concrete domains, most notably a spatial one based on the RCC-8 relations, which have this property. Then, we present a tableau algorithm for reasoning in DLs equipped with concrete domains and GCIs.

1 Introduction

Description Logics (DLs) are an important family of logic-based knowledge representation formalisms [4]. In DL, one of the main research goals is to provide a toolbox of logics such that, given an application, one may select a DL with adequate expressivity. Here, adequate means that, on the one hand, all relevant concepts from the application domain can be captured. On the other hand, no unessential means of expressivity should be included to prevent a (potential) increase in computational complexity. For several relevant applications of DLs such as the semantic web and reasoning about ER and UML diagrams, there is a need for DLs that include, among others, the expressive means *concrete domains* and *general concept inclusions (GCIs)* [3, 8, 15]. The purpose of concrete domains is to enable the definition of concepts with reference to concrete qualities of their instances such as the weight, age, duration, and spatial extension. GCIs play an important role in modern DLs as they allow to represent background knowledge of application domains by stating that the extension of a concept is included in the extension of another concept.

Unfortunately, combining concrete domains with GCIs easily leads to undecidabilty. For example, it has been shown in [18] that the basic DL $\mathcal{ALC}$ extended with GCIs and a rather inexpressive concrete domain based on the natural numbers and providing for equality and incrementation predicates is undecidable,

[⋆] supported by DFG under grant GRK 334/3

see also the survey paper [16]. In view of this discouraging result, it is a natural question whether there are *any* useful concrete domains that can be combined with GCIs in a decidable DL. A positive answer to this question has been given in [17] and [14], where two such well-behaved concrete domains are identified: a temporal one based on the Allen relations for interval-based temporal reasoning, and a numerical one based on the rationals and equipped with various unary and binary predicates such as "$\leq$", "$>_5$", and "$\neq$". Using an automata-based approach, it has been shown in [17, 14] that reasoning in the DLs $\mathcal{ALC}$ and $\mathcal{SHIQ}$ extended with these concrete domains and GCIs is decidable and ExpTime-complete.

The purpose of this paper it to advance the knowledge about decidable DLs with both concrete domains and GCIs. Our contribution is two-fold: first, instead of focussing on particular concrete domains as in previous work, we identify a *general* property of concrete domains, called ω-admissibility, that is sufficient for proving decidability of DLs equipped with concrete domains and GCIs. For defining ω-admissibility, we concentrate on a particular kind of concrete domains: *constraint systems*. Roughly, a constraint system is a concrete domain that only has binary predicates, which are interpreted as jointly exhaustive and pairwise disjoint (JEPD) relations. We exhibit two example constraint systems that are ω-admissible: a temporal one based on the rational line and the Allen relations [1], and a spatial one based on the real plane and the RCC8 relations [6, 20]. The proof of ω-admissibility turns out to be relatively straightforward in the Allen case, but is somewhat cumbersome for RCC8. We believe that there are many other useful constraint systems that can be proved ω-admissible.

Second, for the first time we develop a *tableau algorithm* for DLs with both concrete domains and GCIs. This algorithm is used to establish a general decidability result for $\mathcal{ALC}$ equipped with GCIs and any ω-admissible concrete domain. In particular, we obtain decidability of $\mathcal{ALC}$ with GCIs and the Allen relations as first established in [17], and, as a new result, get decidability of $\mathcal{ALC}$ with GCIs and the RCC8 relations as a concrete domain. As state-of-the-art DL reasoners such as FaCT and RACER are based on tableau algorithms similar to the one described in this paper [11, 10], we view our algorithm as a first step towards an efficient implementation of description logics with (ω-admissible) concrete domains and GCIs.

This paper is organized as follows: in Section 2, we introduce constraint systems and identify some properties of constraint systems that will be useful for defining ω-admissibility. In Section 3, we introduce the description logic $\mathcal{ALC}(\mathcal{C})$ that incorporates constraint systems and GCIs. The tableau algorithm for deciding satisfiability in $\mathcal{ALC}(\mathcal{C})$ is developed in Section 4. In Section 5, we briefly discuss the implementability of our algorithm. This paper is accompanied by a technical report containing full proofs [19].

2 Constraint Systems

We introduce a general notion of *constraint system* that is intended to capture standard constraint systems based on a set of jointly-exhaustive and pairwise-disjoint (JEPD) binary relations.

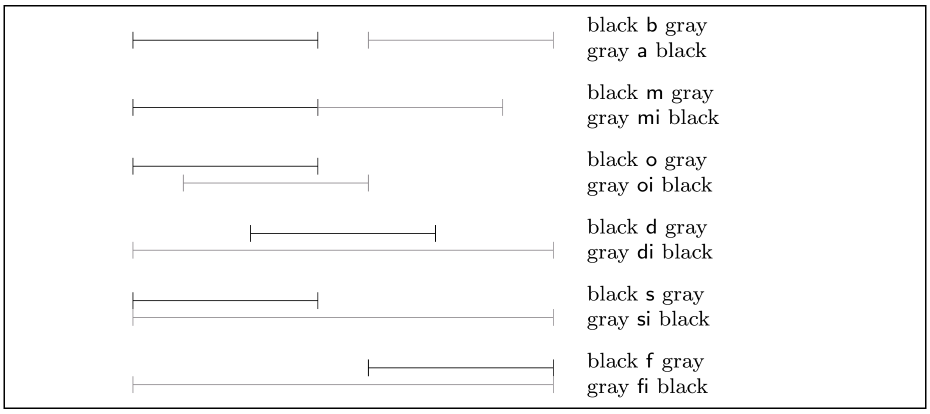

Fig. 1. The thirteen Allen relations, equality omitted.

Let Var be a countably infinite set of variables and Rel a finite set of binary relation symbols. A Rel-*constraint* is an expression $(v\ r\ v')$ with $v, v' \in \mathsf{Var}$ and $r \in \mathsf{Rel}$. A Rel-*network* is a (finite or infinite) set of Rel-constraints. Let N be a Rel-network. We use V_N to denote the variables used in N and say that N is *complete* if, for all $v, v' \in V_N$, there is exactly one constraint $(v\ r\ v') \in N$.

To assign a semantics to networks in an abstract way, we use complete networks as models: N is a *model of a network* N' if N is complete and there is a mapping $\tau : V_{N'} \rightarrow V_N$ such that $(v\ r\ v') \in N'$ implies $(\tau(v)\ r\ \tau(v')) \in N$. In this context, the nodes in N, although from the set Var, are to be understood as values rather than variables (see below for examples).

A *constraint system* $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ consists of a finite set of binary relation symbols Rel and a set $\mathfrak{M}$ of complete Rel-networks (the *models* of $\mathcal{C}$). A Rel-network N is *satisfiable* in $\mathcal{C}$ if $\mathfrak{M}$ contains a model of N.

