A Tableau Algorithm for Description Logics with Concrete Domains and General TBoxes

Carsten Lutz and Maja Miličić
Institute of Theoretical Computer Science
TU Dresden, Germany

Abstract. To use description logics (DLs) in an application, it is crucial to identify a DL that is sufficiently expressive to represent the relevant notions of the application domain, but for which reasoning is still decidable. Two means of expressivity that are required by many modern applications of DLs are *concrete domains* and *general TBoxes*. The former are used for defining concepts based on concrete qualities of their instances such as the weight, age, duration, and spatial extension. The purpose of the latter is to capture background knowledge by stating that the extension of a concept is included in the extension of another concept. Unfortunately, it is well-known that combining concrete domains with general TBoxes often leads to DLs for which reasoning is 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 general TBoxes. 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 general TBoxes.

Keywords: Description logic, concrete domains, decidability, tableau algorithm

Table of Contents

1	Introduction	2
2	Constraint Systems	3
	2.1 RCC8	4
	2.2 Allen's Relations	6
	2.3 Properties of Constraint Systems	7
3	Syntax and Semantics	8
4	A Tableau Algorithm for $\mathcal{ALC}(\mathcal{C})$	11
	4.1 Normal Forms	11
	4.2 Data Structures	13
	4.3 The Tableau Algorithm	16
	4.4 Correctness	18
5	Practicability	29
6	Conclusion	31
A	Properties of RCC8	35
B	Properties of Allen	39

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, for a given application, one can 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 an avoidable 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 TBoxes* [3, 8, 22]. 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. General TBoxes play an important role in modern DLs as they allow to represent background knowledge of application domains by stating via inclusions $C \sqsubseteq D$ that the extension of a concept C is included in the extension of a concept D.

Unfortunately, combining concrete domains with general TBoxes easily leads to undecidability. For example, it has been shown in [25] that the basic DL $\mathcal{ALC}$ extended with general TBoxes and a rather inexpressive concrete domain based on the natural numbers and providing for equality and incrementation predicates is undecidable, see also the survey paper [23]. In view of this discouraging result, it is a natural question whether there are *any* useful concrete domains that can be combined with general TBoxes in a decidable DL. A positive answer to this question has been given in [24] and [20], 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 reals and equipped with various unary and binary predicates such as "$\leq$", "$>_5$", and "$\neq$". Using an automata-based approach, it has been shown in [24, 20] that reasoning in the DLs $\mathcal{ALC}$ and $\mathcal{SHIQ}$ extended with these concrete domains and general TBoxes is decidable and ExpTime-complete.

The purpose of this paper it to advance the knowledge about decidable DLs with both concrete domains and general TBoxes. Our contribution is two-fold: first, instead of focusing 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 general TBoxes. 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 real line and the Allen relations [1], and a spatial one based on the real plane and the RCC8 relations [9, 6, 29]. 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, we develop a *tableau algorithm* for DLs with both general TBoxes and concrete domains. This algorithm is used to establish a general decidability result for $\mathcal{ALC}$ equipped with general TBoxes and any ω-admissible concrete domain. In particular, we obtain decidability of $\mathcal{ALC}$ with general TBoxes and the Allen relations as first established in [24], and, as a new result, prove decidability of $\mathcal{ALC}$ with general TBoxes and the RCC8 relations as a concrete domain. In contrast to existing tableau algorithms [13, 17], we do not impose any restrictions on the concrete domain constructor. 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 [14, 12], we view our algorithm as a first step towards an efficient implementation of description logics with (ω-admissible) concrete domains and general TBoxes. In particular, we identify an expressive fragment of our logic that should be easily integrated into existing DL reasoners.

This paper is organized as follows: in Section 2, we introduce constraint systems and define ω-admissibility. In Section 3, we introduce the description logic $\mathcal{ALC}(\mathcal{C})$ that incorporates constraint systems and general TBoxes. The tableau algorithm for deciding satisfiability in $\mathcal{ALC}(\mathcal{C})$ is developed in Section 4. In Section 5, we discuss the feasibility of our algorithm and identify a fragment for which the tableau algorithm is implementable in a particularly straightforward way.

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. Examples for such systems include spatial constraint networks based on the RCC8 relations [9, 6, 30] or on cardinal direction relations [10], and temporal constraint networks based on Allen's relations of time intervals [1, 34, 28] or on relations between time points [33, 34].

Definition 1 (Rel-network). *Let* Var *be a countably infinite set of variables and* Rel *a finite set of binary relation symbols. A* Rel-constraint

is an expression $(x\ r\ y)$ *with* $x, y \in \mathsf{Var}$ *and* $r \in \mathsf{Rel}$. *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* $x, y \in V_N$, *there is exactly one constraint* $(x\ r\ y) \in N$.

We define the semantics of Rel-network by using complete Rel-networks as models. Intuitively, the nodes in these complete networks should be viewed as concrete values rather than as variables. Equivalently to our network-based semantics, we could proceed as in constraint satisfaction problems, associate each variable with a set of values, and view relations as constraints on these values, see e.g. [31].

Definition 2 (Model, Constraint System). *Let* N *be a* Rel-*network and* N' *a complete* Rel-*networks. We say that* N' *is a* model *of* N *if there is a mapping* $\tau : V_N \to V_{N'}$ *such that* $(x\ r\ y) \in N$ *implies* $(\tau(x)\ r\ \tau(y)) \in N'$.

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.

To emphasize the different role of variables in Rel-networks and in models, we denote variables in the former with $x, y, \ldots$ and in the latter with v, v', etc. Note that Rel-networks used as models have to be complete, which corresponds to the relations in Rel to be jointly exhaustive and mutually exclusive.

Equivalently to our network-based semantics, we could proceed as in constraint satisfaction problems, associate each variable with a set of values, and view relations as constraints on these values, see e.g. [31].

In the following two subsections, we introduce two example constraint systems: one for spatial reasoning based on the RCC8 topological relations in the real plane, and one for temporal reasoning based on the Allen relations in the real line.

2.1. RCC8

The RCC8 relations, which are illustrated in Figure 1, are intended to describe the relation between regions in topological spaces [29]. In this paper, we will use the standard topology of the real plane which is one of the most appropriate 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. 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

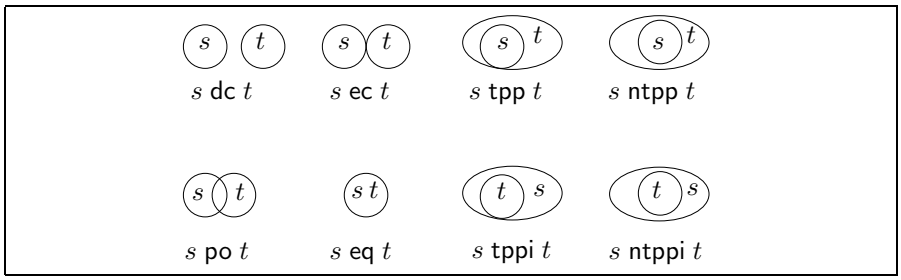

Figure 1. The eight RCC8 relations.

all $s, t \subseteq U$, we have

$$\begin{array}{rclcrcl} \mathbb{I}(U) & = & U & & \mathbb{I}(s) & \subseteq & s \\ \mathbb{I}(s) \cap \mathbb{I}(t) & = & \mathbb{I}(s \cap t) & & \mathbb{II}(s) & = & \mathbb{I}(s). \end{array}$$

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 $\mathbb{CI}(s) = s$. 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 $U_{\mathfrak{T}} \times U_{\mathfrak{T}}$:

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

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 $\mathfrak{T}_{\mathbb{R}^2}$. Then 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

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

Note that using only regular closed sets excludes sub-dimensional regions such as points and lines. This is necessary for the RCC8 relations to be jointly exhaustive and pairwise disjoint.

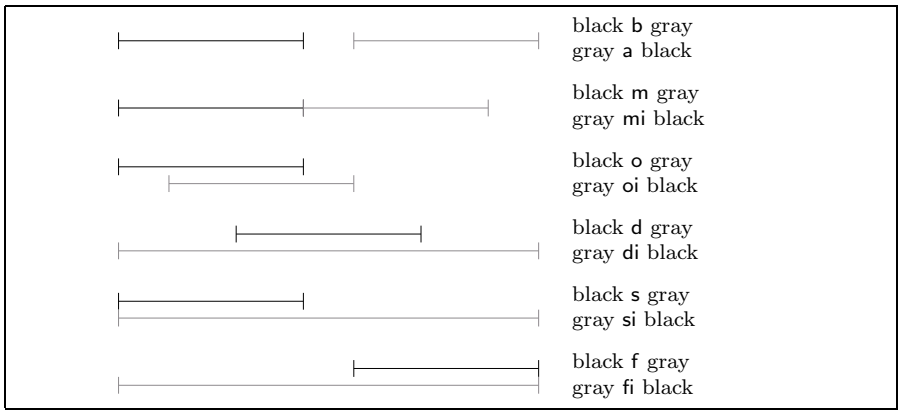

Figure 2. The thirteen Allen relations.

2.2. Allen's Relations

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 2. As the flow of time, we use the real numbers with the usual ordering. Let $\mathsf{Int}_{\mathbb{R}}$ denote the set of all closed intervals $[r_1, r_2]$ over $\mathbb{R}$ with $r_1 < r_2$, i.e., point-intervals are not admitted. The extension $\mathsf{r}^{\mathbb{R}}$ of each Allen relation r is a subset of $\mathsf{Int}_{\mathbb{R}} \times \mathsf{Int}_{\mathbb{R}}$. It is defined in terms of the relationships between endpoints in the obvious way, c.f. Figure 2. We define the constraint system

$$\mathsf{Allen}_{\mathbb{R}} = \langle \mathsf{Allen}, \mathfrak{M}_{\mathbb{R}} \rangle$$

by setting $\mathfrak{M}_{\mathbb{R}} := \{N_{\mathbb{R}}\}$, where $N_{\mathbb{R}}$ is defined by fixing a variable $v_i \in \mathsf{Var}$ for every $i \in \mathsf{Int}_{\mathbb{R}}$ and setting

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

We could also define the constraint system $\mathsf{Allen}_{\mathbb{Q}}$ based on the rationals rather than on the reals: this has no impact on the satisfiability of finite and infinite Allen-networks (which are countable by definition). If we use the natural numbers or the integers, this still holds for finite networks, but not for infinite ones: there are infinite Allen-networks that are satisfiable over the reals and rationals, but not over the natural number and integers.

2.3. 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 that constraint systems satisfy certain properties. First, we need to ensure that satisfiable networks (satisfying some additional conditions) can be "patched" together to a joint network that is also satisfiable. This is ensured by the patchwork property.

Definition 3 (Patchwork Property). *Let $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ be a constraint system, and let N, M be finite complete* Rel*-networks such that, for the intersection parts*

$$I_{N,M} := \{(x\ r\ y) \mid x, y \in V_N \cap V_M \text{ and } (x\ r\ y) \in N\}$$
$$I_{M,N} := \{(x\ r\ y) \mid x, y \in V_N \cap V_M \text{ and } (x\ r\ y) \in M\}$$

we have $I_{N,M} = I_{M,N}$. Then the composition *of N and M is defined as $N \cup M$. We say that $\mathcal{C}$ has the* patchwork property *if the following holds: if N and M are satisfiable then $N \cup M$ is satisfiable.*

The patchwork property is similar to the property of constraint networks formulated by Balbiani in [5], where constraint networks are combined with linear temporal logic.

For using constraint systems with the DL tableau algorithm presented in this paper, we must be sure that, even if we patch together an infinite number of satisfiable networks, the resulting (infinite) network is still satisfiable. This is guaranteed by the compactness property.

Definition 4 (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 $\{(x\, r\, y) \in N \mid x, y \in V\} \subseteq N$. Then $\mathcal{C}$ 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 $\mathcal{C}$.*

Finally, our tableau algorithm has to check satisfiability of certain $\mathcal{C}$-networks. Thus, we have to assume that $\mathcal{C}$-satisfiability is decidable. The properties of constraint systems we require are summarized in the following definition.

Definition 5 (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 of finite $\mathcal{C}$-networks is decidable;*

2. *$\mathcal{C}$ has the patchwork property (c.f. Definition 3);*

3. *$\mathcal{C}$ has the compactness property (c.f. Definition 4).*

In Appendixes A and B, we prove that $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{R}}$ satisfy the patchwork property and the compactness property. Moreover, satisfiability of finite networks is NP-complete (and thus decidable) in both systems: this is proved in [34] for $\mathsf{Allen}_{\mathbb{R}}$ and in [30] for $\mathsf{RCC8}_{\mathbb{R}^2}$. Thus, $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{R}}$ are ω-admissible.

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.

Definition 6 ($\mathcal{ALC}(\mathcal{C})$-concepts). *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 $\mathsf{N_R}$ is partitioned into two countably infinite subsets $\mathsf{N_{aF}}$ and $\mathsf{N_{sR}}$. The elements of $\mathsf{N_{aF}}$ are called* abstract features *and the elements of $\mathsf{N_{sR}}$* standard roles. *A* path *of length $k+1$ with $k \geq 0$ 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 the smallest set such that*[1]

1. *every concept name $A \in \mathsf{N_C}$ is a concept,*

2. *if C and D are concepts and $R \in \mathsf{N_R}$, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall R.C$, and $\exists R.C$ are concepts;*

3. *if u_1 and u_2 are feature paths and $r_1, \ldots, r_k \in \mathsf{Rel}$, then the following are also concepts:*

$$\exists u_1, u_2.(r_1 \vee \cdots \vee r_k) \text{ and } \forall u_1, u_2.(r_1 \vee \cdots \vee r_k);$$

4. *if U_1 and U_2 are paths of length at most two and $r_1, \ldots, r_k \in \mathsf{Rel}$, then the following are also concepts:*

$$\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \text{ and } \forall U_1, U_2.(r_1 \vee \cdots \vee r_k);$$

[1] This is an extension of the language introduced in the conference version of this paper [26].

