TECHNISCHE UNIVERSITÄT DRESDEN

Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

A Tableau Algorithm for DLs with Concrete Domains and GCIs

Carsten Lutz   Maja Miličić

LTCS-Report 05-07

Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Hans-Grundig-Str. 25
01062 Dresden
Germany

A Tableau Algorithm for DLs with Concrete Domains and GCIs

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

Abstract

We identify a general property of concrete domains that is sufficient for proving decidability of DLs equipped with them and GCIs. We show that some useful concrete domains, such as a temporal one based on the Allen relations and a spatial one based on the RCC-8 relations, have this property. Then, we present a tableau algorithm for reasoning in DLs equipped with such concrete domains.

1 Introduction

In many relevant applications of description logics (DLs) such as the semantic web and reasoning about ER and UML diagrams, there is a need for DLs that are equipped with both concrete domains and general concept inclusions (GCIs) [2, 6, 13]. Unfortunately, combining concrete domains with GCIs easily leads to undecidabilty. For example, it has been shown in [16] that the basic DL $\mathcal{ALC}$ extended with GCIs and a concrete domain based on the natural numbers and providing for equality and incrementation predicates is undecidable. More information can be found in the survey paper [14].

In view of this discouraging result, it is a natural question whether there are *any* useful concrete domains such that, when used with a DL providing for GCIs, reasoning remains decidable. A positive answer to this question has been given in [15] and [12], where two such well-behaved concrete domains are identified: a temporal one based on the Allen relations and a numerical one based on the rationals and equipped with various unary and binary predicates such as "$\leq$", "$>_5$", and "$\neq$". Using an automata-based approach, it is shown in [15, 12] that reasoning in the DLs $\mathcal{ALC}$ and $\mathcal{SHIQ}$ extended with these concrete domains and GCIs is decidable and ExpTime-complete.

The purpose of this paper it to elaborate on the existing decidability results. Our contribution is two-fold: first, instead of focussing on particular concrete domains as in previous work, we identify a *general* property of concrete domains, called ω-admissibility, that is sufficient for proving decidability of DLs equipped with concrete domains and GCIs. For defining ω-admissibility, we concentrate on a particular kind of concrete domains: *constraint systems*. Roughly, a constraint system is a concrete domain that only has binary predicates, and these predicates are interpreted as jointly exhaustive and pairwise disjoint (JEPD) relations. We exhibit two example constraint systems that are ω-admissible: a temporal one based on the rational line and the Allen relations [1], and a spatial one based on the real plane and the RCC8 relations [4, 18]. The proof of ω-admissibility turns out to be relatively straightforward in the Allen case, but is somewhat cumbersome for RCC8.

Second, for the first time we develop a tableau algorithm for DLs admitting both concrete domains and GCIs. This algorithm is used to establish decidability of $\mathcal{ALC}$ equipped with ω-admissible concrete domains and GCIs. As state-of-the-art DL reasoners such as FaCT and RACER are based on tableau algorithms similar to the one described by us [9, 8], we view our algorithm as a first step towards an efficient implementation of description logics with (ω-admissible) concrete domains and GCIs. Our decidability result reproves the decidability of $\mathcal{ALC}$ with GCIs and the Allen relations from [15], and, as a new result, establishes decidability of $\mathcal{ALC}$ with GCIs and the RCC8 relations as a concrete domain.

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, for example, the RCC8 relations used for spatial reasoning and Allen's relations of time intervals [4, 1].

Definition 1 (Constraint System). Let Var be a countably infinite set of variables and Rel a finite set of binary relation symbols. A Rel-*constraint* is an expression $(v\ r\ v')$ with $v, v' \in \mathsf{Var}$ and $r \in \mathsf{Rel}$. A Rel-*network* is a (finite or infinite) set of Rel-constraints. 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 $v, v' \in V_N$, there is exactly one constraint $(v\ r\ v') \in N$;
- a *model of a network* N' if N is complete and there is a mapping $\tau : V_{N'} \to V_N$ such that $(v\ r\ v') \in N'$ implies $(\tau(v)\ r\ \tau(v')) \in N$.