We give two examples of constraint systems: a constraint system for temporal reasoning based on the Allen relations in the rational line, and a constraint system for spatial reasoning based on the RCC8 relations in the real plane. Both constraint systems have been extensively studied in the literature.

In artificial intelligence, constraint systems based on Allen's interval relations are a popular tool for the representation of temporal knowledge [1]. Let

$$\mathsf{Allen} = \{\mathsf{b}, \mathsf{a}, \mathsf{m}, \mathsf{mi}, \mathsf{o}, \mathsf{oi}, \mathsf{d}, \mathsf{di}, \mathsf{s}, \mathsf{si}, \mathsf{f}, \mathsf{fi}, =\}$$

denote the thirteen Allen relations. Examples of these relations are given in Figure 1. As the flow of time, we use the rational numbers with the usual ordering. Let $\mathsf{Int}_{\mathbb{Q}}$ denote the set of all closed intervals $[q_1, q_2]$ over $\mathbb{Q}$ with $q_1 < q_2$, i.e., point-intervals are not admitted. The extension $\mathsf{r}^{\mathbb{Q}}$ of each Allen relation r is a subset of $\mathsf{Int}_{\mathbb{Q}} \times \mathsf{Int}_{\mathbb{Q}}$. It is defined in terms of the relationships between endpoints in the obvious way, c.f. Figure 1. We define the constraint system $\mathsf{Allen}_{\mathbb{Q}} = \langle \mathsf{Allen}, \mathfrak{M}_{\mathbb{Q}} \rangle$ by setting $\mathfrak{M}_{\mathbb{Q}} := \{N_{\mathbb{Q}}\}$, where $N_{\mathbb{Q}}$ is defined by fixing a

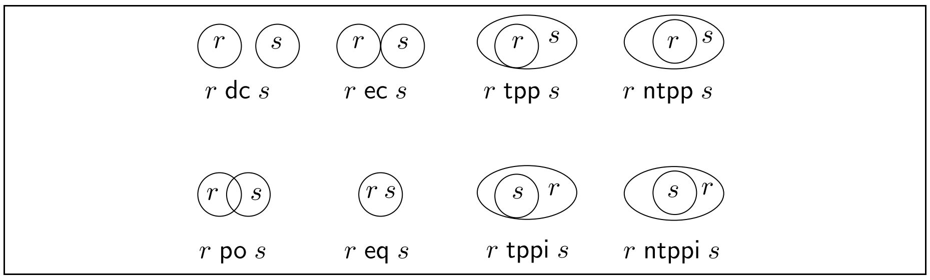

Fig. 2. The eight RCC8 relations.

variable $v_i \in \mathsf{Var}$ for every $i \in \mathsf{Int}_{\mathbb{Q}}$ and setting

$$N_{\mathbb{Q}} := \{(v_i \; r \; v_j) \mid r \in \mathsf{Allen},\; i,j \in \mathsf{Int}_{\mathbb{Q}} \text{ and } (i,j) \in r^{\mathbb{Q}}\}.$$

Whether we use the rationals or the reals for defining this constraint system has no impact on the satisfiability of (finite and infinite) constraint networks.

The RCC8 relations describe the possible relation between two regions in a topological space [20]. In this paper, we use the standard topology of the real plane, one of the most natural topologies for spatial reasoning. Let

$$\mathsf{RCC8} = \{\mathsf{eq}, \mathsf{dc}, \mathsf{ec}, \mathsf{po}, \mathsf{tpp}, \mathsf{ntpp}, \mathsf{tppi}, \mathsf{ntppi}\}$$

denote the RCC8 relations. Examples of these relations are given in Figure 2. Recall that a topological space is a pair $\mathfrak{T} = (U, \mathbb{I})$, where U is a set and $\mathbb{I}$ is an *interior operator* on U, i.e., for all $s, t \subseteq U$, we have

$$\mathbb{I}(U) = U \qquad \mathbb{I}(s) \subseteq s \qquad \mathbb{I}(s) \cap \mathbb{I}(t) = \mathbb{I}(s \cap t) \qquad \mathbb{I}\mathbb{I}(s) = \mathbb{I}(s).$$

As usual, the closure operator $\mathbb{C}$ is defined as $\mathbb{C}(s) = \overline{\mathbb{I}(\overline{s})}$, where $\overline{t} = U \setminus t$, for $t \subseteq U$. As the *regions* of a topological space $\mathfrak{T} = (U, \mathbb{I})$, we use the set of non-empty, regular closed subsets of U, where a subset $s \subseteq U$ is called *regular closed* if $\mathbb{C}\mathbb{I}(s) = s$. Given a topological space $\mathfrak{T}$ and a set of regions $U_{\mathfrak{T}}$, we define the extension of the RCC8 relations as the following subsets of $U_{\mathfrak{T}} \times U_{\mathfrak{T}}$:

$$\begin{array}{rl}
(s,t) \in \mathsf{dc}^{\mathfrak{T}} & \text{iff } \; s \cap t = \emptyset \\
(s,t) \in \mathsf{ec}^{\mathfrak{T}} & \text{iff } \; \mathbb{I}(s) \cap \mathbb{I}(t) = \emptyset \;\wedge\; s \cap t \neq \emptyset \\
(s,t) \in \mathsf{po}^{\mathfrak{T}} & \text{iff } \; \mathbb{I}(s) \cap \mathbb{I}(t) \neq \emptyset \;\wedge\; s \setminus t \neq \emptyset \;\wedge t \setminus s \neq \emptyset \\
(s,t) \in \mathsf{eq}^{\mathfrak{T}} & \text{iff } \; s = t \\
(s,t) \in \mathsf{tpp}^{\mathfrak{T}} & \text{iff } \; s \cap \overline{t} = \emptyset \;\wedge\; s \cap \overline{\mathbb{I}(t)} \neq \emptyset \\
(s,t) \in \mathsf{ntpp}^{\mathfrak{T}} & \text{iff } \; s \cap \overline{\mathbb{I}(t)} = \emptyset \\
(s,t) \in \mathsf{tppi}^{\mathfrak{T}} & \text{iff } \; (t,s) \in \mathsf{tpp}^{\mathfrak{T}} \\
(s,t) \in \mathsf{ntppi}^{\mathfrak{T}} & \text{iff } \; (t,s) \in \mathsf{ntpp}^{\mathfrak{T}}.
\end{array}$$