A concept inclusion *is an expression of the form* $C \sqsubseteq D$*, where* C *and* D *are concepts. We use* $C \doteq D$ *as abbreviation for the two concept inclusions* $C \sqsubseteq D$ *and* $D \sqsubseteq C$*. A finite set of concept inclusions is called a* TBox*.*

Observe that we restrict the length of paths inside the constraint-based constructor to two only if at least one of the paths contains a standard role. The TBox formalism introduced in Definition 6 is often called *general TBox* [4] since it subsumes several weaker variants [7, 19]. Throughout this paper, we use $\top$ as abbreviation for an arbitrary propositional tautology and $C \rightarrow D$ for $\neg C \sqcup D$.

Definition 7 ($\mathcal{ALC}(\mathcal{C})$ Semantics). *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* $\Delta_\mathcal{I}$*;*
- *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* $M_\mathcal{I}$*.*

If $\mathbf{r} = r_1 \vee \cdots \vee r_k$*, where* $r_1, \ldots, r_k \in \mathsf{Rel}$*, we write* $M_\mathcal{I} \models (x\ \mathbf{r}\ y)$ *iff there exists an* $i \in \{1, \ldots, k\}$ *such that* $(x\ r_i\ y) \in M_\mathcal{I}$*. The interpretation function is then extended to arbitrary concepts as follows:*

$$
\begin{aligned}
\neg C^\mathcal{I} &:= \Delta^\mathcal{I} \setminus C^\mathcal{I},\\
(C \sqcap D)^\mathcal{I} &:= C^\mathcal{I} \cap D^\mathcal{I},\\
(C \sqcup D)^\mathcal{I} &:= C^\mathcal{I} \cup D^\mathcal{I},\\
(\exists R.C)^\mathcal{I} &:= \{d \in \Delta^\mathcal{I} \mid \exists e \in \Delta^\mathcal{I} \text{ with } (d,e) \in R^\mathcal{I} \text{ and } e \in C^\mathcal{I}\},\\
(\forall R.C)^\mathcal{I} &:= \{d \in \Delta^\mathcal{I} \mid \forall e \in \Delta^\mathcal{I}, \text{ if } (d,e) \in R^\mathcal{I}, \text{ then } e \in C^\mathcal{I}\},\\
(\exists U_1, U_2.\mathbf{r})^\mathcal{I} &:= \{d \in \Delta^\mathcal{I} \mid \exists v_1 \in U_1^\mathcal{I}(d) \text{ and } v_2 \in U_2^\mathcal{I}(d)\\
&\qquad\qquad \text{with } M_\mathcal{I} \models (v_1\ \mathbf{r}\ v_2)\},\\
(\forall U_1, U_2.\mathbf{r})^\mathcal{I} &:= \{d \in \Delta^\mathcal{I} \mid \forall v_1 \in U_1^\mathcal{I}(d) \text{ and } v_2 \in U_2^\mathcal{I}(d),\\
&\qquad\qquad \text{we have } M_\mathcal{I} \models (v_1\ \mathbf{r}\ v_2)\}
\end{aligned}
$$

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

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

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 concept inclusions* $C \sqsubseteq D$ *in* $\mathcal{T}$*.*

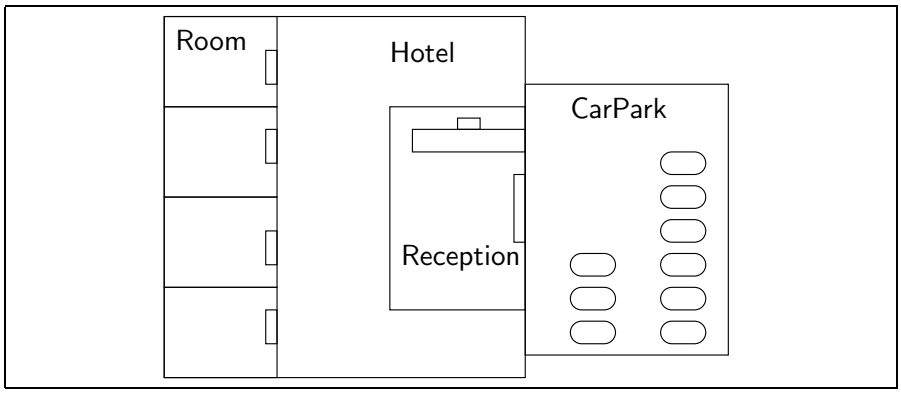

Figure 3. An example of a CarFriendlyHotel.

Observe that the network $\mathfrak{M}$ in Definition 7 is a model of the constraint system $\mathcal{C}$, whence variables in this network correspond to values in $\mathcal{C}$ and are denoted with v, v' rather than x, y.

The following example TBox describes some properties of a hotel using the constraint system $\mathsf{RCC8}_{\mathbb{R}^2}$, where has-room is a role, has-reception and has-carpark are abstract features (assuming that a hotel has at most a single reception and car park), loc is a concrete feature, and all capitalized words are concept names.

$$\begin{aligned}
\mathsf{Hotel} &\sqsubseteq \forall \mathsf{has\text{-}room}.\mathsf{Room} \sqcap \forall \mathsf{has\text{-}reception}.\mathsf{Reception} \\
&\quad \sqcap \forall \mathsf{has\text{-}carpark}.\mathsf{CarPark} \\
\mathsf{Hotel} &\sqsubseteq \forall (\mathsf{has\text{-}room}\ \mathsf{loc}), (\mathsf{loc}).\mathsf{tpp} \vee \mathsf{ntpp} \\
&\quad \sqcap \forall (\mathsf{has\text{-}room}\ \mathsf{loc}), (\mathsf{has\text{-}room}\ \mathsf{loc}).\mathsf{dc} \vee \mathsf{ec} \vee \mathsf{eq} \\
\mathsf{CarFriendlyHotel} &\doteq \mathsf{Hotel} \sqcap \exists (\mathsf{has\text{-}reception}\ \mathsf{loc}), (\mathsf{loc}).\mathsf{tpp} \\
&\quad \sqcap \exists (\mathsf{has\text{-}carpark}\ \mathsf{loc}), (\mathsf{loc}).\mathsf{ec} \\
&\quad \sqcap \exists (\mathsf{has\text{-}carpark}\ \mathsf{loc}), (\mathsf{has\text{-}reception}\ \mathsf{loc}).\mathsf{ec}
\end{aligned}$$

The first concept inclusion expresses that hotels are related via the three roles to objects of the proper type. The second concept inclusion says that the rooms of a hotel are spatially contained in the hotel, and that rooms do not overlap. Finally, the last concept inclusion describes hotels that are convenient for car owners: they have a carpark that is directly next to the reception. This situation is illustrated in Figure 3.

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. $\mathcal{T}$. This allows us to concentrate on concept satisfiability when devising reasoning procedures.

4. A Tableau Algorithm for $\mathcal{ALC}(\mathcal{C})$

We present a tableau algorithm that decides satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts w.r.t. TBoxes. Tableau algorithms are among the most popular decision procedures for description logics since they are amenable to various optimization techniques and often can be efficiently implemented. Therefore, we view the algorithm presented in this paper as a first step towards practicable reasoning with concrete domains and general TBoxes. On the flipslide, algorithms such as the one developed in this section usually do not yield tight upper complexity bounds.

The algorithm developed in the following is independent of the constraint system $\mathcal{C}$. This is achieved by delegating reasoning in $\mathcal{C}$ to an external reasoner that decides satisfiability of $\mathcal{C}$-networks. Throughout this section, we assume $\mathcal{C}$ to be ω-admissible.

4.1. Normal Forms

It is convenient to first convert the input concept and TBox into an appropriate syntactic form. More precisely, we convert concepts and TBoxes into negation normal form (NNF) and restrict the length of paths that appear inside the constraint-based concept constructors. We start with describing NNF conversion. A concept is said to be in *negation normal form* if negation occurs only in front of concept names. The following lemma shows that NNF can be assumed without loss of generality. For a path $U = R_1 \cdots R_k g$, we write $\mathsf{ud}(U)$ to denote the concept $\forall R_1. \cdots \forall R_k.(\forall g, g.r \sqcap \forall g, g.r')$ where $r, r' \in \mathsf{Rel}$ are arbitrary such that $r \neq r'$.[2]

Lemma 1 (NNF Conversion). *Exhaustive application of the following rewrite rules translates $\mathcal{ALC}(\mathcal{C})$-concepts to equivalent ones in NNF.*

$$\begin{aligned} \neg\neg C &\rightsquigarrow C \\ \neg(C \sqcap D) &\rightsquigarrow \neg C \sqcup \neg D \\ \neg(C \sqcup D) &\rightsquigarrow \neg C \sqcap \neg D \\ \neg(\exists R.C) &\rightsquigarrow (\forall R.\neg C) \end{aligned}$$

[2] This presupposes the natural assumptions that Rel has cardinality at least two.

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

$$\neg(\forall U_1, U_2.(r_1 \vee \cdots \vee r_k)) \rightsquigarrow \begin{cases} \bot & \textit{if } \mathsf{Rel} = \{r_1, \ldots, r_k\} \\ \exists U_1, U_2.(\bigvee_{r \in \mathsf{Rel} \setminus \{r_1, \ldots, r_k\}} r) & \textit{otherwise} \end{cases}$$

$$\neg(\exists U_1, U_2.(r_1 \vee \cdots \vee r_k)) \rightsquigarrow \begin{cases} \mathsf{ud}(U_1) \sqcup \mathsf{ud}(U_2) & \textit{if } \mathsf{Rel} = \{r_1, \ldots, r_k\} \\ \forall U_1, U_2.(\bigvee_{r \in \mathsf{Rel} \setminus \{r_1, \ldots, r_k\}} r) & \textit{otherwise} \end{cases}$$

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

In Lemma 1, the last two transformations are equivalence preserving since the Rel-networks used as models in $\mathcal{C}$ are complete.

We now show how to restrict the length of paths by converting concepts and TBoxes into path normal form. This normal form was first considered in [24] in the context of the description logic $\mathcal{TDL}$ and in [20] in the context of $\mathbb{Q}$-$\mathcal{SHIQ}$.

Definition 8 (Path Normal Form). *An* $\mathcal{ALC}(\mathcal{C})$*-concept* C *is in* path normal form (PNF) *if it is in NNF and for all subconcepts*

$$\exists U_1, U_2.(r_1 \vee \ldots \vee r_k) \textit{ and } \forall U_1, U_2.(r_1 \vee \ldots \vee r_k)$$

of C*, the length of* U_1 *and* U_2 *is at most two. An* $\mathcal{ALC}(\mathcal{C})$*-TBox* $\mathcal{T}$ *is in path normal form iff* $\mathcal{T}$ *is of the form* $\{\top \sqsubseteq C\}$*, with* C *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 reduced in polynomial time to satisfiability of* $\mathcal{ALC}(\mathcal{C})$*-concepts in PNF w.r.t. TBoxes in PNF.*

Proof. We first define an auxiliary mapping and then use this mapping to translate $\mathcal{ALC}(\mathcal{C})$-concepts into equivalent ones in PNF. Let C be an $\mathcal{ALC}(\mathcal{C})$-concept. By Lemma 1, we may assume w.l.o.g. that C is in NNF. 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 fresh concrete features. We inductively define a mapping λ from feature paths u in C to concepts as follows:

$$\begin{aligned} \lambda(g) &= \top \\ \lambda(fu) &= (\exists f[u], [fu].=) \sqcap \exists f.\lambda(u) \end{aligned}$$

For every $\mathcal{ALC}(\mathcal{C})$-concept C, a corresponding concept $\rho(C)$ is obtained as follows: first replace all subconcepts $\forall u_1, u_2.(r_1 \vee \cdots \vee r_k)$ (with u_1, u_2 feature paths) with

$$\mathsf{ud}(u_1) \sqcup \mathsf{ud}(u_2) \sqcup \exists u_1, u_2.(r_1 \vee \cdots \vee r_k)$$

Then replace all subconcepts $\exists u_1, u_2.(r_1 \vee \cdots \vee r_k)$ with

$$\exists [u_1], [u_2].(r_1 \vee \cdots \vee r_k) \sqcap \lambda(u_1) \sqcap \lambda(u_2).$$

We extend the mapping ρ to TBoxes. For a TBox $\mathcal{T}$ we define

$$D_{\mathcal{T}} := \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \mathsf{nnf}(C \to D).$$

and set

$$\rho(\mathcal{T}) = \{\top \sqsubseteq \rho(D_{\mathcal{T}})\}.$$

Clearly, $\rho(C)$ and $\rho(\mathcal{T})$ are in PNF and the translation can be done in polynomial time. Moreover, it is easy to check that C is satisfiable w.r.t. $\mathcal{T}$ iff $\rho(C)$ is satisfiable w.r.t. $\rho(\mathcal{T})$: if $\mathcal{I}$ is a model of $\rho(C)$ and $\rho(\mathcal{T})$, then it can be seen that $\mathcal{I}$ is also a model of C and $\mathcal{T}$ as well. For the other direction, let $\mathcal{I}$ be a model of C and $\mathcal{T}$. A model $\mathcal{J}$ of $\rho(C)$ and $\rho(\mathcal{T})$ is obtained by extending $\mathcal{I}$ with the interpretion of freshly introduced concrete features in the following way:

$$[f_1 \dots f_n g]^{\mathcal{J}} := f_1^{\mathcal{J}} \circ \dots \circ f_n^{\mathcal{J}} \circ g^{\mathcal{J}}.$$

□

The previous lemma shows that, in what follows, we may assume w.l.o.g. that all concepts and TBoxes are in PNF.