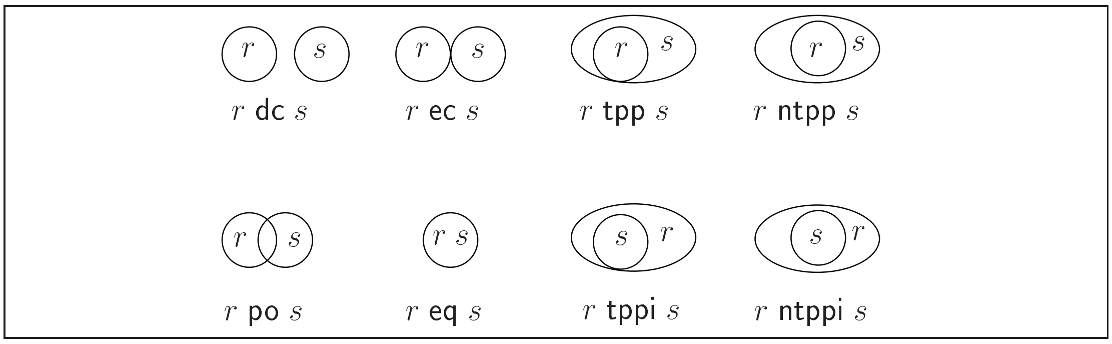

Figure 1: The eight RCC8 relations.

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. △

As examples, we define two constraint systems: one system 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 [18]. 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. Examples of these relations are given in Figure 1. Recall that a topological space is a pair $\mathfrak{T} = (U, \mathbb{I})$, where U is a set and $\mathbb{I}$ is an *interior operator* on U, i.e., for all $s, t \subseteq U$, we have

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

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

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

Let $\mathfrak{T}_{\mathbb{R}^2}$ be the standard topology on $\mathbb{R}^2$ induced by the Euclidean metric, and let $\mathcal{RS}_{\mathbb{R}^2}$ be the set of all non-empty regular-closed subsets of $\mathfrak{T}_{\mathbb{R}^2}$. 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 $N_{\mathbb{R}^2} := \{(v_s \ r \ v_t) \mid r \in \mathsf{RCC8}, \ s,t \in \mathcal{RS}_{\mathbb{R}^2} \text{ and } (s,t) \in r^{\mathfrak{T}_{\mathbb{R}^2}}\}$.

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 rational numbers with the usual ordering. Let $\mathsf{Int}_{\mathbb{Q}}$ denote the set of all closed intervals $[q_1, q_2]$ over $\mathbb{Q}$ with $q_1 < q_2$, i.e., point-intervals are not admitted. The extension $\mathsf{r}^{\mathbb{Q}}$ of each Allen relation r is a subset of $\mathsf{Int}_{\mathbb{Q}} \times \mathsf{Int}_{\mathbb{Q}}$. It is defined in terms of the relationships between endpoints in the obvious way, c.f. Figure 2. We define the constraint system

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

by setting $\mathfrak{M}_{\mathbb{Q}} := \{N_{\mathbb{Q}}\}$, where $N_{\mathbb{Q}}$ is defined by fixing a variable $v_i \in \mathsf{Var}$ for every $i \in \mathsf{Int}_{\mathbb{Q}}$ and setting $N_{\mathbb{Q}} := \{(v_i \ r \ v_j) \mid r \in \mathsf{Allen}, \ i,j \in \mathsf{Int}_{\mathbb{Q}} \text{ and } (i,j) \in r^{\mathbb{Q}}\}$.

We could also define the constraint system $\mathsf{Allen}_{\mathbb{R}}$ based on the reals rather than on the rationals: this has no impact on the satisfiability of (finite and infinite) Allen-networks. 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 or integers.