Let $\mathfrak{T}_{\mathbb{R}^2}$ be the standard topology on $\mathbb{R}^2$ induced by the Euclidean metric, and let $\mathcal{RS}_{\mathbb{R}^2}$ be the set of all non-empty regular-closed subsets of $\mathfrak{T}_{\mathbb{R}^2}$. Intuitively, regular closedness is required to eliminate sub-dimensional regions such as 0-dimensional points and 1-dimensional spikes. We define the constraint system $\mathsf{RCC8}_{\mathbb{R}^2} = \langle \mathsf{RCC8}, \mathfrak{M}_{\mathbb{R}^2} \rangle$ by setting $\mathfrak{M}_{\mathbb{R}^2} := \{N_{\mathbb{R}^2}\}$, where $N_{\mathbb{R}^2}$ is defined by fixing a variable $v_s \in \mathsf{Var}$ for every $s \in \mathcal{RS}_{\mathbb{R}^2}$ and setting

$$N_{\mathbb{R}^2} := \{(v_s \; r \; v_t) \mid r \in \mathsf{RCC8}, \; s,t \in \mathcal{RS}_{\mathbb{R}^2} \text{ and } (s,t) \in r^{\mathfrak{T}_{\mathbb{R}^2}}\}.$$

Properties of Constraint Systems

We will use constraint systems as a concrete domain for description logics. To obtain sound and complete reasoning procedures for DLs with such concrete domains, we require constraint system to have certain properties.

Definition 1 (Patchwork Property, Compactness). *Let $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ be a constraint system. If N is a* Rel*-network and $V \subseteq V_N$, we write $N|_V$ to denote the network $\{(v \, r \, v') \in N \mid v, v' \in V\} \subseteq N$. We say that*

- *$\mathcal{C}$ has the* patchwork property *if the following holds: for all finite, complete, and satisfiable* Rel*-networks N, M that agree on their (possibly empty) intersection (i.e. $N|_{V_N \cap V_M} = M|_{V_N \cap V_M}$), $N \cup M$ is satisfiable;*
- *$\mathcal{C}$ has the* compactness property *if the following holds: a* Rel*-network N with V_N infinite is satisfiable in $\mathcal{C}$ if and only if, for every finite $V \subseteq V_N$, the network $N|_V$ is satisfiable in $\mathcal{C}$.*

Intuitively, the patchwork property ensures that satisfiable networks (satisfying some additional conditions) can be "patched" together to a joint network that is also satisfiable. Compactness ensures that this even works when patching together an infinite number of satisfiable networks. In [5], where constraint systems are combined with linear temporal logic, Balbiani and Condotta formulate a property closely related to ours. This property requires that partial models of networks can be extended to complete models. For our purposes, such a property could be used alternatively to the patchwork property and compactness (in fact, it implies both of them).

In the technical report [19], we prove the following:

Theorem 1. $\mathsf{RCC8}_{\mathbb{R}^2}$ *and* $\mathsf{Allen}_{\mathbb{Q}}$ *satisfy the patchwork property and the compactness property.*

The proof of compactness works by devising a satisfiability-preserving translation of constraint networks to sets of first-order formulas, and then appealing to compactness of the latter. In the case of $\mathsf{Allen}_{\mathbb{Q}}$, we need compactness of first-order logic on structures $\langle \mathbb{Q}, < \rangle$, while arbitrary structures are sufficient for $\mathsf{RCC8}_{\mathbb{R}^2}$. The proof of the patchwork property is relatively straightforward in the case of $\mathsf{Allen}_{\mathbb{Q}}$: given two finite, satisfiable, and complete networks N and M that agree on the overlapping part, we show how models of N and M can be manipulated into a model of $N \cup M$. Finally, the proof of the patchwork property of $\mathsf{RCC8}_{\mathbb{R}^2}$ requires quite some machinery. We consider RCC8-networks interpreted on topologies that are induced by so-called fork frames, and then use the standard translation of RCC8-networks into the model logic S4 and repeated careful applications of a theorem from [9] to establish the patchwork property.

3 Syntax and Semantics

We introduce the description logic $\mathcal{ALC}(\mathcal{C})$ that allows to define concepts with reference to the constraint system $\mathcal{C}$. Different incarnations of $\mathcal{ALC}(\mathcal{C})$ are obtained by instantiating it with different constraint systems.

Let $\mathcal{C} = (\mathsf{Rel}, \mathfrak{M})$ be a constraint system, and let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_{cF}}$ be mutually disjoint and countably infinite sets of *concept names*, *role names*, and *concrete features*. We assume that $\mathsf{N_R}$ has a countably infinite subset $\mathsf{N_{aF}}$ of *abstract features*. A *path* is a sequence $R_1 \cdots R_k g$ consisting of roles $R_1, \ldots, R_k \in \mathsf{N_R}$ and a concrete feature $g \in \mathsf{N_{cF}}$. A path $R_1 \cdots R_k g$ with $\{R_1, \ldots, R_k\} \subseteq \mathsf{N_{aF}}$ is called *feature path*. The set of $\mathcal{ALC}(\mathcal{C})$-concepts is built according to the following syntax rule

$$C ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \exists U_1, U_2.r \mid \forall U_1, U_2.r$$

where A ranges over $\mathsf{N_C}$, R ranges over $\mathsf{N_R}$, r ranges over Rel, and U_1, U_2 are both feature paths or $U_1 = Rg_1$ and $U_2 = g_2$ with $R \in \mathsf{N_R}$ and $g_1, g_2 \in \mathsf{N_{cF}}$ or vice versa. Throughout this paper, we use $\top$ as abbreviation for an arbitrary propositional tautology and $C \rightarrow D$ for $\neg C \sqcup D$.

A *general concept inclusion axiom (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are concepts. A finite set of GCIs is called *TBox*. The TBox formalism introduced here is often called *general TBox* since it subsumes several other, much weaker variants [7, 13]. We use $C \doteq D$ to abbreviate $C \sqsubseteq D$ and $D \sqsubseteq C$. For example, the following TBox describes some properties of cities using the concrete domain $\mathsf{RCC}_{\mathbb{R}^2}$:

$$\begin{aligned}
\mathsf{City} &\sqsubseteq \forall \mathsf{waters}.(\mathsf{River} \sqcup \mathsf{Lake} \sqcup \mathsf{Ocean}) \sqcap \forall \mathsf{trade\text{-}partner}.\mathsf{City} \\
\mathsf{RegionalTrader} &\doteq \mathsf{City} \sqcap \exists (\mathsf{trade\text{-}partner\ loc}), (\mathsf{province\ loc}).\mathsf{ntpp} \\
\mathsf{HarborCity} &\doteq \exists (\mathsf{waters\ loc}), \mathsf{loc}.\mathsf{po} \ \sqcap \ \exists (\mathsf{port\ loc}), \mathsf{loc}.\mathsf{ntpp} \\
&\quad \sqcap \exists (\mathsf{waters\ loc}), (\mathsf{port\ loc}).\mathsf{ec}
\end{aligned}$$

Here, trade−partner is a role, province, waters, and port are abstract features, and loc is a concrete feature. The second GCI says that RegionalTraders trade with at least one city located in the same province. The third GCI says that HarborCitys overlap some water and contain a port externally connected to this water.