4.2. Data Structures

We introduce the data structures underlying the tableau algorithm, an operation for extending this data structure, and a cycle detection mechanism that is needed to ensure termination of the algorithm. As already said, we assume that the input concept C_0 is in PNF, and that the input TBox $\mathcal{T}$ is of the form $\mathcal{T} = \{\top \sqsubseteq C_{\mathcal{T}}\}$, where $C_{\mathcal{T}}$ is in PNF.

The main ingredient of the data structure underlying our algorithm is a tree that, in case of a successful run of the algorithm, represents a single model of the input concept and TBox. Due to the presence of the constraint system $\mathcal{C}$, this tree has two types of nodes: abstract ones that represent individuals of the logic domain $\Delta_{\mathcal{I}}$ and concrete ones that represent values of the concrete domain. We use $\mathsf{sub}(C)$ to denote the set of subconcepts of the concept C and set $\mathsf{sub}(C_0, \mathcal{T}) := \mathsf{sub}(C_0) \cup \mathsf{sub}(C_{\mathcal{T}})$.

Definition 9 (Completion system). *Let* $\mathsf{O_a}$ *and* $\mathsf{O_c}$ *be disjoint and countably infinite sets of* abstract nodes *and* concrete nodes. *A* completion tree *for an* $\mathcal{ALC}(\mathcal{C})$*-concept* C *and a TBox* $\mathcal{T}$ *is a finite, labeled tree* $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ *with nodes* $\mathsf{V_a} \cup \mathsf{V_c}$ *and edges* $E \subseteq (\mathsf{V_a} \times (\mathsf{V_a} \cup \mathsf{V_c}))$ *such that* $\mathsf{V_a} \subseteq \mathsf{O_a}$ *and* $\mathsf{V_c} \subseteq \mathsf{O_c}$. *The tree is labeled as follows:*

1. *each node* $a \in \mathsf{V_a}$ *is labeled with a subset* $\mathcal{L}(a)$ *of* $\mathsf{sub}(C, \mathcal{T})$,
2. *each edge* $(a, b) \in E$ *with* $a, b \in \mathsf{V_a}$ *is labeled with a role name* $\mathcal{L}(a, b)$ *occurring in* C *or* $\mathcal{T}$*;*
3. *each edge* $(a, x) \in E$ *with* $a \in \mathsf{V_a}$ *and* $x \in \mathsf{V_c}$ *is labeled with a concrete feature* $\mathcal{L}(a, x)$ *occurring in* C *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* $x \in \mathsf{V_c}$ *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 an* $\mathcal{ALC}(\mathcal{C})$*-concept* C *and a TBox* $\mathcal{T}$ *is a pair* $S = (T, \mathcal{N})$ *where* $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ *is a completion tree for* C *and* $\mathcal{T}$ *and* $\mathcal{N}$ *is a* Rel*-network with* $V_{\mathcal{N}} = \mathsf{V_c}$.

We now define an operation that is used by the tableau algorithm to add new nodes to completion trees. The operation respects the functionality of abstract and concrete features.

Definition 10 ($\oplus$ Operation). *An abstract or concrete node is called* fresh *in 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:*

- *if* $a \in \mathsf{V_a}$, $b \in \mathsf{O_a}$ *fresh in* T, *and* $R \in \mathsf{N_R}$, *then* $S \oplus aRb$ *yields the 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.
- *if* $a \in \mathsf{V_a}$, $x \in \mathsf{O_c}$ *fresh in* T, *and* $g \in \mathsf{N_{cF}}$, *then* $S \oplus agx$ *yields the 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 w.r.t. T and setting

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

The tableau algorithm works by starting with an initial completion system that is then successively expanded with the goal of constructing a model of the input concept and TBox. To ensure termination, we need a mechanism for detecting cyclic expansions, which is commonly called *blocking*. Informally, we detect nodes in the completion tree that are similar to previously created ones and then block them, i.e., stop further expansion at such nodes. To introduce blocking, we start with some preliminaries. For $a \in \mathsf{V_a}$, we define the set of features of a as

$$\mathsf{feat}(a) := \{\ g \in \mathsf{N_{cF}} \mid a \text{ has a } g\text{-successor }\}.$$

Next, we define the *concrete neighborhood* of a as the constraint network

$$\mathcal{N}(a) \ := \ \{\ (x\ r\ y) \mid \begin{array}{l}\text{there exist } g, g' \in \mathsf{feat}(a) \text{ s.t. } x \text{ is a } g\text{-succ.}\\ \text{of } a,\ y \text{ is a } g'\text{-succ. of } a, \text{ and } (x\ r\ y) \in \mathcal{N}\ \}\end{array}$$

Finally, if $a, b \in \mathsf{V_a}$ and $\mathsf{feat}(a) = \mathsf{feat}(b)$, we write $\mathcal{N}(a) \sim \mathcal{N}(b)$ to express that $\mathcal{N}(a)$ and $\mathcal{N}(b)$ are isomorphic, i.e., that the mapping $\pi : V_{\mathcal{N}(a)} \to V_{\mathcal{N}(b)}$ defined by mapping the g-successor of a to the g-successor of b for all $g \in \mathsf{feat}(a)$ is an isomorphism.

If T is a completion tree and a and b are abstract nodes in T, then we say that a is an *ancestor* of b if b is reachable from a in the tree T.

Definition 11 (Blocking). *Let $S = (T, \mathcal{N})$ be a completion system for a concept C_0 and a TBox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, and let $a, b \in \mathsf{V_a}$. We say that $a \in \mathsf{V_a}$ is* potentially blocked by *b if the following holds:*

1. *b is an ancestor of a in T,*
2. *$\mathcal{L}(a) \subseteq \mathcal{L}(b)$,*
3. *$\mathsf{feat}(a) = \mathsf{feat}(b)$.*

We say that a is directly blocked *by b if the following holds:*

1. *a is potentially blocked by b,*
2. *$\mathcal{N}(a)$ and $\mathcal{N}(b)$ are complete, and*
3. *$\mathcal{N}(a) \sim \mathcal{N}(b)$.*

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

$R\sqcap$	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 b 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 $\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$, 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 $(x_1\ r_i\ x_2) \in \mathcal{N}$ for some i with $1 \leq i \leq k$ then set $S := S \oplus aU_1x_1 \oplus aU_2x_2$ with $x_1, x_2 \in \mathsf{O_c}$ fresh and $\mathcal{N} := \mathcal{N} \cup \{(x_1\ r_i\ x_2)\}$ for some i with $1 \leq i \leq k$
$R\forall_c$	if $\forall U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$, 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 $(x_1\ r_i\ x_2) \notin \mathcal{N}$ for all i with $1 \leq i \leq k$ then set $\mathcal{N} := \mathcal{N} \cup \{(x_1\ r_i\ x_2)\}$ for some i with $1 \leq i \leq k$
Rnet	if a is potentially blocked by b or vice versa and $\mathcal{N}(a)$ is not complete then non-deterministically guess a completion $\mathcal{N}'$ of $\mathcal{N}(a)$ and set $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$
Rtbox	if $C_\mathcal{T} \notin \mathcal{L}(a)$ then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_\mathcal{T}\}$

Figure 4. The completion rules.

4.3. The Tableau Algorithm

To decide the satisfiability of an $\mathcal{ALC}(\mathcal{C})$-concept C_0 w.r.t. a TBox $\mathcal{T}$, the tableau algorithm is started with the initial completion system $S_{C_0} = (T_{C_0}, \emptyset)$, where the initial completion tree T_{C_0} is defined by setting

$$T_{C_0} := (\{a_0\}, \emptyset, \emptyset, \{a_0 \mapsto \{C_0\}\}).$$

The algorithm then repeatedly applies the completion rules given in Figure 4. In the formulation of Rnet, a *completion* of a Rel-network N is a satisfiable and complete Rel-network N' such that $V_N = V_{N'}$ and $N \subseteq N'$. Later on, we will argue that the completion to be guessed always exists.

As has already been noted above, rule application can be understood as the step-wise construction of a model of C_0 and $\mathcal{T}$. Among the rules, there are four non-deterministic ones: $R\sqcup$, $R\exists_c$, $R\forall_c$, and $R\mathsf{net}$.[3] Rules are applied until an obvious inconsistency (as defined below) is detected or the completion system becomes *complete*, i.e., no more rules are applicable. The algorithm returns "satisfiable" if there is a way to apply the rules such that a complete completion system is found that does not contain a contradiction. Otherwise, it returns "unsatisfiable".

All rules except $R\mathsf{net}$ are rather standard, see for example [2, 21].[4] The purpose of $R\mathsf{net}$ is to resolve a potential blocking situation between two nodes a and b into either an actual blocking situation or a non-blocking situation. This is achieved by completing the networks $\mathcal{N}(a)$ and $\mathcal{N}(b)$. For ensuring termination, an appropriate interplay between this rule and the blocking condition is crucial. Namely, we have to apply $R\mathsf{net}$ with highest precedence. It can be seen that the blocking mechanism obtained in this way is a refinement of pairwise blocking as known from [18]. In particular, the conditions $\mathcal{L}(a) \subseteq \mathcal{L}(b)$ and $\mathsf{feat}(a) = \mathsf{feat}(b)$ are implied by the standard definition of pairwise blocking due to path normal form.

We now define what we mean by an obvious inconsistency. As soon as such an inconsistency is encountered, the tableau algorithm returns "unsatisfiable".

Definition 12 (Clash). *Let $S = (T, \mathcal{N})$ be a completion system for a concept C and a TBox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. S contains a* clash *if one of the following conditions holds:*

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

If S does not contain a clash, S is called clash-free.

We present the tableau algorithm in pseudo-code notation in Figure 5. It is started with the initial completion system as argument, i.e., by calling $\mathsf{sat}(S_{C_0})$.

Note that checking for clashes before rule application is crucial for $R\mathsf{net}$ to be well-defined: if $R\mathsf{net}$ is applied to a node a, we must be sure that there indeed exists a completion $\mathcal{N}'$ of $\mathcal{N}(a)$ to be guessed, i.e., a *satisfiable* network $\mathcal{N}'$ such that $V'_{\mathcal{N}} = V_{\mathcal{N}(a)}$ and $\mathcal{N}(a) \subseteq \mathcal{N}'$.

[3] By disallowing disjunctions of relations in the constraint-based concept constructors, $R\exists_c$ and $R\forall_c$ can easily be made deterministic.

[4] Note that our version of the $R\exists$ rule uses the operation $S \oplus aRb$ which initializes the label $\mathcal{L}(b)$, and thus the rule only *adds* C to the already existing label.

```
procedure sat(S)
  if S contains a clash then return unsatisfiable
  if S is complete then return satisfiable
  if Rnet is applicable
    then S' := application of Rnet to S
    else S' := application of any applicable completion rule to S
  return sat(S')
```

Figure 5. The (non-deterministic) algorithm for satisfiability in $\mathcal{ALC}(\mathcal{C})$.

Clash checking before rule application ensures that the network $\mathcal{N}$ is satisfiable when $R\mathsf{net}$ is applied. Clearly, this implies the existence of the required completion.

4.4. Correctness

We prove termination, soundness and completeness of the presented tableau algorithm. In the following, we use $|M|$ to denote the cardinality of a set M. With $\mathsf{N}_\mathsf{C}^{C_0,\mathcal{T}}$, $\mathsf{N}_\mathsf{R}^{C_0,\mathcal{T}}$ and $\mathsf{N}_\mathsf{cF}^{C_0,\mathcal{T}}$, we denote the sets of concept names, role names, and concrete features that occur in the concept C_0 and the TBox $\mathcal{T}$. We use $|C|$ to denote the length of a concept C and $|\mathcal{T}|$ to denote $\sum_{C \sqsubseteq D \in \mathcal{T}} (|C| + |D|)$.

Lemma 3 (Termination). *The tableau algorithm terminates on every input.*

Proof. Let $S_0, S_1, \ldots$ be the sequence of completion systems generated during the run of the tableau algorithm started on input C_0, $\mathcal{T}$, and let $S_i = (T_i, \mathcal{N}_i)$. Set $n := |C_0| + |\mathcal{T}|$. Obviously, we have $|\mathsf{sub}(C_0, \mathcal{T})| \leq n$. We first show the following:

(a) For all $i \geq 0$, the out-degree of T_i is bounded by n.

(b) For $i \geq 0$, the depth of T_i is bounded by $\ell = 2^{2n} \cdot |\mathsf{Rel}|^{n^2} + 2$.

First for (a). Nodes from $\mathsf{V_c}$ do not have successors. Let $a \in \mathsf{V_a}$. Successors of a are created only by applications of the rules $R\exists$ and $R\exists_c$. The rule $R\exists$ generates at most one abstract successor (i.e., element of $\mathsf{V_a}$) of a for each $\exists R.C \in \mathsf{sub}(C_0, \mathcal{T})$, and $R\exists_c$ generates at most two abstract successors of a for every $\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathsf{sub}(C_0, \mathcal{T})$. Moreover, $R\exists_c$ generates at most one concrete successor for every element of $\mathsf{N}_\mathsf{cF}^{C_0,\mathcal{T}}$. It is not difficult to verify that this implies that the number of (abstract and concrete) successors of a is bounded by n.

Now for (b). Assume, to the contrary of what is to be shown, that there is an $i \geq 0$ such that the depth of T_i exceeds $\ell = 2^{2n} \cdot |\mathsf{Rel}|^{n^2} + 2$.

Moreover, let i be smallest with this property. This means that S_i has been obtained from S_{i-1} by applying one of the rules $R\exists$ and $R\exists_c$ to a node on level ℓ, or by applying $R\exists_c$ to a node on level $\ell - 1$.