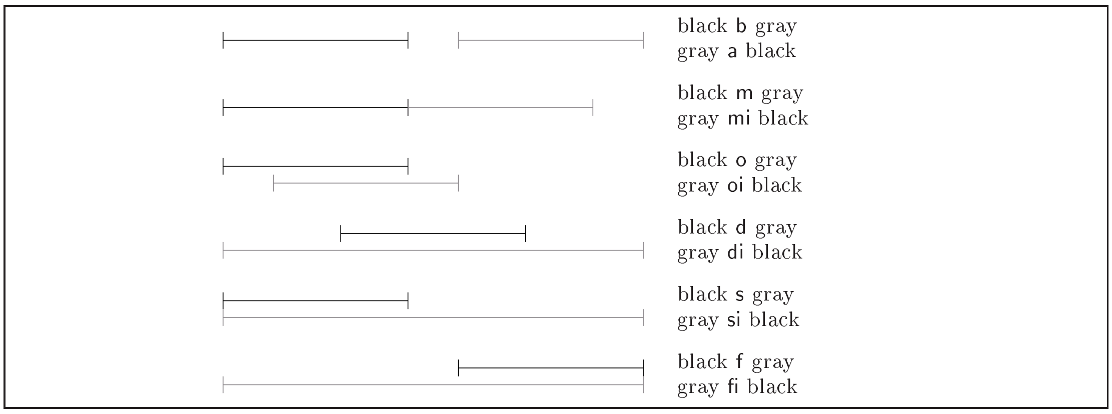

Figure 2: The thirteen Allen relations.

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 constraint system to have certain properties.

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 2 (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

$$\begin{array}{lcl} I_{N,M} & := & \{(v\ r\ v') \mid v, v' \in V_N \cap V_M \text{ and } (v\ r\ v') \in N\} \\ I_{M,N} & := & \{(v\ r\ v') \mid v, v' \in V_N \cap V_M \text{ and } (v\ r\ v') \in M\} \end{array}$$

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 in [3], where constraint networks are combined with linear temporal logic.

For using constraint systems with the DL reasoning algorithms 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. As we shall see, to guarantee this it suffices to require constraint systems to be compact.

Definition 3 (Compactness). Let $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ be a constraint system. If N is a Rel-network and $V \subseteq V_N$, we write $N|_V$ to denote the network $\{(v\,r\,v') \in N \mid v, v' \in V\} \subseteq N$. 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 $\mathcal{C}$. △

2.4 Properties of RCC8

We show that $\mathsf{RCC8}_{\mathbb{R}^2}$ has the patchwork property and the compactness property, and thus can be used together with the algorithms developed in the current paper. 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 $v \in \mathsf{Var}$, we have

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

A *fork model* is a (finite or infinite) disjoint union of forks. If a fork model M is the 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 $\pi_M(v)$ for $\bigcup_{i\geq 0} \pi_{F_i}(v)$. 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 $X \subseteq W_M$,

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

(and thus $\mathbb{C}_M X = \{x \in W_M \mid \exists y\ (x R_M y \wedge y \in X)\}$). 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 non-empty regular closed $X \subseteq W_M$ and setting $N_M := \{(v_X\ r\ v_{X'}) \mid r \in \mathsf{RCC8},\ X, X' \in W_M, \text{ and } (X, X') \in r^{\mathfrak{T}_M}\}$.

It was shown by Renz and Nebel that satisfiability of finite constraint networks in $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{RCC8}_{\mathsf{Fork}}$ coincides [19]. This was extended to infinite networks in [17]:

Theorem 4. *An* RCC8*-network is satisfiable in* $\mathsf{RCC8}_{\mathbb{R}^2}$ *iff it is satisfiable in* $\mathsf{RCC8}_{\mathsf{Fork}}$.

Due to Theorem 4, 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. [7]. To formulate it, we need to introduce the standard translation [4, 19] 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 [7] for more information.

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 $(v\, r\, v')$, we define a corresponding $\mathsf{S4}_u$-formula $(v\,r\,v')^{\bowtie}$ as follows:[1]

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

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 $N^{\bowtie}$ as follows: $N^{\bowtie} := \{(v\,\mathsf{r}\,v')^{\bowtie} \mid (v\,\mathsf{r}\,v') \in N\}$. An important property of the translation $\cdot^{\bowtie}$ is the following, as established in [19]:

Theorem 5. *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 $(v\,r\,v')$, we use $(v\,r\,v')^{\forall}$ to denote the formula obtained from $(v\,r\,v')^{\bowtie}$ by dropping all conjuncts starting with $\Diamond_u$ (with $(v\,r\,v')^{\forall}$ being the constant true if all conjuncts are dropped), and likewise for $(v\,r\,v')^{\exists}$. 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

$$\mathsf{Fork}_N := \{F \text{ a fork} \mid F \models \bigwedge_{(v\,\mathsf{r}\,v') \in N} (v\,\mathsf{r}\,v')^{\forall}\}.$$

We say that two forks F and F' are *V-equivalent*, for V a set of variables when, for all $v \in V$, we have that (i) $r_F \in \pi_F(v)$ iff $r_{F'} \in \pi_{F'}(v)$ and (ii) $\ell_F \in \pi_F(v)$ iff $\ell_{F'} \in \pi_{F'}(v)$. The following theorem forms the basis for our proof that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the patchwork property. It is is easily seen to be a consequence of Theorem 16.17 in [7].[2]

Theorem 6 (Gabbay et al.). *Let N be a finite, complete, satisfiable* RCC8*-network, $v \notin V_N$, and*

$$N' = N \cup \{(v\ r_v\ v'),\ (v'\ \mathsf{Inv}(r_v)\ v) \mid v \in V_N\}$$

for some family of relations $(r_v)_{v \in V_N}$, 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.