The semantics of $\mathcal{ALC}(\mathcal{C})$ is defined in terms of interpretations as usual. To deal with the *constraint constructors* $\exists U_1, U_2.r$ and $\forall U_1, U_2.r$, interpretations comprise a model of $\mathcal{C}$ as an additional component: an *interpretation* $\mathcal{I}$ is a tuple $(\Delta_\mathcal{I}, \cdot^\mathcal{I}, M_\mathcal{I})$, where $\Delta_\mathcal{I}$ is a set called the *domain*, $\cdot^\mathcal{I}$ is the *interpretation function*, and $M_\mathcal{I} \in \mathfrak{M}$. The interpretation function maps

- each concept name C to a subset $C^\mathcal{I}$ of $\Delta_\mathcal{I}$,
- each role name R to a subset $R^\mathcal{I}$ of $\Delta_\mathcal{I} \times \Delta_\mathcal{I}$,
- each abstract feature f to a partial function $f^\mathcal{I}$ from $\Delta_\mathcal{I}$ to $\Delta_\mathcal{I}$, and
- each concrete feature g to a partial function $g^\mathcal{I}$ from $\Delta_\mathcal{I}$ to the set of variables $V_{M_\mathcal{I}}$ of $M_\mathcal{I}$.

The interpretation function is extended to arbitrary concepts as follows:

$$\begin{aligned}
\neg C^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
(C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\
(\exists R.C)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d,e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}, \\
(\forall R.C)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}} : (d,e) \in R^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}}\}, \\
(\exists U_1, U_2.r)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \exists x_1 \in U_1^{\mathcal{I}}(d),\ x_2 \in U_2^{\mathcal{I}}(d) : (x_1 r x_2) \in M_{\mathcal{I}}\} \\
(\forall U_1, U_2.r)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \forall x_1 \in U_1^{\mathcal{I}}(d),\ x_2 \in U_2^{\mathcal{I}}(d) : (x_1 r x_2) \in M_{\mathcal{I}}\}
\end{aligned}$$

where, for every path $U = R_1 \cdots R_k g$ and $d \in \Delta_{\mathcal{I}}$, $U^{\mathcal{I}}(d)$ is defined as

$$\begin{aligned}
\{x \in V_{M_{\mathcal{I}}} \mid \exists e_1, \ldots, e_{k+1} : d = e_1,& \\
(e_i, e_{i+1}) \in R_i^{\mathcal{I}} \text{ for } 1 \leq i \leq k,& \text{ and } g^{\mathcal{I}}(e_{k+1}) = x\}.
\end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* of a concept C iff $C^{\mathcal{I}} \neq \emptyset$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all GCIs $C \sqsubseteq D$ in $\mathcal{T}$.

The most important reasoning tasks for DLs are satisfiability and subsumption: a concept C is called *satisfiable with respect to a TBox $\mathcal{T}$* iff there exists a common model of C and $\mathcal{T}$. A concept D *subsumes* a concept C with respect to $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{T}$. It is well-known that subsumption can be reduced to (un)satisfiability: $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$. Therefore, in the current paper we only consider concept satisfiability.

4 Tableau Algorithm

In this section, we present a tableau algorithm which decides satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts w.r.t. TBoxes. Tableau algorithms are the most popular decision procedures for description logics since, despite not always yielding tight upper complexity bounds, they are amenable to various optimizations and can often be efficiently implemented. In general, tableau algorithms for DLs decide satisfiability of a concept by trying to construct a model for it. The underlying data structure is a tree which, in case of a successful run of the algorithm, represents a *single* tree model of the input concept and TBox in a straightforward way: the nodes of the tree are the domain elements and the edges denote the extension of roles. Note that this is in contrast to many modal and first-order tableaux, where models of the input formula correspond to *branches* of the tree generated by the tableau algorithm.

Before presenting the tableau algorithm for $\mathcal{ALC}(\mathcal{C})$, we need some prerequisites. In particular, we assume a certain normal form for concepts and TBoxes: negation is only allowed in front of concept names, and the length of paths is restricted.

A concept is said to be in *negation normal form (NNF)* if negation occurs only in front of concept names. We now show that NNF can be assumed without loss of generality: for every $\mathcal{ALC}(\mathcal{C})$-concept, an eqi-satisfiable one in NNF can be computed in linear time. Note that usual NNF transformations are even equivalence-preserving, which cannot be achieved in our case. We assume that the constraint system $\mathcal{C}$ has an equality predicate "=", i.e., $= \in \mathsf{Rel}$ such that, for all $M \in \mathfrak{M}$ and $v \in V_M$, we have $(v = v) \in M$.

Lemma 1 (NNF Conversion). *Exhaustive application of the following rewrite rules translates $\mathcal{ALC}(\mathcal{C})$-concepts to eqi-satisfiable ones in NNF. The number of rule applications is linear in the length of the original concept.*

$$\neg\neg C \rightsquigarrow C \qquad \neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D \qquad \neg(C \sqcup D) \rightsquigarrow \neg C \sqcap \neg D$$

$$\neg(\exists R.C) \rightsquigarrow (\forall R.\neg C) \qquad \neg(\forall R.C) \rightsquigarrow (\exists R.\neg C)$$

$$\neg(\forall U_1, U_2.r) \rightsquigarrow \bigsqcup_{r' \in \mathsf{Rel}, r' \neq r} \exists U_1, U_2.r'$$

$$\neg(\exists U_1, U_2.r) \rightsquigarrow \bigsqcup_{r' \in \mathsf{Rel}, r' \neq r} \forall U_1, U_2.r' \qquad \textit{where } U_1, U_2 \textit{ are feature paths}$$

$$\neg(\exists R g_1, g_2.r) \rightsquigarrow (\forall R g^*, g_2.=) \sqcap \bigsqcup_{r' \in \mathsf{Rel}, r' \neq r} \forall R.(\forall g_1, g^*.r')$$

where $R \in \mathsf{N_R} \setminus \mathsf{N_{aF}}$ and g^* is a fresh concrete feature

By $\mathsf{nnf}(C)$*, we denote the result of converting C into NNF using the above rules.*

In the last transformation, the fresh concrete feature g^* is used to propagate the value of g_2 to all R successors. This transformation is the reason for the fact that our NNF translation is not equivalence-preserving. Intuitively, giving an equivalence preserving-translation would require to allow the formation of complex $\mathcal{C}$-relations from atomic ones by means of union.

We now introduce path normal form for $\mathcal{ALC}(\mathcal{C})$-concepts and TBoxes. Path normal form was first considered in [17, 14].