Let $T_{i-1} = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. Since T_i is obtained from T_{i-1} by application of $R\exists$ or $R\exists_c$ and since $R\mathsf{net}$ is applied with highest precedence, $R\mathsf{net}$ is not applicable to T_{i-1}. This means that, for every $a, b \in \mathsf{V_a}$ such that b is potentially blocked by a, $\mathcal{N}_{i-1}(a)$ and $\mathcal{N}_{i-1}(b)$ are complete. Let us define a binary relation $\approx$ on $\mathsf{V_a}$ as follows:

$$a \approx b \quad \text{iff} \quad \mathcal{L}(a) = \mathcal{L}(b),\ \mathsf{feat}(a) = \mathsf{feat}(b),\ \text{and } \mathcal{N}_{i-1}(a) \sim \mathcal{N}_{i-1}(b).$$

Obviously, $\approx$ is an equivalence relation on $\mathsf{V_a}$. The definition of blocking implies that if a is an ancestor of b and $a \approx b$, then b is blocked by a in S_{i-1}. Let $\mathsf{V_a}/_{\approx}$ denote the set of $\approx$-equivalence classes and set $m := |\mathsf{N}_{\mathsf{cF}}^{C_0,\mathcal{T}}|$. Since $\mathcal{L}(a) \subseteq \mathsf{sub}(C_0, \mathcal{T})$, and $\mathcal{N}_{i-1}(a)$ is a complete Rel-network with $|V_{\mathcal{N}_{i-1}(a)}| \leq m$ for all $a \in \mathsf{V_a}$, it is not difficult to verify that

$$|\mathsf{V_a}/_{\approx}| \leq 2^{|\mathsf{sub}(C_0,\mathcal{T})|} \sum_{i=0}^{m} \binom{m}{i} |\mathsf{Rel}|^{i^2}$$

Since $m \leq n$, we obtain $|\mathsf{V_a}/_{\approx}| \leq 2^n \cdot 2^n \cdot |\mathsf{Rel}|^{n^2} = 2^{2n} \cdot |\mathsf{Rel}|^{n^2}$. Let $a \in \mathsf{V_a}$ be the node to which a rule is applied in T_{i-1} to obtain T_i. As already noted, the level k of a in T_{i-1} is at least $\ell - 1 \geq |\mathsf{V_a}/_{\approx}| + 1$. Let $a_0, \ldots, a_k$ be the path in T_{i-1} leading from the root to a. Since $k > |\mathsf{V_a}/_{\approx}|$, we have $a_i \approx a_j$ for some i, j with $0 \leq i < j \leq k$. This means that a is blocked and contradicts the assumption that a completion rule was applied to a. Thus, the proof of (b) is finished.

The tableau algorithm terminates due to the following reasons:

1. It constructs a finitely labeled completion tree T of bounded outdegree and depth (by (a) and (b)) in a monotonic way, i.e., no nodes are removed from T and no concepts are removed from node labels. Also, no constraints are removed from the constraint system $\mathcal{N}$;

2. every rule application adds new nodes or node labels to T, or new constraints to $\mathcal{N}$;

3. the cardinality of node labels is bounded by $|\mathsf{sub}(C_0, \mathcal{T})|$ and the number of constraints in $\mathcal{N}$ is bounded by $|\mathsf{Rel}| \cdot k^2$, with k the (bounded) number of concrete nodes.

□

Lemma 4 (Soundness). *If the tableau algorithm returns* satisfiable*, then the input concept C_0 is satisfiable w.r.t. the input TBox $\mathcal{T}$.*

Proof. If the tableau algorithm returns $\mathsf{satisfiable}$, then there exists a complete and clash-free completion system $S = (T, \mathcal{N})$ for C_0 and $\mathcal{T}$. Our aim is to use S for defining a model $\mathcal{I}$ for C_0 and $\mathcal{T}$. We start with a brief outline of the proof.

To obtain the desired model $\mathcal{I}$, the completition tree T is unravelled to another (possibly infinite) tree by replacing directly blocked nodes with nodes that block them. The second condition of "potentially blocked" ensures that by doing this, we do not violate any existential or universal conditions in the predecessor of a directly blocked node. This yields only the abstract part of $\mathcal{I}$. Defining the concrete part is less straightforward. To start with, the described unravelling process can be seen as follows. We start with the tree T where all indirectly blocked nodes are dropped, and then repeatedly patch subtrees of T to the existing tree. More precisely, such a patched subtree is rooted by a node that blocks the node onto which the root of the subtree is patched. The third condition of "directly blocked" ensures that the networks $\mathcal{N}(a)$ and $\mathcal{N}(b)$ (which comprise only the concrete successors a and b) are complete and identical if a is blocked by b. This means that we can obtain a (possibly infinite) constraint network $\mathbf{N}$ that corresponds to the unravelled tree by patching together fragments of $\mathcal{N}$ which coincide on overlapping parts. Since $\mathcal{N}$ is satisfiable, patchwork and compactness property ensure that the network $\mathbf{N}$ is satisfiable as well and thus we can use a model of $\mathbf{N}$ to define the concrete part of the model $\mathcal{I}$.

Formally, we proceed in several steps. Let $S = (T, \mathcal{N})$ be as above, $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, and let $\mathsf{root} \in \mathsf{V_a}$ denote the root of T. 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 is easily seen that, by definition of blocking, such 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 $b_1, \ldots, b_n$ nodes from $\mathsf{V_a}$, such that, for $1 \leq i < n$, one of the following holds:

1. a_{i+1} is a successor of a_i in T, a_{i+1} is unblocked, and $b_{i+1} = a_{i+1}$;
2. b_{i+1} is a successor of a_i in T and $a_{i+1} = \mathsf{blocks}(b_{i+1})$.

Intuitively, a path $\frac{a_1}{b_1}, \ldots, \frac{a_n}{b_n}$ represents the sequence of nodes $a_1, \ldots, a_n$, and the b_i provide justification for the existence of the path in case of blocking situations. Observe that b_{i+1} is always a successor of a_i. We use Paths to denote the set of all paths in S including the empty path. For $p \in \mathsf{Paths}$ nonempty, $\mathsf{tail}(p)$ denotes the last pair of p. We now define the "abstract part" of the model $\mathcal{I}$ we are constructing:

$$\Delta_{\mathcal{I}} := \{p \in \mathsf{Paths} \mid p \text{ non-empty and first pair is } \frac{\mathsf{root}}{\mathsf{root}}\}$$

$$A^{\mathcal{I}} := \{p \in \Delta_{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a}{b} \text{ and } A \in \mathcal{L}(a)\},\ A \in \mathsf{N}_{\mathsf{C}}^{C_0,\mathcal{T}}$$

$$R^{\mathcal{I}} := \{(p, p \cdot \frac{a}{b}) \in \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a'}{b'} \text{ and } b \text{ is } R\text{-successor of } a' \text{ in } T\ \},\ R \in \mathsf{N}_{\mathsf{R}}^{C_0,\mathcal{T}}$$

Observe that

(i) $\Delta_{\mathcal{I}}$ is non-empty, since $\frac{\mathsf{root}}{\mathsf{root}} \in \Delta_{\mathcal{I}}$.

(ii) $f^{\mathcal{I}}$ is functional for every $f \in \mathsf{N}_{\mathsf{aF}}$: this is ensured by the "$\oplus$" operation which generates at most one f-successor per abstract node, and by the definition of Paths in which we choose only a single blocking node to be put into a path.

Intuitively, the abstract part of $\mathcal{I}$ as defined above is obtained by "patching together" parts of the completion tree T. For defining the concrete part of $\mathcal{I}$, we make this patching explicit: For $p \in \Delta_{\mathcal{I}}$, 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 is blocked by a). We use Hooks to denote the set of all hooks. 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, for $p \in \Delta_{\mathcal{I}}$ and $q \in \mathsf{Hooks}$, we call p a *q-companion* if 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 $\mathsf{tail}(q')$. Then, the part of $\mathcal{I}$ patched at 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 p and 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.

To define the concrete part of $\mathcal{I}$, we need to establish some additional notions. Since S is clash-free, $\mathcal{N}$ is satisfiable. It is an easy exercise to show that then there exists a completion of $\mathcal{N}$. We fix such a completion $\mathcal{N}^c$ with the nodes renamed as follows: each concrete node x that is a g-successor of an abstract node a is renamed to the pair (a, g). This naming scheme is well-defined since the "$\oplus$" operation ensures that every abstract node a has at most one g-successor, for every $g \in \mathsf{N}_{\mathsf{cF}}$. We now define a network $\mathbf{N}$ which, intuitively, describes the constraints put on the concrete part of the model. If $q \in \mathsf{Hooks}$, $p \in P(q)$, and $\mathsf{tail}(p) = \frac{a}{b}$, we set

$$\mathsf{rep}_q(p) := \begin{cases} b & \text{if } p \neq q \text{ and } a \neq b \\ a & \text{otherwise} \end{cases}$$

Intuitively, this notion is needed for the following reason: let $p, q \in \mathsf{Hooks}$ with q a successor of p. Then $\mathsf{tail}(q) = \frac{a}{b}$ with b blocked by a, $q \in P(p)$, and $q \in P(q)$. As part of $P(p)$, q represents the blocked node b. As part of $P(q)$, q represents the blocking node a. This overlapping of patched parts at hooks is made explicit via the notion $\mathsf{rep}_q(p)$. Now define $\mathbf{N}$ as follows:

$$\mathbf{N} := \{((p,g)\, r\, (p',g')) \mid \text{there is a } q \in \mathsf{Hooks} \text{ such that } p, p' \in P(q) \text{ and } ((\mathsf{rep}_q(p), g)\, r\, (\mathsf{rep}_q(p'), g')) \in \mathcal{N}^c\}$$

Our next aim is to show that $\mathbf{N}$ is satisfiable. To this end, we first show that $\mathbf{N}$ is patched together from smaller networks: every hook p gives rise to a part of $\mathbf{N}$ as follows:

$$N(p) := \mathbf{N}|_{\{(q,g) \in V_{\mathbf{N}} \mid q \in P(p)\}},$$

i.e, $N(p)$ is the restriction of $\mathbf{N}$ to those variables (q, g) such that q is a p-companion.

The following claim shows that $\mathbf{N}$ is patched together from the networks $N(p)$, $p \in \mathsf{Hooks}$.

Claim 1. The following holds:

(a) $\mathbf{N} = \bigcup_{p \in \mathsf{Hooks}} N(p)$.

(b) if $p, q \in \mathsf{Hooks}$, $p \neq q$, q is not a successor of p, and p is not a successor of q, then $V_{N(p)} \cap V_{N(q)} = \emptyset$;

(c) if $p, q \in \mathsf{Hooks}$ and q is a successor of p, then $N(p)|_{V_{N(p)} \cap V_{N(q)}} = N(q)|_{V_{N(p)} \cap V_{N(q)}}$;

Proof. (a) As $\mathbf{N} \supseteq \bigcup_{p \in \mathsf{Hooks}} N(p)$ is immediate by definition of $N(p)$, it remains to show $\mathbf{N} \subseteq \bigcup_{p \in \mathsf{Hooks}} N(p)$. Thus, let $((p,g)\, r\, (p',g')) \in \mathbf{N}$. Then there is a $q \in \mathsf{Hooks}$ such that $p, p' \in P(q)$. By definition of $N(q)$, this implies $((p,g)\, r\, (p',g')) \in N(q)$.

(b) We show the contrapositive. Let $(q^*, g) \in V_{N(p)} \cap V_{N(q)}$. It follows that $q^* \in P(p) \cap P(q)$, i.e., there are $q', q'' \in \mathsf{Paths}$ such that (i) $q^* = pq'$, $q^* = qq''$, and (ii) all nodes $\frac{a}{b}$ in q', q'' satisfy $a = b$, with the possible exception of the last one. Due to (i), $p = q$, p is a prefix of q, or vice versa. In the first case, we are done. In the second case, since $q \in \mathsf{Hooks}$ we have that $\mathsf{tail}(q) = \frac{a}{b}$ for some a, b with $a \neq b$. Together with $q^* = pq'$, (ii), and since p is a prefix of q is a prefix of q^*, this implies that $q = q^*$. Thus $q = pq'$. Again by (ii), we have that q is a successor of p. The third case is analogous to the second.

(c) By definition of $N(p)$ and $N(q)$, we have $N(p)|_{V_{N(p)} \cap V_{N(q)}} = \mathbf{N}|_{V_{N(p)} \cap V_{N(q)}} = N(q)|_{V_{N(p)} \cap V_{N(q)}}$ for all $p, q \in \mathsf{Hooks}$.

Claim 1 shows that $\mathbf{N}$ is patched together from smaller networks. Our aim is to apply the patchwork and compactness property to derive satisfiability of $\mathbf{N}$. For being able to do this, we additionally need to know that the smaller networks are complete and satisfiable, and that they agree on overlapping parts. Before we prove this, we establish some crucial properties.

(P1) If $q, q' \in \mathsf{Hooks}$ with q' successor of q, then $V_{P(q)} \cap V_{P(q')} = \{q'\}$.

(P2) If $((q,g)\, r\, (q',g')) \in N(p)$ then $((\mathsf{rep}_p(q),g)\, r\, (\mathsf{rep}_p(q'),g')) \in \mathcal{N}^c$.

(P1) is obvious by definition of hooks and q-companions. For (P2), let $((q,g)\, r\, (q',g')) \in N(p)$. Then $q, q' \in P(p)$. Since $N(p) \subseteq \mathbf{N}$, there is a $p' \in \mathsf{Hooks}$ such that $q, q' \in P(p')$ and

$(*)$ $((\mathsf{rep}_{p'}(q),g)\, r\, (\mathsf{rep}_{p'}(q'),g')) \in \mathcal{N}^c$.