Definition 2 (Path Normal Form). *An $\mathcal{ALC}(\mathcal{C})$-concept C is in* path normal form (PNF) *iff it is in NNF and, for all subconcepts $\exists U_1, U_2.r$ and $\forall U_1, U_2.r$ of C, we have either*

1. *$U_1 = g_1$ and $U_2 = g_2$ for some $g_1, g_2 \in \mathsf{N_{cF}}$ or*
2. *$U_1 = R g_1$ and $U_2 = g_2$ for some $R \in \mathsf{N_R}$ and $g_1, g_2 \in \mathsf{N_{cF}}$ or*
3. *$U_1 = g_1$ and $U_1 = R g_2$ for some $R \in \mathsf{N_R}$ and $g_1, g_2 \in \mathsf{N_{cF}}$.*

An $\mathcal{ALC}(\mathcal{C})$-TBox $\mathcal{T}$ is in path normal form iff all concepts in $\mathcal{T}$ are in PNF.

The following lemma shows that we can w.l.o.g. assume $\mathcal{ALC}(\mathcal{C})$-concepts and TBoxes to be in PNF.

Lemma 2. *Satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts w.r.t. TBoxes can be polynomially reduced to satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts in PNF w.r.t. TBoxes in PNF.*

Proof. Let C be an $\mathcal{ALC}(\mathcal{C})$-concept. For every feature path $u = f_1 \cdots f_n g$ used in C, we assume that $[g], [f_n g], \ldots, [f_1 \cdots f_n g]$ are concrete features not used in C. We inductively define a mapping λ from feature paths u in C to concepts as follows:

$$\lambda(g) = \top \qquad \lambda(fu) = (\exists f[u], [fu].=) \sqcap \exists f.\lambda(u)$$

For every $\mathcal{ALC}(\mathcal{C})$-concept C, a corresponding concept $\rho(C)$ is obtained by

– first replacing all subconcepts $\forall u_1, u_2.r$, where $u_i = f_1^{(i)} \cdots f_{k_i}^{(i)} g_i$ for $i \in \{1,2\}$, with

$$\forall f_1^{(1)}. \cdots \forall f_{k_1}^{(1)}.\forall g_1, g_1.\mathsf{r}^{\neq} \sqcup \forall f_1^{(2)}. \cdots \forall f_{k_2}^{(2)}.\forall g_2, g_2.\mathsf{r}^{\neq} \sqcup \exists u_1, u_2.r$$

where $\mathsf{r}^{\neq} \in \mathsf{Rel} \setminus \{=\}$ is arbitrary, but fixed;

– and then replacing all subconcepts $\exists u_1, u_2.r$ with $\exists [u_1], [u_2].r \sqcap \lambda(u_1) \sqcap \lambda(u_2)$.

We extend the mapping ρ to TBoxes in the obvious way: replace each GCI $C \sqsubseteq D$ with $\rho(C) \sqsubseteq \rho(D)$. To convert a concept to PNF, we may first convert to NNF and then apply the above translation ρ. It is easily verified that (un)satisfiability is preserved, and that the translation can be done in polynomial time. $\square$

In what follows, we generally assume that all concepts and TBoxes are in path normal form. Moreover, we require that constraint systems are ω-admissible:

Definition 3 (ω-admissible). *Let $\mathcal{C} = (\mathsf{Rel}, \mathfrak{M})$ be a constraint system. We say that $\mathcal{C}$ is* ω-admissible *iff the following holds:*

1. *satisfiability in $\mathcal{C}$ is decidable;*
2. *$\mathcal{C}$ has the patchwork property;*
3. *$\mathcal{C}$ has the compactness property.*

In Section 2, we have shown that $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{Q}}$ have the patchwork and compactness property. Moreover, satisfiability in $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{Q}}$ is NP-complete [21, 22]. Thus, these constraint systems are ω-admissible and may be used with our tableau algorithm.

Let C_0 be a concept and $\mathcal{T}$ a TBox such that satisfiability of C_0 w.r.t. $\mathcal{T}$ is to be decided. The *concept form $C_\mathcal{T}$* is defined as

$$C_\mathcal{T} = \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \mathsf{nnf}(C \rightarrow D).$$

We define the set of subconcepts $\mathsf{sub}(C_0, \mathcal{T}) := \mathsf{sub}(C_0) \cup \mathsf{sub}(C_\mathcal{T})$, with $\mathsf{sub}(C)$ denoting the set of all subconcepts of C, including C.

As already noted, our algorithm uses trees as the main data structure, and nodes of this tree represent elements of the interpretation domain. Due to the presence of concrete domains, trees have two types of nodes: abstract ones that represent individuals of the logic domain $\Delta_\mathcal{I}$, and concrete ones representing values of the concrete domain. Likewise, edges represent either roles or concrete features.

Definition 4 (Completion system). *Let $\mathsf{O_a}$ and $\mathsf{O_c}$ be disjoint and countably infinite sets of* abstract *and* concrete nodes. *A* completion tree *for C_0, $\mathcal{T}$ is a finite, labelled tree $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ with nodes $\mathsf{V_a} \cup \mathsf{V_c}$, such that $\mathsf{V_a} \subseteq \mathsf{O_a}$, $\mathsf{V_c} \subseteq \mathsf{O_c}$, and all nodes from $\mathsf{V_c}$ are leaves. The tree is labelled as follows:*

1. *each node $a \in \mathsf{V_a}$ is labelled with a subset $\mathcal{L}(a)$ of $\mathsf{sub}(C_0, \mathcal{T})$,*
2. *each edge $(a, b) \in E$ with $a, b \in \mathsf{V_a}$ is labelled with a role name $\mathcal{L}(a, b)$ occurring in C_0 or $\mathcal{T}$;*

3. *each edge $(a,x) \in E$ with $a \in V_a$ and $x \in \mathsf{V_c}$ is labelled with a concrete feature $\mathcal{L}(a,x)$ occurring in C_0 or $\mathcal{T}$.*

A node $b \in \mathsf{V_a}$ is an R-successor of a node $a \in \mathsf{V_a}$ if $(a,b) \in E$ and $\mathcal{L}(a,b) = R$, while an $x \in \mathsf{V_c}$ is a g-successor of a if $(a,x) \in E$ and $\mathcal{L}(a,x) = g$. The notion u-successor for a path u is defined in the obvious way. A completion system *for C_0 and $\mathcal{T}$ is a tuple $S = (T, \mathcal{N})$ where $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ is a completion tree for C_0 and $\mathcal{T}$ and $\mathcal{N}$ is a* Rel*-network with $V_\mathcal{N} = \mathsf{V_c}$.*

To decide the satisfiability of C_0 w.r.t. $\mathcal{T}$ (both in PNF), the tableau algorithm is started with the initial completion system

$$S_{C_0} = (T_{C_0}, \emptyset), \text{ where } T_{C_0} = (\{a_0\}, \emptyset, \emptyset, \{a_0 \mapsto \{C_0\}\}).$$

The algorithm applies completion rules to the completion system until an obvious inconsistency (clash) is detected or no completion rule is applicable any more. Before we define the completion rules for $\mathcal{ALC}(\mathcal{C})$, we introduce an operation that is used by completion rules to add new nodes to completion trees.