If $p = p'$, we are done. Thus, let $p \neq p'$. By Claim 1(b) and (P1), $q, q' \in P(p) \cap P(p')$ implies that $q = q' = p$ and p is a successor-hook of p', or $q = q' = p'$ and p' is a successor-hook of p. W.l.o.g., assume that the former is the case. Let $\mathsf{tail}(q) = \frac{a}{b}$. Since $q = p$ and p is a hook, we have $a \neq b$, and thus b is blocked by a in T. By definition of rep, we have $\mathsf{rep}_{p'}(q) = b$ and $\mathsf{rep}_p(q) = a$. Thus, $(*)$ yields $((b,g)\, r\, (b,g')) \in \mathcal{N}^c$. Since b is blocked by a, the blocking condition yields $((a,g)\, r\, (a,g')) \in \mathcal{N}^c$ and we are done. This finshes the proof of Claim 1.

Claim 2. For every $p \in \mathsf{Hooks}$, $N(p)$ is finite, complete, and satisfiable.

Proof. Let $p \in \mathsf{Hooks}$. Since the completion tree T is finite, so are $P(p)$ and $N(p)$. Next, we show that $N(p)$ is complete. This involves two subtasks: showing that (i) for all $(q,g), (q',g') \in V_{N(p)}$, there is at least one relation r with $((q,g)\, r\, (q',g')) \in N(p)$; and (ii) there is at most one such relation.

For (i), let $(q,g), (q',g') \in V_{N(p)}$. By (P2), we obtain that $(\mathsf{rep}_p(q),g)$, $(\mathsf{rep}_p(q'),g') \in V_{\mathcal{N}^c}$. Since $\mathcal{N}^c$ is complete, there is an r such that $((\mathsf{rep}_p(q),g)\, r\, (\mathsf{rep}_p(q'),g')) \in \mathcal{N}^c$. By definition of $\mathbf{N}$ and $N(p)$, we have $((q,g)\, r\, (q',g')) \in N(p)$. For (ii), assume that $((q,g)\, r\, (q',g')) \in N(p)$, for each $r \in \{r_1, r_2\}$. Then, (P2) implies $((\mathsf{rep}_p(q),g)\, r_i\, (\mathsf{rep}_p(q'),g')) \in \mathcal{N}^c$ for each $r \in \{r_1, r_2\}$. Thus, completeness of $\mathcal{N}^c$ implies that $r_1 = r_2$ as required.

Finally, we show satisfiability of $N(p)$. By (P2), $((q,g)\, r\, (q',g')) \in N(p)$ implies $((\mathsf{rep}_p(q),g)\, r\, (\mathsf{rep}_p(q'),g')) \in \mathcal{N}^c$. Thus, satisfiability of $\mathcal{N}^c$, yields satisfiability of $N(p)$.

We are now ready to apply the patchwork and compactness properties.

Claim 3. $\mathbf{N}$ is satisfiable.

Proof. First assume that there are no blocked nodes in S. Then, $\mathsf{Hooks} = \{\frac{\mathsf{root}}{\mathsf{root}}\}$. By Claim 1(a), we have that $\mathbf{N} = N(\frac{\mathsf{root}}{\mathsf{root}})$, and by Claim 2 we obtain that $\mathbf{N}$ is satisfiable. Now assume that there are blocked nodes in S. Since $\mathsf{V_a}$ is finite (c.f. Lemma 3), Hooks is a countably infinite set. Moreover, the "successor" relation on Hooks is easily seen to arrange Hooks in an infinite tree whose out-degree is bounded by the cardinality of $\mathsf{V_a}$. Therefore, we can fix an enumeration $\{p_0, p_1, ...\}$ of Hooks such that:

- $p_0 = \frac{\mathsf{root}}{\mathsf{root}}$,
- if p_i is a successor of p_j, then $i > j$.

By Claim 1(a), we have that $\mathbf{N} = \bigcup_{i\geq 0} N(p_i)$. We first show by induction that, for all $k \geq 0$, the network $\mathbf{N}_k := \bigcup_{0\leq i\leq k} N(p_i)$ is satisfiable.

- $k = 0$: $\mathbf{N}_0 = N(p_0)$ is satisfiable by Claim 2.
- $k > 0$. We have that $\mathbf{N}_k = \mathbf{N}_{k-1} \cup N(p_k)$. By induction, $\mathbf{N}_{k-1}$ is satisfiable. Let $\mathbf{N}^c_{k-1}$ be a completion of $\mathbf{N}_{k-1}$ and let $\mathbf{N}'_k = \mathbf{N}^c_{k-1} \cup N(p_k)$. There exists a unique $p_n \in \mathsf{Hooks}$, $n < k$, such that p_k is a successor of p_n. By definition of $\mathbf{N}_{k-1}$ and Claim 1(b), and since $V_{\mathbf{N}^c_{k-1}} = V_{\mathbf{N}_{k-1}}$, we have that

$$V_{\mathbf{N}^c_{k-1}} \cap V_{N(p_k)} = V_{N(p_n)} \cap V_{N(p_k)}.$$

Moreover, by Claim 2, $N(p_n)$ is complete, and thus

$$\mathbf{N}^c_{k-1}|_{V_{N(p_n)}\cap V_{N(p_k)}} = N(p_n)|_{V_{N(p_n)}\cap V_{N(p_k)}}.$$

Finally, Claim 1(c) yields

$$N(p_n)|_{V_{N(p_n)}\cap V_{N(p_k)}} = N(p_k)|_{V_{N(p_n)}\cap V_{N(p_k)}}.$$

Summing up, we obtain that the intersection parts of $\mathbf{N}^c_{k-1}$ and $N(p_k)$ are identical:

$$\begin{aligned}
\mathbf{N}^c_{k-1}|_{V_{\mathbf{N}^c_{k-1}}\cap V_{N(p_k)}} &= \mathbf{N}^c_{k-1}|_{V_{N(p_n)}\cap V_{N(p_k)}} \\
&= N(p_n)|_{V_{N(p_n)}\cap V_{N(p_k)}} \\
&= N(p_k)|_{V_{N(p_n)}\cap V_{N(p_k)}} \\
&= N(p_k)|_{V_{\mathbf{N}^c_{k-1}}\cap V_{N(p_k)}}
\end{aligned}$$

By Claim 2, we have that $N(p_k)$ is finite, complete and satisfiable. The same holds for $\mathbf{N}^c_{k-1}$. Thus, the patchwork property of $\mathcal{C}$ yields that $\mathbf{N}'_k$ is satisfiable. Since $\mathbf{N}_k \subseteq \mathbf{N}'_k$, $\mathbf{N}_k$ is satisfiable.

Now, satisfiability of the networks $\mathbf{N}_k$, $k \geq 0$, and the compactness property of $\mathcal{C}$ imply satisfiability of $\mathbf{N}$. This finishes the proof of Claim 3.

We are now ready to define the concrete part of the model $\mathcal{I}$. Since $\mathbf{N}$ is satisfiable, there is an $M_\mathcal{I} \in \mathfrak{M}$ and a mapping $\tau : V_\mathbf{N} \to V_{M_\mathcal{I}}$ such that $(x\ r\ y) \in \mathbf{N}$ implies $(\tau(x)\ r\ \tau(y)) \in V_{M_\mathcal{I}}$. Define $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I}, M_\mathcal{I})$ with $\Delta_\mathcal{I}$ and $\cdot^\mathcal{I}$ defined as above, and, additionally:

$$g^\mathcal{I} := \{(p, \tau(p,g)) \in \Delta_\mathcal{I} \times V_{M_\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a}{b} \text{ and } g \in \mathsf{feat}(a)\},\ g \in \mathsf{N}_\mathsf{cF}^{C_0,\mathcal{T}}$$

Note that, by definition, $g^\mathcal{I}$ is functional for every $g \in \mathsf{N}_\mathsf{cF}$. In order to show that $\mathcal{I}$ is a model of C_0 and $\mathcal{T}$, we require one more claim:

Claim 4. For all $s \in \Delta_\mathcal{I}$ and $C \in \mathsf{sub}(C_0, \mathcal{T})$, if $\mathsf{tail}(s) = \frac{a}{b}$ and $C \in \mathcal{L}(a)$, then $s \in C^\mathcal{I}$.

Proof. We prove the claim by structural induction on C. Let $s \in \Delta_\mathcal{I}$, $\mathsf{tail}(s) = \frac{a}{b}$, and $C \in \mathcal{L}(a)$. In the following, we will implicitly use the fact that, by construction of Paths, a is not blocked in S. We make a case distinction according to the topmost operator in C:

1. C is a concept name. By construction of $\mathcal{I}$, we have $s \in C^\mathcal{I}$.
2. $C = \neg D$. Since C is in NNF, D is a concept name. Clash-freeness of S implies $D \notin \mathcal{L}(a)$. The construction of $\mathcal{I}$ implies $s \notin D^\mathcal{I}$ which yields $s \in (\neg D)^\mathcal{I}$.
3. $C = D \sqcap E$. The completeness of S implies $\{D, E\} \subseteq \mathcal{L}(a)$. The induction hypothesis yields $s \in D^\mathcal{I}$ and $s \in E^\mathcal{I}$, therefore $s \in (D \sqcap E)^\mathcal{I}$.
4. $C = D \sqcup E$. The completeness of S implies $\{D, E\} \cap \mathcal{L}(a) \neq \emptyset$. By induction hypothesis it holds that $s \in D^\mathcal{I}$ or $s \in E^\mathcal{I}$, and therefore $s \in (D \sqcup E)^\mathcal{I}$.
5. $C = \exists R.D$. Since the $R\exists$ rule is not applicable, a has an R-successor c such that $D \in \mathcal{L}(c)$. By definition of $\mathcal{I}$, there is a $t = s \cdot \frac{d}{c} \in \Delta_\mathcal{I}$ such that either $c = d$ or c is blocked by d in S. Since $\mathcal{L}(c) \subseteq \mathcal{L}(d)$ in both cases, we have that $D \in \mathcal{L}(d)$. By induction, it holds that $t \in D^\mathcal{I}$. By definition of $\mathcal{I}$, we have $(s,t) \in R^\mathcal{I}$ and this implies $s \in C^\mathcal{I}$.
6. $C = \forall R.D$. Let $(s,t) \in R^\mathcal{I}$. By construction of $\mathcal{I}$, $t = s \cdot \frac{d}{c}$ such that c is an R-successor of a. Since $R\forall$ is not applicable, we have that $D \in \mathcal{L}(c)$. Since $\mathcal{L}(c) \subseteq \mathcal{L}(d)$ (as in the previous case), we have $C \in \mathcal{L}(d)$, and by induction $t \in C^\mathcal{I}$. Since this holds independently of the choice of t, we obtain $s \in C^\mathcal{I}$.

7. $C = \exists U_1, U_2.(r_1 \vee \cdots \vee r_k)$. Since C is in PNF, U_i is either a concrete feature or of the form Rg, for each $i \in \{1, 2\}$. We consider only the case $U_1 = R_1 g_1$, $U_2 = R_2 g_2$, as the remaining cases are similar but easier. Since the $R\exists_c$ rule is not applicable, there exists an R_j-successor c_j of a and a g_j-successor y_j of c_j for $j = 1, 2$ such that $(y_1\ r_i\ y_2) \in \mathcal{N}$ for some $1 \leq i \leq k$. Then $((c_1, g_1)\ r_i\ (c_2, g_2)) \in \mathcal{N}^c$. Moreover, there is a $t_j = s \cdot \frac{d_j}{c_j} \in \Delta_\mathcal{I}$ such that $c_j = d_j$ or c_j is blocked by d_j, $j = 1, 2$. By definition of $R_j^\mathcal{I}$, we have that $(s, t_j) \in R_j^\mathcal{I}$, $j = 1, 2$. Moreover, since a is not blocked and c_1 and c_2 are its successors, there is a $p \in \mathsf{Hooks}$ such that t_1 and t_2 are p-companions and $\mathsf{rep}_p(t_1) = c_1$, $\mathsf{rep}_p(t_2) = c_2$. Thus, by definition of $\mathbf{N}$ we obtain $((t_1, g_1)\ r_i\ (t_2, g_2)) \in \mathbf{N}$, implying $(\tau(t_1, g_1)\ r_i\ \tau(t_2, g_2)) \in M_\mathcal{I}$. Since $g_1^\mathcal{I}(t_1) = \tau(t_1, g_1)$ and $g_2^\mathcal{I}(t_2) = \tau(t_2, g_2)$, we obtain that $s \in C^\mathcal{I}$.

8. $C = \forall U_1, U_2.(r_1 \vee \cdots \vee r_k)$. As in the previous case, we will assume that U_1 and U_2 are of the form $U_1 = Rg_1$, $U_2 = R_2 g_2$. Let t_1, t_2 be such that $(s, t_j) \in R_j^\mathcal{I}$ and $g_j^\mathcal{I}(t_j)$ is defined, $j = 1, 2$. By definition of $\mathcal{I}$, we have that $t_j = s \cdot \frac{d_j}{c_j} \in \Delta_\mathcal{I}$ such that c_j is an R_j-successor of a, $j = 1, 2$. Moreover, there is a g_j-successor y_j of c_j for $j = 1, 2$. Since $R\forall_c$ is inapplicable, $\forall U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$ implies that $(y_1\ r_i\ y_2) \in \mathcal{N}$ for some $1 \leq i \leq k$. Thus, $((c_1, g_1)\ r\ (c_2, g_2)) \in \mathcal{N}^c$. Moreover, since a is unblocked there is a $p \in \mathsf{Hooks}$ such that t_1 and t_2 are p-companions and $\mathsf{rep}_p(t_1) = c_1$, $\mathsf{rep}_p(t_2) = c_2$. Thus, by definition of $\mathbf{N}$, we have that $((t_1, g_1)\ r_i\ (t_2, g_2)) \in \mathbf{N}$, which implies $(\tau(t_1, g_1)\ r_i\ \tau(t_2, g_2)) \in M_\mathcal{I}$. Thus, $s \in C^\mathcal{I}$.

This finishes the proof of Claim 4.

Since $C_0 \in \mathcal{L}(\mathsf{root})$ and $\frac{\mathsf{root}}{\mathsf{root}} \in \Delta_\mathcal{I}$, Claim 4 implies that $\mathcal{I}$ is a model of C_0. Finally, let us show that $\mathcal{I}$ is a model of the input TBox $\mathcal{T} = \{\top \sqsubseteq C_\mathcal{T}\}$. Choose an $s \in \Delta_\mathcal{I}$. Let $\mathsf{tail}(s) = \frac{a}{b}$. Since S is complete, $R\mathsf{tbox}$ is not applicable, and thus $C_\mathcal{T} \in \mathcal{L}(a)$. By Claim 4 we have that $s \in C_\mathcal{T}^\mathcal{I}$. Since this holds independtly of the choice of s, we have $C^\mathcal{I} = \Delta_\mathcal{I}$ as required. □

Lemma 5 (Completeness). *If the input concept C_0 is satisfiable w.r.t. the input TBox $\mathcal{T}$, then the algorithm returns* satisfiable.

Proof. Let C_0 be satisfiable w.r.t. $\mathcal{T}$, $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I}, M_\mathcal{I})$ a common model of C_0 and $\mathcal{T}$, and $a_0 \in \Delta_\mathcal{I}$ such that $a_0 \in C_0^\mathcal{I}$. We use $\mathcal{I}$ to guide (the non-deterministic parts of) the algorithm such that it constructs a complete and clash-free completion system. A completion system $S = (T, \mathcal{N})$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ is called *$\mathcal{I}$-compatible* if there exist

mappings $\pi : \mathsf{V_a} \to \Delta_{\mathcal{I}}$ and $\tau : \mathsf{V_c} \to V_{M_{\mathcal{I}}}$ (i.e., to the variables used in $M_{\mathcal{I}}$) such that

(Ca) $C \in \mathcal{L}(a) \Rightarrow \pi(a) \in C^{\mathcal{I}}$

(Cb) b is an R-successor of $a \Rightarrow (\pi(a), \pi(b)) \in R^{\mathcal{I}}$

(Cc) x is a g-successor of $a \ \Rightarrow g^{\mathcal{I}}(\pi(a)) = \tau(x)$

(Cd) $(x\, r\, y) \in \mathcal{N} \Rightarrow (\tau(x)\, r\, \tau(y)) \in M_{\mathcal{I}}$

We first show the following.

Claim 1: If a completion system S is $\mathcal{I}$-compatible and a rule R is applicable to S, then R can be applied such that an $\mathcal{I}$-compatible completion system S' is obtained.

Proof. Let $S = (T, \mathcal{N})$ be an $\mathcal{I}$-compatible completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, let π and τ be functions satisfying (Ca) to (Cd), and let R be a completion rule applicable to S. We make a case distinction according to the type of R.

$R\sqcap$ The rule is applied to a concept $C_1 \sqcap C_2 \in \mathcal{L}(a)$. By (Ca), $C_1 \sqcap C_2 \in \mathcal{L}(a)$ implies $\pi(a) \in (C_1 \sqcap C_2)^{\mathcal{I}}$ and hence $\pi(a) \in C_1^{\mathcal{I}}$ and $\pi(a) \in C_2^{\mathcal{I}}$. Since the rule adds C_1 and C_2 to $\mathcal{L}(a)$, it yields a completion system that is $\mathcal{I}$-compatible via π and τ.

$R\sqcup$ The rule is applied to $C_1 \sqcup C_2 \in \mathcal{L}(a)$. $C_1 \sqcup C_2 \in \mathcal{L}(a)$ implies $\pi(a) \in C_1^{\mathcal{I}}$ or $\pi(a) \in C_2^{\mathcal{I}}$. Since the rule adds either C_1 or C_2 to $\mathcal{L}(a)$, it can be applied such that it yields a completion system that is $\mathcal{I}$-compatible via π and τ.

$R\exists$ The rule is applied to $\exists R.C \in \mathcal{L}(a)$. By (Ca), $\pi(a) \in (\exists R.C)^{\mathcal{I}}$ and hence there exists a $d \in \Delta_{\mathcal{I}}$ such that $(\pi(a), d) \in R^{\mathcal{I}}$ and $d \in C^{\mathcal{I}}$. By definition of $R\exists$ and the "$\oplus$" operation, rule application either (i) adds a new R-successor b of a and sets $\mathcal{L}(b) = \{C\}$; or (ii) reuses an existing R-successor, renames it to b in T and sets $\mathcal{L}(b) = \mathcal{L}(b) \cup \{C\}$. Extend π by setting $\pi(b) = d$. The resulting completion system is $\mathcal{I}$-compatible via the extended π and the original τ.

$R\forall$ The rule is applied to $\forall R.C \in \mathcal{L}(a)$ and it adds C to the label $\mathcal{L}(b)$ of an existing R-successor of a. By (Ca), $\pi(a) \in (\forall R.C)^{\mathcal{I}}$ and by (Cb), $(\pi(a), \pi(b)) \in R^{\mathcal{I}}$. Therefore, $\pi(b) \in C^{\mathcal{I}}$ and the resulting completion system is $\mathcal{I}$-compatible via π and τ.

$R\exists_c$ The rule is applied to a concept $\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$. We assume that $U_1 = R_1 g_1$ and $U_2 = R_2 g_2$. The case where one

or both of U_1, U_2 are only concrete features is similar, but easier. The rule application generates new abstract nodes b_1 and b_2 and concrete nodes x_1 and x_2 (or re-uses existing ones and renames them) such that

- b_j is an R_j-successor of a and
- x_j is a g_j-successor of b_j for $j = 1, 2$.

By (Ca), we have $\pi(a) \in (\exists U_1, U_2.(r_1 \vee \cdots \vee r_k))^{\mathcal{I}}$. Thus, there exist $d_1, d_2 \in \Delta_{\mathcal{I}}$, $v_1, v_2 \in V_{M_{\mathcal{I}}}$ and an i with $1 \leq i \leq k$ such that

- $(\pi(a), d_j) \in R_j^{\mathcal{I}}$,
- $g_j^{\mathcal{I}}(d_j) = v_j$ for $j = 1, 2$, and
- $(v_1\, r_i\, v_2) \in M_{\mathcal{I}}$.

Thus, the rule can be guided such that it adds $(x_1\, r_i\, x_2)$ to $\mathcal{N}$. Extend π by setting $\pi(b_j) := d_j$, and extend τ by setting $\tau(x_j) := v_j$ for $j = 1, 2$. It is easily seen that the resulting completion system is $\mathcal{I}$-compatible via the extended π and τ.

$R\forall_c$ The rule is applied to an abstract node a with $\forall U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$ such that there are $x_1, x_2 \in \mathsf{V_c}$ with x_i a U_i-successor of a, for $i = 1, 2$. By (Ca), $\pi(a) \in (\forall U_1, U_2.(r_1 \vee \cdots \vee r_k))^{\mathcal{I}}$. By (Cb) and (Cc), we have $(\pi(a), \tau(x_1)) \in U_1^{\mathcal{I}}$ and $(\pi(a), \tau(x_2)) \in U_2^{\mathcal{I}}$. By the semantics, it follows that there is an i with $1 \leq i \leq k$ such that $(\tau(x_1)\, r_i\, \tau(x_2)) \in M_{\mathcal{I}}$. The application rule can be guided such that it adds $(x_1\ r_i\ x_2)$ to $\mathcal{N}$. Thus, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

Rnet The rule is applied to an abstract node a such that a is potentially blocked by an abstract node b and $\mathcal{N}(a)$ is not complete (the symmetric case is analogous). The rule application guesses a completion $\mathcal{N}'$ of $\mathcal{N}(a)$, and sets $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$. Define

$$\mathcal{N}' := \{(x\, r\, y) \mid x \text{ is a } g\text{-successor of } a, \\ y \text{ is a } g'\text{-successor of } a, \text{ and } (\tau(x)\, r\, \tau(y)) \in M_{\mathcal{I}}\}.$$

By definition of $\mathcal{N}(a)$, we have $V_{\mathcal{N}(a)} = V_{\mathcal{N}'}$. By (Cd), we have $\mathcal{N}(a) \subseteq \mathcal{N}'$. Since $M_{\mathcal{I}}$ is complete, $\mathcal{N}'$ is complete. Finally, τ witnesses that $M_{\mathcal{I}}$ is a model of $\mathcal{N}'$, and thus $\mathcal{N}'$ is satisfiable. It follows that $\mathcal{N}'$ is a completion of $\mathcal{N}(a)$. Apply Rnet such that $\mathcal{N}'$ is guessed. Then, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

*R*tbox The rule application adds $C_{\mathcal{T}}$ to $\mathcal{L}(a)$, for some $a \in \mathsf{V_a}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$, we have $\pi(a) \in C_{\mathcal{T}}^{\mathcal{I}}$. Thus, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

We now show that $\mathcal{I}$-compatibility implies clash-freeness.

Claim 2: Every $\mathcal{I}$-compatible completion system is clash-free.

Proof. Let $S = (T, \mathcal{N})$ be an $\mathcal{I}$-compatible completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. Consider the two kinds of a clash:

- Due to (Ca), a clash of the form $\{A, \neg A\} \in \mathcal{L}(a)$ contradicts the semantics.
- Property (Cd) implies that $M_{\mathcal{I}}$ is a model of $\mathcal{N}$. Thus, $\mathcal{N}$ is satisfiable.

We can now describe the "guidance" of the tableau algorithm by the model $\mathcal{I}$: we ensure that, at all times, the considered completion systems are $\mathcal{I}$-compatible. This obviously holds for the initial completion system. By Claim 1, we can guide the rule applications such that only $\mathcal{I}$-compatible completion systems are obtained. By Lemma 3, the algorithm always terminates, hence also when guided in this way. Since, by Claim 2, we will not find a clash, the algorithm returns satisfiable. □

As an immediate consequence of Lemmas 3, 4 and 5, we get the following theorem:

Theorem 1. *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.*

5. Practicability

With Theorem 1, we have achieved the main aim of this paper: providing a general decidability result for description logics with both general TBoxes and concrete domains. Our second aim is to identify an algorithm that is more practicable than the existing approaches based on automata [24, 20], i.e., that can be implemented such that an acceptable runtime behaviour is observed on realistic inputs. Since we have not yet implemented our algorithm,[5] an empirical evaluation is out of reach. In the following, we discuss the practicability on a general level.

[5] This is a non-trivial task since a large number of sophisticated optimization techniques is required, c.f. [15].

Regarding an efficient implementation, the main difficulties of our algorithm compared with successfully implemented tableau algorithms such as the ones in [32, 16] are the following:

- Our algorithm requires satisfiability checks of the network $\mathcal{N}$ constructed as part of the completion system. The problem is that this check involves the *whole* network $\mathcal{N}$ rather than only small parts of it. In practice, the constructed completion systems (and associated networks) are often too large to be considered as a whole.

- The rules $R\exists_c$, $R\forall_c$, and $R\mathsf{net}$ introduce additional non-determinism. In implementations, this non-determinism induces backtracking.

It is possible that these difficulties can be overcome by developing appropriate heuristics and optimization techniques. However, there is also an easy way around them. In the following, we argue that there is a fragment of our language that still provides interesting expressive power and in which the implementation difficulties discussed above are non-existent.

The fragment of $\mathcal{ALC}(\mathcal{C})$ that we consider is obtained by making the following assumptions:

- There is only a single concrete feature g. Note that this is acceptable with constraint systems such as $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{R}}$, where g could be $\mathsf{has\text{-}extension}$ and $\mathsf{has\text{-}lifetime}$, respectively.

- There are no paths of length greater than 2, i.e., Clause 3 is eliminated from Definition 6. This is necessary since we need to introduce additional concrete features to establish path normal form if Clause 3 is present. We believe that paths of length three or more are only needed in exceptional cases, anyway.

- There exists a unique equality predicate eq in $\mathcal{C}$, i.e., for all models $N \in \mathfrak{M}$ and all $v \in V_N$, we have $(v\ \mathsf{eq}\ v) \in N$.

Going to this fragment of $\mathcal{ALC}(\mathcal{C})$ allows the following simplification of our tableau algorithm.

1. The non-deterministic $R\mathsf{net}$ rule can simply be dropped because, for each abstract node a, the network $\mathcal{N}(a)$ is either empty or consists of a single node that is related to itself via eq. Thus, every potential blocking situation is an actual blocking situation.

2. We can localize the satisfiability check of the network $\mathcal{N}$ as follows. For $a \in \mathsf{V_a}$, let $\widehat{\mathcal{N}}(a)$ denote the restriction of $\mathcal{N}$ to the g-successor

of a and the g-successors of all abstract successors of a. Instead of checking the whole network $\mathcal{N}$ for satisfiability, we separately check, for each $a \in \mathsf{V_a}$, satisfiability of $\widehat{\mathcal{N}}(a)$. It can be seen as follows that this is equivalent to a global check: first, $\mathcal{C}$ has the patchwork property. Second, due to the fact that there is only a single concrete feature g, the networks $\widehat{\mathcal{N}}(a)$ overlap at single nodes only. Due to the presence of the equality predicate eq, the overlapping part of two such networks is thus complete. Finally, it is easy to see that the patchwork property implies a more general version of itself where only the overlapping part of the two involved networks is complete, but the networks themselves are not.

Hence, the only difficulty that remains is the non-determinism of the rules $R\exists_c$ and $R\forall_c$. However, we believe that this non-determinism is not too difficult to deal with. To see this, observe that the non-deterministic choices made by these rules have only a very local impact: they only influence the outcome of the satisfiability check of the relevant local network $\widehat{\mathcal{N}}(a)$. Therefore, it does not seem necessary to implement a complex backtracking/backjumping machinery. If the concrete domain reasoner used for deciding $\mathcal{C}$-satisfiability supports disjunctions, it is even possible to push the non-determinism out of the tableau algorithm into the reasoner for $\mathcal{C}$-satisfiability. Roughly, one would need to allow disjunctions in the constraint network $\mathcal{N}$ and pass these on to the reasoner for $\mathcal{C}$.