Definition 5 ($\oplus$ Operation). *An abstract or concrete node is called* fresh *w.r.t. a completion tree T if it does not appear in T. Let $S = (T, \mathcal{N})$ be a completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. We use the following operations:*

- *$S \oplus aRb$ ($a \in \mathsf{V_a}$, $b \in \mathsf{O_a}$ fresh in T, $R \in \mathsf{N_R}$) yields a completion system obtained from S in the following way:*
  - *if $R \notin \mathsf{N_{aF}}$ or $R \in \mathsf{N_{aF}}$ and a has no R-successors, then add b to $\mathsf{V_a}$, (a,b) to E and set $\mathcal{L}(a,b) = R$, $\mathcal{L}(b) = \emptyset$;*
  - *if $R \in \mathsf{N_{aF}}$ and there is a $c \in \mathsf{V_a}$ such that $(a,c) \in E$ and $\mathcal{L}(a,c) = R$ then rename c in T with b.*
- *$S \oplus agx$ ($a \in \mathsf{V_a}$, $x \in \mathsf{O_c}$ fresh in T, $g \in \mathsf{N_{cF}}$) yields a completion system obtained from S in the following way:*
  - *if a has no g-successors, then add x to $\mathsf{V_c}$, (a,x) to E and set $\mathcal{L}(a,x) = g$;*
  - *if a has a g-successor y, then rename y in T and $\mathcal{N}$ with x.*

Let $u = R_1 \cdots R_n g$ be a path. With $S \oplus aux$, where $a \in \mathsf{V_a}$ and $x \in \mathsf{O_c}$ is fresh in T, we denote the completion system obtained from S by taking distinct nodes $b_1, ..., b_n \in \mathsf{O_a}$ which are fresh in T and setting

$$S' := S \oplus aR_1b_1 \oplus \cdots \oplus b_{n-1}R_nb_n \oplus b_ngx$$

To ensure termination of the tableau algorithm, we need a mechanism for detecting cyclic expansions, commonly called *blocking*. Informally, we detect nodes in the completion tree "similar" to previously created ones and "block" them, i.e., apply no more completion rules to such nodes. To define the blocking condition, we need a couple of notions. For $a \in \mathsf{V_a}$, define:

$$\begin{aligned}
\mathsf{cs}(a) &:= \{g \in \mathsf{N_{cF}} \mid a \text{ has a } g\text{-successor}\} \\
\mathcal{N}(a) &:= \{(g\ r\ g') \mid \text{ there are } x, y \in \mathsf{V_c} \text{ such that } x \text{ is a } g\text{-successor of } a, \\
&\qquad y \text{ is a } g'\text{-successor of } a, \text{ and } (x\ r\ y) \in \mathcal{N}\} \\
\mathcal{N}'(a) &:= \{(x\ r\ y) \mid \text{there exist } g, g' \in \mathsf{cs}(a) \text{ s.t. } x \text{ is a } g\text{-successor of } a, \\
&\qquad y \text{ is a } g'\text{-successor of } a, \text{ and } (x\ r\ y) \in \mathcal{N}\}
\end{aligned}$$

$R\sqcap$	if $C_1 \sqcap C_2 \in \mathcal{L}(a)$, a is not blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(a)$, then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_1, C_2\}$
$R\sqcup$	if $C_1 \sqcup C_2 \in \mathcal{L}(a)$, a is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset$, then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$R\exists$	if $\exists R.C \in \mathcal{L}(a)$, a is not blocked, and there is no R-successor of a such that $C \in \mathcal{L}(b)$, then set $S := S \oplus aRb$ for a fresh $b \in O_a$ and $\mathcal{L}(b) := \mathcal{L}(b) \cup \{C\}$
$R\forall$	if $\forall R.C \in \mathcal{L}(a)$, a is not blocked, and b is an R-successor of a such that $C \notin \mathcal{L}(b)$, then set $\mathcal{L}(b) := \mathcal{L}(b) \cup \{C\}$
$R\exists_c$	if $\exists U_1, U_2.r \in \mathcal{L}(a)$, a is not blocked, and there exist no $x_1, x_2 \in V_c$ such that x_i is a U_i-successor of a for $i = 1, 2$ and $(x_1\ r\ x_2) \in \mathcal{N}$ then set $S := (S \oplus aU_1x_1 \oplus aU_2x_2)$ with $x_1, x_2 \in \mathsf{O_c}$ fresh and $\mathcal{N} := \mathcal{N} \cup \{(x_1\ r\ x_2)\}$
$R\forall_c$	if $\forall U_1, U_2.r \in \mathcal{L}(a)$, a is not blocked, and there are $x_1, x_2 \in \mathsf{V_c}$ such that x_i is a U_i-successor of a for $i = 1, 2$ and $(x_1\ r\ x_2) \notin \mathcal{N}$, then set $\mathcal{N} := \mathcal{N} \cup \{(x_1\ r\ x_2)\}$
$R\mathsf{net}$	if a is potentially blocked by b and $\mathcal{N}(a)$ is not complete, then non-deterministically guess a completion $\mathcal{N}'$ of $\mathcal{N}'(a)$ and set $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$
$R\mathsf{net}'$	if a is potentially blocked by b and $\mathcal{N}(b)$ is not complete, then non-deterministically guess a completion $\mathcal{N}'$ of $\mathcal{N}'(b)$ and set $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$
$R\mathsf{gci}$	if $C_\mathcal{T} \notin \mathcal{L}(a)$, then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_\mathcal{T}\}$

Fig. 3. The Completion Rules.

A *completion* of a Rel-network N is a satisfiable and complete Rel-network N' such that $V_N = V_{N'}$ and $N \subseteq N'$.

Definition 6 (Blocking). *Let $S = (T, \mathcal{N})$ be a completion system for a concept C_0 and a TBox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. Let $a, b \in \mathsf{V_a}$. We say that $a \in \mathsf{V_a}$ is*

– potentially blocked by b *if b is an ancestor of a in T, $\mathcal{L}(a) \subseteq \mathcal{L}(b)$, and $\mathsf{cs}(a) = \mathsf{cs}(b)$.*
– directly blocked *by b if a is potentially blocked by b, $\mathcal{N}(a)$ and $\mathcal{N}(b)$ are complete, and $\mathcal{N}(a) = \mathcal{N}(b)$.*

Finally, a is blocked *if it or one of its ancestors is directly blocked.*

We are now ready to define the completion rules, which are given in Figure 3. Among the rules, there are three non-deterministic ones: $R\sqcup$, $R\mathsf{net}$ and $R\mathsf{net}'$. All rules except $R\mathsf{net}$ and $R\mathsf{net}'$ are rather standard, as they are variants of the corresponding rules from existing algorithms for DLs with concrete domains, see e.g. [2]. The purpose of these additional rules is to resolve potential blocking situations into actual blocking situations (or non-blocking situations) by completing the parts of the network $\mathcal{N}$ that correspond to the "blocked" and "blocking" node. To ensure an appropriate interplay between $R\mathsf{net}/R\mathsf{net}'$, and the blocking condition and thus to guarantee termination, we apply these rules with highest precedence.