6. Conclusion

We have proved decidability of $\mathcal{ALC}$ with ω-admissible constraint systems and general TBoxes. A close inspection of our algorithm shows that it runs in 2-NExpTime if $\mathcal{C}$-satisfiability is in NP. We conjecture that, by mixing the techniques from the current paper with those from [24, 20], 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 such as transitive roles and number restrictions should also be possible in a straightforward way.

We also exhibited the first tableau algorithm for DLs with concrete domains and general TBoxes in which the concrete domain constructors are not limited to concrete features. The algorithm has some aspects that are likely to have a negative impact on practicability unless addressed by dedicated optimization techniques. However, in Section 5 we have identified a useful fragment of $\mathcal{ALC}(\mathcal{C})$ in which these impairing aspects of the algorithm can be avoided.

While we have proved that ω-admissibility of $\mathcal{C}$ is a sufficient condition for decidability of $\mathcal{ALC}(\mathcal{C})$, it is not clear whether it is also a necessary one. We leave this question open, conjecturing that ω-admissibility is not a necessary condition.

Acknowledgements

The first authorwas supported by the EU funded IST-2005-7603 FET Project Thinking Ontologies (TONES). The second author was supported by the DFG Graduiertenkolleg 334.

References

1. Allen, J.: 1983, 'Maintaining Knowledge about Temporal Intervals'. *Communications of the ACM* **26**(11).
2. Baader, F. and P. Hanschke: 1991, 'A Scheme for Integrating Concrete Domains into Concept Languages'. In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91*. Sydney (Australia), pp. 452–457.
3. Baader, F., I. Horrocks, and U. Sattler: 2003a, 'Description Logics as Ontology Languages for the Semantic Web'. In: D. Hutter and W. Stephan (eds.): *Festschrift in honor of Jörg Siekmann*.
4. Baader, F., D. L. McGuiness, D. Nardi, and P. Patel-Schneider: 2003b, *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.
5. Balbiani, P. and J.-F. Condotta: 2002, 'Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning'. In: *Frontiers of Combining Systems (FroCoS 2002)*. pp. 162–176.
6. Bennett, B.: 1997, 'Modal Logics for Qualitative Spatial Reasoning'. *Journal of the Interest Group in Pure and Applied Logic* **4**(1).
7. Calvanese, D.: 1996, 'Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity'. In: *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*. pp. 303–307.
8. Calvanese, D., M. Lenzerini, and D. Nardi: 1998, 'Description Logics for Conceptual Data Modeling'. In: J. Chomicki and G. Saake (eds.): *Logics for Databases and Information Systems*. Kluwer Academic Publisher, pp. 229–263.
9. Egenhofer, M. J. and R. Franzosa: 1991, 'Point-set topological spatial relations.'. *International Journal of Geographical Information Systems* **5**(2), 161–174.
10. Frank, A.: 1996, 'Qualitative Spatial Reasoning: Cardinal Directions as an Example'. *International Journal of Geographical Information Systems* **10**(3), 269–290.
11. Gabbay, D. M., A. Kurucz, F. Wolter, and M. Zakharyaschev: 2003, *Many-Dimensional Modal Logics: Theory and Applications*, No. 148 in Studies in Logic and the Foundations of Mathematics. Elsevier.
12. Haarslev, V. and R. Möller: 2001, 'RACER system description'. In: R. Goré, A. Leitsch, and T. Nipkow (eds.): *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*. pp. 701–705.

13. Haarslev, V., R. Möller, and M. Wessel: 2001, 'The Description Logic $\mathcal{ALCNH}_{R^+}$ Extended with Concrete Domains: A Practically Motivated Approach'. In: R. Goré, A. Leitsch, and T. Nipkow (eds.): *Proceedings of the First International Joint Conference on Automated Reasoning IJCAR'01*. pp. 29–44.
14. Horrocks, I.: 1998, 'Using an Expressive Description Logic: FaCT or Fiction?'. In: *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR98)*. pp. 636–647.
15. Horrocks, I. and P. F. Patel-Schneider: 1999, 'Optimising Description Logic Subsumption'. *Journal of Logic and Computation* **9**(3), 267–293.
16. Horrocks, I. and U. Sattler: 1999, 'A Description Logic with Transitive and Inverse Roles and Role Hierarchies'. *Journal of Logic and Computation* **9**(3).
17. Horrocks, I. and U. Sattler: 2001, 'Ontology Reasoning in the $\mathcal{SHOQ}(D)$ Description Logic'. In: B. Nebel (ed.): *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*. pp. 199–204.
18. Horrocks, I., U. Sattler, and S. Tobies: 1999, 'Practical Reasoning for Expressive Description Logics'. In: H. Ganzinger, D. McAllester, and A. Voronkov (eds.): *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*. pp. 161–180.
19. Lutz, C.: 1999, 'Complexity of Terminological Reasoning Revisited'. In: H. Ganzinger, D. McAllester, and A. Voronkov (eds.): *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*. pp. 181–200.
20. Lutz, C.: 2002a, 'Adding Numbers to the $\mathcal{SHIQ}$ Description Logic—First Results'. In: *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*. pp. 191–202.
21. Lutz, C.: 2002b, 'PSPACE Reasoning with the Description Logic $\mathcal{ALCF}(\mathcal{D})$'. *Logic Journal of the IGPL* **10**(5), 535–568.
22. Lutz, C.: 2002c, 'Reasoning about Entity Relationship Diagrams with Complex Attribute Dependencies'. In: I. Horrocks and S. Tessaris (eds.): *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*. pp. 185–194.
23. Lutz, C.: 2003, 'Description Logics with Concrete Domains—A Survey'. In: *Advances in Modal Logics Volume 4*. pp. 265–296.
24. Lutz, C.: 2004a, 'Combining Interval-based Temporal Reasoning with General TBoxes'. *Artificial Intelligence* **152**(2), 235–274.
25. Lutz, C.: 2004b, 'NExpTime-complete Description Logics with Concrete Domains'. *ACM Transactions on Computational Logic* **5**(4), 669–705.
26. Lutz, C. and M. Milicic: 2005, 'A Tableau Algorithm for Description Logics with Concrete Domains and GCIs'. In: *Proceedings of the 14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods TABLEAUX 2005*. Koblenz, Germany, pp. 201–216.
27. Lutz, C. and F. Wolter: 2004, 'Modal Logics of Topological Relations'. In: *Proceedings of Advances in Modal Logics 2004*.
28. Nebel, B. and H.-J. Bürckert: 1995, 'Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra'. *Journal of the ACM* **42**(1), 43–66.
29. Randell, D. A., Z. Cui, and A. G. Cohn: 1992, 'A spatial logic based on regions and connection'. In: B. Nebel, C. Rich, and W. Swartout (eds.): *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. pp. 165–176.

30. Renz, J. and B. Nebel: 1999, 'On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus'. *Artificial Intelligence* **108**(1–2), 69–123.
31. Russell, S. J. and P. Norvig: 1995, *Artificial Intelligence: A Modern Approach*. Prentice Hall.
32. Schmidt-Schauß, M. and G. Smolka: 1991, 'Attributive concept descriptions with complements'. *Artificial Intelligence* **48**(1), 1–26.
33. Vilain, M. and H. Kautz: 1986, 'Constraint Propagation Algorithms for Temporal Reasoning'. In: *National Conference on Artificial Intelligence of the American Association for AI (AAAI 86)*. pp. 377–382.
34. Vilain, M., H. Kautz, and P. van Beek: 1990, 'Constraint propagation algorithms for temporal reasoning: a revised report'. In: *Readings in qualitative reasoning about physical systems*. San Francisco, CA, USA: Morgan Kaufmann, pp. 373–381.

Appendix

A. Properties of RCC8

We show that $\mathsf{RCC8}_{\mathbb{R}^2}$ has the patchwork property and the compactness property. To this end, we consider a different variant of the constraint system $\mathsf{RCC8}_{\mathbb{R}^2}$. To introduce it, we need a couple of definitions. A *fork* F is a structure $\langle W_F, R_F, \pi_F\rangle$, where

- W_F is a set $\{b_F, r_F, \ell_F\}$ of cardinality three,
- R_F is the reflexive closure of $\{(b_F, r_F), (b_F, \ell_F)\}$, and
- $\pi_F : \mathsf{Var} \to 2^{W_F}$ is a valuation such that, for each $x \in \mathsf{Var}$, we have

$$b_F \in \pi_F(x) \text{ iff } \ell_F \in \pi_F(x) \text{ or } r_F \in \pi_F(x).$$

A *fork model* M is a (finite or infinite) disjoint union of forks $F_0, F_1, \ldots$. We write W_M for $\bigcup_{i\geq 0} W_{F_i}$, R_M for $\bigcup_{i\geq 0} R_{F_i}$, and $\pi_M(x)$ for $\bigcup_{i\geq 0} \pi_{F_i}(x)$. We may interpret the RCC8 relations on a fork model M by associating a topological space $\mathfrak{T}_M$ with M: define an interior operator $\mathbb{I}_M$ by setting, for all $X \subseteq W_M$,

$$\mathbb{I}_M X := \{x \in \bigcup_{i\geq 0} W_M \mid \forall y\, (xR_M y \to y \in X)\}$$

(and thus $\mathbb{C}_M X = \{x \in W_M \mid \exists y\, (xR_M y \wedge y \in X)\}$). Let $\mathcal{RS}_M$ denote the set of non-empty regular closed subsets of W_M. We now define the constraint system

$$\mathsf{RCC8}_{\mathsf{Fork}} := \langle \mathsf{RCC8}, \mathfrak{M}_{\mathsf{Fork}}\rangle$$

by setting $\mathfrak{M}_{\mathsf{Fork}} := \{N_M \mid M \text{ a fork model}\}$, where N_M is defined by fixing a variable $v_X \in \mathsf{Var}$ for every $X \in \mathcal{RS}_M$ and setting

$$N_M := \{(v_X\ r\ v_{X'}) \mid r \in \mathsf{RCC8},\ X, X' \in \mathcal{RS}_M, \text{ and } (X, X') \in r^{\mathfrak{T}_M}\}.$$

It was shown by Renz and Nebel that satisfiability of finite constraint networks in $\mathsf{RCC8}_{\mathbb{R}^2}$ coincides with satisfiability in $\mathsf{RCC8}_{\mathsf{Fork}}$ [30]. This was extended to infinite networks in [27]:

Theorem 2. *An* RCC8*-network is satisfiable in* $\mathsf{RCC8}_{\mathbb{R}^2}$ *iff it is satisfiable in* $\mathsf{RCC8}_{\mathsf{Fork}}$.

Due to Theorem 2, it suffices to prove the patchwork property and compactness for $\mathsf{RCC8}_{\mathsf{Fork}}$. This is what we do in the following. Our proof of the patchwork property is based on a result of Gabbay et

al. [11]. To formulate it, we need to introduce the standard translation [6, 30] of RCC8-networks to the modal logic $\mathsf{S4}_u$, i.e., Lewis' (uni-modal) S4 enriched with the universal modality. We refrain from giving the syntax and semantics of $\mathsf{S4}_u$ and refer, e.g., to [11] for more information. Note, however, that formulas of $\mathsf{S4}_u$ can be interpreted in fork models.

We use $\mathbb{I}$ to denote the S4 box operator, $\Box_u$ to denote the universal box, and write $\Diamond_u\varphi$ for $\neg\Box_u\neg\varphi$ as usual. Given an RCC8-constraint $(x\,r\,y)$, we define a corresponding $\mathsf{S4}_u$-formula $(x\,r\,y)^{\bowtie}$ as follows:

$$\begin{aligned}
(x\,\mathsf{eq}\,y)^{\bowtie} &= \Box_u(x \leftrightarrow y)\\
(x\,\mathsf{dc}\,y)^{\bowtie} &= \Box_u(\neg x \vee \neg y)\\
(x\,\mathsf{ec}\,y)^{\bowtie} &= \Diamond_u(x \wedge y) \wedge \Box_u(\neg\mathbb{I}x \vee \neg\mathbb{I}y)\\
(x\,\mathsf{po}\,y)^{\bowtie} &= \Diamond_u(\mathbb{I}x \wedge \mathbb{I}y) \wedge \Diamond_u(x \wedge \neg y) \wedge \Diamond_u(\neg x \wedge y)\\
(x\,\mathsf{tpp}\,y)^{\bowtie} &= \Box_u(x \rightarrow y) \wedge \Diamond_u(x \wedge \neg\mathbb{I}y) \wedge \Diamond_u(\neg x \wedge y)\\
(x\,\mathsf{ntpp}\,y)^{\bowtie} &= \Box_u(x \rightarrow \mathbb{I}y) \wedge \Diamond_u(\neg x \wedge y)
\end{aligned}$$

Constraints $(x\,\mathsf{tppi}\,y)$ and $(x\,\mathsf{ntppi}\,y)$ are converted into $(y\,\mathsf{tpp}\,x)$ and $(y\,\mathsf{ntpp}\,x)$, respectively, and then translated as above. Observe that variables of the network are translated into propositional variables of $\mathsf{S4}_u$. For every RCC8-constraint network N, we define a corresponding set of $\mathsf{S4}_u$ formulas $N^{\bowtie}$ by setting $N^{\bowtie} := \{(x\,\mathsf{r}\,y)^{\bowtie} \mid (x\,\mathsf{r}\,y) \in N\}$. The most important property of the translation $\cdot^{\bowtie}$ is the following, as established in [30]:

Theorem 3. *Let N be a finite* RCC8*-network. Then N is satisfiable in* $\mathsf{RCC8}_{\mathsf{Fork}}$ *iff the set of* $\mathsf{S4}_u$ *formulas $N^{\bowtie}$ is satisfiable in a fork model.*

For a constraint $(x\,r\,y)$, we use $(x\,r\,y)^{\forall}$ to denote the formula obtained from $(x\,r\,y)^{\bowtie}$ by dropping all conjuncts starting with $\Diamond_u$ (assuming that $(x\,r\,y)^{\forall}$ is the constant true if all conjuncts are dropped), and likewise for $(x\,r\,y)^{\exists}$ and $\Box_u$. For networks, the notions $N^{\forall}$ and $N^{\exists}$ are defined in the obvious way.

For what follows, it will be important to identify a particular class of forks induced by a constraint network. Intuitively, this class of forks can be viewed as a canonical model for the inducing network, if this network is satisfiable. For N an RCC8-network, we set

$$\mathsf{Fork}_N := \{F \text{ a fork} \mid F \text{ satisfies } N^{\forall}\}.$$

We say that two forks F and F' are *V-equivalent*, for V a set of variables, when for all $x \in V$, we have that (i) $r_F \in \pi_F(x)$ iff $r_{F'} \in \pi_{F'}(x)$ and (ii) $\ell_F \in \pi_F(x)$ iff $\ell_{F'} \in \pi_{F'}(x)$ (recall that by definition of forks, the value of b_F is determined by those of r_F and ℓ_F). The following theorem forms the basis for our proof that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the patchwork

property. It is a reformulation of Theorem 16.17 in [11]. For $r \in \mathsf{RCC8}$, we use $\mathsf{Inv}(r)$ to denote the inverse of the relation r, e.g. $\mathsf{Inv}(\mathsf{po}) = \mathsf{po}$.

Theorem 4 (Gabbay et al.). *Let N be a finite, complete, satisfiable* RCC8*-network, $x \notin V_N$, and*

$$N' = N \cup \{(x\ r_y\ y),\ (y\ \mathsf{Inv}(r_y)\ x) \mid y \in V_N\}$$

for some family of relations $(r_y)_{y \in V_N}$, such that N' is satisfiable. Then, for each $F \in \mathsf{Fork}_N$, there exists an $F' \in \mathsf{Fork}_{N'}$ such that F and F' are V_N-equivalent.

The following corollary is easily proved by induction on the cardinality of $V_M \setminus V_N$.

Corollary 1. *Let N and M be two finite complete satisfiable* RCC8*-networks, such that $N \subseteq M$. Then, for each $F \in \mathsf{Fork}_N$, there exists an $F' \in \mathsf{Fork}_M$ such that F and F' are V_N-equivalent.*

We may now establish the patchwork property.

Lemma 6. $\mathsf{RCC8}_{\mathbb{R}^2}$ *has the patchwork property.*

Proof. By Theorem 2, it suffices to show that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the patchwork property. Let N and M be finite and complete RCC8-networks that are satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$ and whose intersection parts $I_{N,M}$ and $I_{M,N}$ (as defined in Definition 3) are identical. We have to prove that $N \cup M$ is also satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$. By Theorem 3, it suffices to show that $(N \cup M)^{\bowtie}$ is satisfiable in a fork model. We show that a satisfying model is provided by $\mathcal{F}_{N,M} := \mathsf{Fork}_N \cap \mathsf{Fork}_M$. We distinguish between the universal and existential part of $(N \cup M)^{\bowtie}$.

(i) $\mathcal{F}_{N,M}$ satisfies $(N \cup M)^{\forall} = N^{\forall} \cup M^{\forall}$. It suffices to show that every $F \in \mathcal{F}_{N,M}$ satisfies $N^{\forall}$ and $M^{\forall}$. The former is an immediate consequence of $\mathcal{F}_{N,M} \subseteq \mathsf{Fork}_N$ and the definition of Fork_N. The argument for the latter is analogous.

(ii) $\mathcal{F}_{N,M}$ satisfies $(N \cup M)^{\exists} = N^{\exists} \cup M^{\exists}$. To show this, it is sufficient to show that (a) for every $F \in \mathsf{Fork}_N$, there is an $F' \in \mathcal{F}_{M,N}$ which is V_N-equivalent to F and (b) for every $F \in \mathsf{Fork}_M$, there is an $F' \in \mathcal{F}_{M,N}$ which is V_M-equivalent to F. Then, since Fork_N satisfies $N^{\bowtie}$, all $\Diamond_u \varphi \in N^{\exists}$ will be satisfied by $\mathcal{F}_{M,N}$, and likewise for M. We only show (a) as (b) is analogous. For brevity, let I denote $I_{N,M}$ $(= I_{M,N})$. Take an $F \in \mathsf{Fork}_N$. Clearly, since $I \subseteq N$, we have that $F \in \mathsf{Fork}_I$. Moreover, I is finite, complete, and satisfiable since N and M are. Thus, by Corollary 1 there exists an $F' \in \mathsf{Fork}_M$ that

is V_I-equivalent to F. Now define a fork $F'' = (W_{F''}, R_{F''}, \pi_{F''})$ as follows:

$$\pi_{F''}(x) := \begin{cases} \pi_F(x) & \text{if } x \in V_N \\ \pi_{F'}(x) & \text{otherwise} \end{cases}$$

It is not difficult to see that F'' is V_N-equivalent to F and V_M-equivalent to F'. Since V_N is clearly closed under V_N-equivalence (and likewise for V_M), this yields $F'' \in \mathsf{Fork}_N \cap \mathsf{Fork}_M = \mathcal{F}_{M,N}$.

□

It remains to treat compactness.

Lemma 7. $\mathsf{RCC8}_{\mathbb{R}^2}$ *has the compactness property.*

Proof. It is easily seen that satisfiability of an infinite $\mathsf{RCC8}$-network N implies satisfiability of $N|_V$, for every finite $V \subseteq V_N$. To show the converse, we give a satisfiability preserving translation of $\mathsf{RCC8}$-networks N to a set $\Gamma(N)$ of first-order sentences in the following signature: a binary predicate R representing the partial order in fork frames and unary predicates $(P_x)_{x \in \mathsf{Var}}$ for variables. We then use compactness of first-order logic to deduce that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the compactness property. By Theorem 2, it follows that $\mathsf{RCC8}_{\mathbb{R}^2}$ has the compactness property. Let N be a (possibly infinite) $\mathsf{RCC8}$-network. The set of first-order sentences $\Gamma(N)$ consists of the following:

– a formula stating that R is a disjoint union of forks:

$$\begin{aligned} \forall w \exists x, y, z (&xRx \wedge yRy \wedge zRz \wedge xRy \wedge xRz \wedge \\ &\forall u(xRu \rightarrow (u = x \vee u = y \vee u = z)) \wedge \\ &\forall u(yRu \rightarrow u = y) \wedge \\ &\forall u(zRu \rightarrow u = z) \wedge \\ &\forall u(uRx \rightarrow u = x) \wedge \\ &\forall u(uRy \rightarrow (u = x \vee u = y)) \wedge \\ &\forall u(uRz \rightarrow (u = x \vee u = z)) \wedge \\ &x \neq y \wedge x \neq z \wedge y \neq z \wedge \\ &(w = x \vee w = y \vee w = z)) \end{aligned}$$

– to ensure the restriction that is imposed on valuations of fork models, for each unary predicate P, we add the following formula:

$$\forall x(\mathsf{root}(x) \rightarrow (P(x) \leftrightarrow \exists y(xRy \wedge x \neq y \wedge P(y))))$$

where $\mathsf{root}(x) := \forall y(yRx \rightarrow x = y)$ expresses that x is the root of a fork.

– the translation of each constraint in N. We only treat the case $(x\ \mathsf{ec}\ y)$ explicitly:

$$\exists z(P_x(z) \wedge P_y(z)) \wedge \neg\exists z(\mathsf{Int}_x(z) \wedge \mathsf{Int}_y(z))$$

where $\mathsf{Int}_x(z) := P_x(z) \wedge \forall z'(zRz' \rightarrow P_y(z'))$ describes the interior points of P_x (to see this, consider the way in which fork frames induce topologies). The other cases are easily obtained by referring to the semantics of the RCC8 relations.

Now let N be an infinite RCC8-network such that $N|_V$ is satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$ for every finite $V \subseteq V_N$. We have to show that N is satisfiable. Let Ψ be a finite subset of $\Gamma(N)$, and let N' be the fragment of N that contains precisely those constraints whose translation is in Ψ. By Theorem 3, N' has a model that is the topology of a fork model M. Define a first-order structure $\mathfrak{M}$ with domain W_M by setting $R^{\mathfrak{M}} := R_M$ and $P_x^{\mathfrak{M}} := \pi_M(x)$ for all $x \in V$. It is readily checked that $\mathfrak{M}$ is a model of Ψ. Thus, every finite subset of $\Gamma(N)$ is satisfiable and compactness of first-oder logic implies that $\Gamma(N)$ is satisfiable. Take a model $\mathfrak{N}$ of $\Gamma(N)$ with domain $\mathfrak{A}$. Clearly, $M' = (\mathfrak{A}, R^{\mathfrak{N}}, \{x \mapsto P_x^{\mathfrak{N}}\})$ is a fork model. It is readily checked that the topology $\mathfrak{T}_{M'}$ is a model of N. □

B. Properties of Allen

We prove that the constraint system $\mathsf{Allen}_{\mathbb{R}}$ has both the patchwork property and the compactness property.

Lemma 8. $\mathsf{Allen}_{\mathbb{R}}$ *has the patchwork property.*

Proof. Let N and M be finite complete Allen-networks that are satisfiable in $\mathsf{Allen}_{\mathbb{R}}$ and whose intersection parts $I_{N,M}$ and $I_{M,N}$ (defined as in Definition 3) are identical. We have to prove that $N \cup M$ is also satisfiable. Satisfiability of N means that there exists a mapping $\tau_N : V_N \rightarrow \mathsf{Int}_{\mathbb{R}}$ such that $(x\,\mathsf{r}\,y) \in N$ implies $(\tau_N(x), \tau_N(y)) \in \mathsf{r}^{\mathbb{R}}$, and an analogous mapping τ_M for M. Define

$$\begin{aligned} S_N \ := \ & \{(x, L, r) \mid x \in V_{I_{N,M}} \text{ and } \tau_N(v) = [r, r'] \text{ for some } r' \in \mathbb{R}\} \cup \\ & \{(x, R, r) \mid x \in V_{I_{N,M}} \text{ and } \tau_N(v) = [r', r] \text{ for some } r' \in \mathbb{R}\} \end{aligned}$$

Now arrange the elements of S_N in a sequence $(v_0, D_0, r_0), \ldots, (v_k, D_k, r_k)$ such that $i < j$ implies $r_i \leq r_j$. Define a corresponding sequence $(v_0, D_0, r'_0), \ldots, (v_k, D_k, r'_k)$ for M by setting, for $i \leq k$,

$$r'_i := \begin{cases} r & \text{if } D_i = L \text{ and } \tau_M(x_i) = (r, r') \text{ for some } r' \in \mathbb{R} \\ r & \text{if } D_i = R \text{ and } \tau_M(x_i) = (r', r) \text{ for some } r' \in \mathbb{R}. \end{cases}$$

Since $I_{N,M} = I_{M,N}$, we have that $i < j$ implies $r'_i \leq r'_j$. Fix, for each $i < k$, a bijection π_i from the interval $[r'_i, r'_{i+1})$ to the interval $[r_i, r_{i+1})$ that is an isomorphism w.r.t. "<". Moreover, fix additional isomorphisms $\pi^* : (-\infty, r'_0)$ to $(-\infty, r_0)$ and $\pi^\dagger : [r'_k, \infty)$ to $[r_k, \infty)$. For $r \in \mathbb{R}$, set

$$\pi(r) := \begin{cases} \pi^*(r) & \text{if } r < r'_0 \\ \pi_i(r) & \text{if } r_i \leq r < r'_{i+1} \\ \pi^\dagger(r) & \text{if } r \geq r_k \end{cases}$$

Now define a mapping $\tau'_M : V_M \to \mathsf{Int}_{\mathbb{R}}$ by setting $\tau'_M(x) := [\pi(r), \pi(r')]$ if $\tau_M(x) = [r, r']$. It is readily checked that τ_N and τ'_M agree on $V_{I_{N,M}}$, and that $\tau_N \cup \tau'_M$ witnesses satisfaction of $N \cup M$ in $\mathsf{Allen}_{\mathbb{R}}$. □

Lemma 9. $\mathsf{Allen}_{\mathbb{R}}$ *has the compactness property.*

Proof. As in the case of RCC8, it is easily seen that satisfiability of an infinite Allen-network N implies satisfiability of $N|_V$, for every finite $V \subseteq V_N$. To show the converse, we give a satisfiability preserving translation of Allen-networks N to a set $\Gamma(N)$ of first-order sentences in the following signature: a binary predicate $<$ representing the ordering on $\mathbb{R}$, and constants $(b_x)_{x \in \mathsf{Var}}$ and $(e_x)_{x \in \mathsf{Var}}$ denoting the begin and end points of intervals. Let N be a (possibly infinite) constraint network. The set of first-order sentences $\Gamma(N)$ consists of the following:

- one sentence for each constraint in N. The translation is easily read off from the definition of the Allen relations. E.g., $(x \ \mathsf{m} \ y)$ translates to $e_x = b_y$;
- for each $x \in V_N$, a sentence ensuring the correct ordering of endpoints: $b_x < e_x$.

It is easily seen that each finite or infinite Allen-network N is satisfiable in $\mathsf{Allen}_{\mathbb{R}}$ iff $\Gamma(N)$ is satisfiable in a structure $(\mathbb{R}, <, P_1^{\mathfrak{M}}, P_2^{\mathfrak{M}}, \ldots)$. Thus, compactness of first-order logic on such structures implies that $\mathsf{Allen}_{\mathbb{R}}$ has the compactness property. □