Note that the blocking mechanism obtained in this way is *dynamic* in the sense that blocking situations can be broken again after they have been established. Also note that the conditions $\mathcal{L}(a) \subseteq \mathcal{L}(b)$ and $\mathsf{cs}(a) = \mathsf{cs}(b)$ can be viewed as a refinement of pairwise blocking as known from [12]: due to path normal form, pairwise blocking is a strictly sharper condition than the above two.

The algorithm applies completion rules until no more rules are applicable (such a completion system is called *complete*), or a clash is encountered.

Definition 7 (Clash). *Let $S = (T, \mathcal{N})$ be a completion system for a concept C and a TBox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_a}, E, \mathcal{L})$. S is said to contain a* clash *if*

– *there is an $a \in \mathsf{V_a}$ and an $A \in \mathsf{N_C}$ such that $\{A, \neg A\} \subseteq \mathcal{L}(a)$, or*
– *$\mathcal{N}$ is not satisfiable in $\mathcal{C}$.*

Note that the existence of clashes is decidable since we require that satisfiability in $\mathcal{C}$ is decidable. In an actual implementation of our algorithm, checking for clashes would require calling an external reasoner for satisfiability in the constraint system used. The tableau algorithm checks for clashes before each rule application. It returns "satisfiable" if there is a way to apply the non-deterministic rules such that a complete and clash-free completion system is generated. Otherwise, it returns "unsatisfiable". In actual implementations of our algorithm, non-determinism has to be replaced by backtracking and search.

Note that checking for clashes before every rule application ensures that $R\mathsf{net}$ and $R\mathsf{net}'$ are well-defined: if $R\mathsf{net}$ is applied, then there indeed exists a completion $\mathcal{N}'$ of $\mathcal{N}(a)$ to be guessed: due to clash checking, the network $\mathcal{N}$ is satisfiable, and it is readily checked that this implies the existence of the required completion.

Theorem 2. *If $\mathcal{C}$ is an ω-admissible constraint system, the tableau algorithm decides satisfiability of $\mathcal{ALC}(\mathcal{C})$ concepts w.r.t. general TBoxes.*

Proof. Termination of the algorithm is ensured by the blocking condition, the $R\mathsf{net}$ and $R\mathsf{net}'$ rules, and the fact that these rules are executed with highest precedence. Completeness can be proved in the standard way, by showing that if the input concept C_0 and TBox $\mathcal{T}$ have a common model $\mathcal{I}$, we can guide the (non-deterministic parts of) the tableau algorithm according to $\mathcal{I}$, such that it ends up with a clash-free completion system. Detailed proofs are given in [19].

Here we sketch the soundness proof. We have to show, that, if the tableau algorithm returns "satisfiable", then the input concept C_0 is satisfiable w.r.t. the input TBox $\mathcal{T}$. If the tableau algorithm returns "satisfiable", then there exists a complete and clash-free completion system $S = (T, \mathcal{N})$ of C_0 and $\mathcal{T}$. Let $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, and let $\mathsf{root} \in \mathsf{V_a}$ denote the root of T. Our aim is to define a model $\mathcal{I}$ of C_0 and $\mathcal{T}$. We proceed in several steps.

Let blocks be a function that for every directly blocked $b \in \mathsf{V_a}$, returns an unblocked $a \in \mathsf{V_a}$ such that b is blocked by a in S. It can easliy seen that, by definition of blocking, such a node a always exists. A *path* in S is a (possibly empty) sequence of pairs of nodes $\frac{a_1}{b_1}, \ldots, \frac{a_n}{b_n}$, with $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ from $\mathsf{V_a}$, such that, for $1 \leq i < n$, b_{i+1} is a successor of a_i in T and

$$a_{i+1} := \begin{cases} b_{i+1} & \text{if } b_{i+1} \text{ is not blocked,} \\ \mathsf{blocks}(b_{i+1}) & \text{otherwise.} \end{cases}$$

We use Paths to denote the set of all paths in S, $\mathsf{head}(p)$ to denote the first pair of a path p and $\mathsf{tail}(p)$ to denote the last pair of p (if p is nonempty). We now define the "abstract part" of the the model $\mathcal{I}$ we are constructing:

$$\begin{aligned} \Delta_{\mathcal{I}} &:= \{p \in \mathsf{Paths} \mid p \text{ non-empty and } \mathsf{head}(p) = \frac{\mathsf{root}}{\mathsf{root}}\} \\ A^{\mathcal{I}} &:= \{p \in \Delta_{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a}{b} \text{ and } A \in \mathcal{L}(a)\}, \\ R^{\mathcal{I}} &:= \{(p, p \cdot \frac{a}{b}) \in \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a'}{b'} \text{ and } b \text{ is } R\text{-successor of } a' \text{ in } T\ \} \end{aligned}$$

for all $A \in \mathsf{N_C}$ and $R \in \mathsf{N_R}$. Observe that $\Delta_{\mathcal{I}}$ is non-empty since $\frac{\mathsf{root}}{\mathsf{root}} \in \Delta_{\mathcal{I}}$, and that $f^{\mathcal{I}}$ is functional for every $f \in \mathsf{N_{aF}}$, which is ensured by the "$\oplus$" operation and by definition of Paths.

Intuitively, the abstract part of $\mathcal{I}$ as defined above is "patched together" from (copies of) parts of the completion tree T. For defining the concrete part of $\mathcal{I}$, we make this patching explicit: For $p, q \in \mathsf{Paths}$,

- p is called a *hook* if $p = \frac{\mathsf{root}}{\mathsf{root}}$ or $\mathsf{tail}(p) = \frac{a}{b}$ with $a \neq b$ (and thus b blocked by a). We use Hooks to denote the set of all hooks.
- we call p a *q-companion* if q is a hook and there exists $q' \in \mathsf{Paths}$ such that $p = qq'$ and all nodes $\frac{a}{b}$ in q' satisfy $a = b$, with the possible exception of $\mathsf{tail}(q')$.

Intuitively, the hooks, which are induced by blocking situations in T, are the points where we patch together parts of T. The part of T patched at a hook p with $\mathsf{tail}(p) = \frac{a}{b}$ is comprised of (copies of) all the nodes c in T that are reachable from a, except indirectly blocked ones. Formally, the part of $\mathcal{I}$ belonging to the hook p is defined as $P(p) := \{q \in \Delta_{\mathcal{I}} \mid q \text{ is a } p\text{-companion}\}$. For $p, q \in \mathsf{Hooks}$, q is called a *successor* of p if q is a p-companion and $p \neq q$. Observe that, for each hook p, $P(p)$ includes all successor hooks of p. Intuitively, this means that the parts patched together to obtain the abstract part of $\mathcal{I}$ are overlapping at the hooks.

For space limitations, we only sketch how the concrete part of $\mathcal{I}$ is defined. The full construction with proofs can be found in [19]. Since the completion system S is clash-free, its constraint network $\mathcal{N}$ is satisfiable. Therefore, there exists a completion $\mathcal{N}^c$ of $\mathcal{N}$. For every $p \in \mathsf{Hooks}$, we define a constraint network $N(p)$ that defines the constraints that have to be satisfied by the concrete part of $\mathcal{I}$ corresponding to $P(p)$. More precisely, $N(p)$ is defined as (a copy of) the part of $\mathcal{N}^c$ that corresponds to the part of T patched at p.

Then the network $\mathbf{N} = \bigcup_{p \in \mathsf{Hooks}} N(p)$ describes the constraints that have to be satisfied by the concrete part of the whole model $\mathcal{I}$. By construction, the networks $N(p)$ are finite, complete, satisfiable, and overlap at the hooks. Due to the blocking condition, their overlapping parts are identical. Thus, we can use the patchwork and compactness property of $\mathcal{C}$ to show that $\mathbf{N}$ is satisfiable in $\mathcal{C}$. Then a model $M_{\mathcal{I}} \in \mathfrak{M}$ of $\mathbf{N}$ becomes the last argument of our interpretation $\mathcal{I}$, and we can define extensions of concrete features in $\mathcal{I}$. To show that $\mathcal{I}$ is indeed a model of C_0 and $\mathcal{T}$, we can prove by structural induction that for all $p \in \Delta_{\mathcal{I}}$ with $\mathsf{tail}(p) = \frac{a}{b}$ and for all $C \in \mathsf{sub}(C_0, \mathcal{T})$ the following holds: if $C \in \mathcal{L}(a)$ then $p \in C^{\mathcal{I}}$. Since $C_0 \in \mathcal{L}(\mathsf{root})$ we have that $\frac{\mathsf{root}}{\mathsf{root}} \in C_0^{\mathcal{I}}$. Finally, $C_{\mathcal{T}} \in \mathcal{L}(a)$ for all unblocked $a \in \mathsf{V_a}$ implies that $p \in C_{\mathcal{T}}^{\mathcal{I}}$ for all $p \in \Delta_{\mathcal{I}}$, and thus $\mathcal{I}$ models $\mathcal{T}$.

5 Conclusion

We have proved decidability of $\mathcal{ALC}$ with ω-admissible constraint systems and GCIs. We conjecture that, by mixing the techniques from the current paper with those from [17, 14], it is possible to prove ExpTime-completeness of satisfiability in $\mathcal{ALC}(\mathcal{C})$ provided that satisfiability in $\mathcal{C}$ can be decided in ExpTime. Various language extensions, both on the logical and concrete side, should also be possible in a straightforward way.

We also exhibited the first tableau algorithm for DLs with concrete domains and GCIs in which the concrete domain constructors are not limited to concrete features. We view this algorithm as a first step towards an implementation, although there is clearly room for improvements: the rules $R\mathsf{net}$ and $R\mathsf{net}'$ add considerable non-determinism, clash checking involves the whole network $\mathcal{N}$ rather than only a local part of it, and blocking can be further refined.

We believe that, in general, getting rid of the additional non-determinism introduced by $R\mathsf{net}$ and $R\mathsf{net}'$ is difficult. One possible way out may be to permit only a single concrete feature: then $R\mathsf{net}$ and $R\mathsf{net}'$ become deterministic (in fact they can be omitted), and "potentially blocking" coincides with "directly blocking". We believe that having only one concrete feature is actually rather natural: for the Allen/RCC8 concrete domains, the concrete feature could be $\mathsf{hasTime}$ and $\mathsf{hasLocation}$, respectively.

However, a complication is posed by the fact that path normal form introduces additional concrete features. Simply requiring, as an additional restriction, that only concepts and TBoxes in PNF are allowed is rather severe: it can be seen that, then, satisfiability in $\mathcal{ALC}(\mathcal{C})$ instantiated with the RCC8 and Allen constraint systems can be decided by adding some simple clash conditions. In particular, there is no need to use an external reasoner for the constraint system at all. Therefore, it is more interesting to find a tableau algorithm for $\mathcal{ALC}(\mathcal{C})$ with only one concrete feature that does not rely on PNF, but still avoids the non-determinism and global satisfiability check of $\mathcal{N}$.

References

1. J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

2. F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, Morgan Kaufman. 1991.
3. F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jörg Siekmann*, LNAI. Springer-Verlag, 2003.
4. F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications.* Cambridge University Press, 2003.
5. P. Balbiani and J.-F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *Proc. of FroCoS 2002*, number 2309 in LNAI, pages 162–176. Springer, 2002.
6. B. Bennett. Modal logics for qualitative spatial reasoning. *Journal of the IGPL*, 4(1), 1997.
7. D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proc. of ECAI-96*, pages 303–307, 1996.
8. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In *Logics for Databases and Information Systems*, pages 229–263. Kluwer, 1998.
9. D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications.* Elsevier, 2003.
10. V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR'01*, number 2083 in LNAI, pages 701–705. Springer-Verlag, 2001.
11. I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proc. of KR98*, pages 636–647, 1998.
12. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer, 1999.
13. C. Lutz. Complexity of terminological reasoning revisited. In *Proc. of LPAR'99*, number 1705 in LNAI, pages 181–200. Springer, 1999.
14. C. Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic—First results. In *Proc. of KR2002*, pages 191–202. Morgan Kaufman, 2002.
15. C. Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. In *Proc. of DL2002*, number 53 in CEUR-WS (http://ceur-ws.org/), pages 185–194, 2002.
16. C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003.
17. C. Lutz. Combining interval-based temporal reasoning with general tboxes. *Artificial Intelligence*, 152(2):235–274, 2004.
18. C. Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.
19. C. Lutz and M. Miličić. A tableau algorithm for DLs with concrete domains and GCIs. LTCS-Report 05-07, TU Dresden, 2005. See http://lat.inf.tu-dresden.de/research/reports.html.
20. D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. of KR'92*, pages 165–176. Morgan Kaufman, 1992.
21. J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1–2):69–123, 1999.
22. M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: a revised report. In *Readings in qualitative reasoning about physical systems*, pages 373–381. Morgan Kaufmann, 1990.