[1]We do not consider constraints of the form $(v\,\mathsf{tppi}\,v')$ and $(v\,\mathsf{ntppi}\,v')$ since these can be translated into $(v'\,\mathsf{tpp}\,v)$ and $(v'\,\mathsf{ntpp}\,v)$, respectively.

[2]It is easily checked that this holds although our definition of Fork_N differs from the corresponding definition in [7]: Gabbay et al. define Fork_N such that, intuitively, it contains no two forks that are isomorphic. This is not necessary in our case.

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

Corollary 7. *Let N and M be two finite complete satisfiable* $\mathsf{RCC8}$*-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 8. $\mathsf{RCC8}_{\mathbb{R}^2}$ *has the patchwork property.*

Proof. By Theorem 4, it suffices to show that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the patchwork property. Let N and M be finite and complete $\mathsf{RCC8}$-networks that are satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$ and whose intersection parts $I_{N,M}$ and $I_{M,N}$ (defined as in Definition 2) are identical. We have to prove that $N \cup M$ is also satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$. By Theorem 5, it suffices to show that the fork model $\mathcal{F}_{N,M} := \mathsf{Fork}_N \cap \mathsf{Fork}_M$ satisfies $(N \cup M)^{\bowtie}$. We distinguish between the universal and existential part of $(N \cup M)^{\bowtie}$.

(i) $\mathcal{F}_{N,M}$ satisfies $(N \cup M)^{\forall}$ due to the definition of Fork_N and Fork_M:

$$\begin{aligned}
\mathcal{F}_{N,M} &= \mathsf{Fork}_N \cap \mathsf{Fork}_M \\
&= \{F \text{ a fork} \mid F \models \bigwedge_{(v\,\mathsf{r}\,v') \in N} (v\,\mathsf{r}\,v')^{\forall}\} \cap \\
&\quad\ \{F \text{ a fork} \mid F \models \bigwedge_{(v\,\mathsf{r}\,v') \in M} (v\,\mathsf{r}\,v')^{\forall}\} \\
&= \{F \text{ a fork} \mid F \models \bigwedge_{(v\,\mathsf{r}\,v') \in N \cup M} (v\,\mathsf{r}\,v')^{\forall}\} = \mathsf{Fork}_{N \cup M}
\end{aligned}$$

(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 is a model of N, all existential restrictions from $N^{\exists}$ will be satisfied by $\mathcal{F}_{M,N}$, and likewise for M. We only show (a) as (b) is equivalent. 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 complete since N and M are. Thus, by Corollary 7 there exists an $F' \in \mathsf{Fork}_M$ which is V_I-equivalent to F. Now define a fork $F'' = (W_{F'}, R_{F'}, \pi_{F'})$ as follows:

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

Obviously, F'' is V_N-equivalent to F. Moreover, since F'' is I-equivalent to F, we have that F'' is V_M-equivalent to F'. Therefore, $F' \in \mathsf{Fork}_M$ implies that $F'' \in \mathsf{Fork}_M$.

❑

It remains to treat compactness.

Lemma 9. $\mathsf{RCC8}_{\mathbb{R}^2}$ *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 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 $(P_v)_{v \in \mathsf{Var}}$ for variables. We then use compactness of first-order logic to deduce that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the compactness property. By Theorem 4, it follows that $\mathsf{RCC8}_{\mathbb{R}^2}$ has the compactness property as desired.

Let N be a (possibly infinite) constraint network. The set of first-order sentences $\Gamma(N)$ consists of the following:

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

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

- to ensure the constraint on valuations for fork models, for each $v \in \mathsf{Var}$, the following formula:

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

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

- one sentence for each constraint in N. We only treat the case $(v \;\mathsf{ec}\; v')$ explicitly:

$$\exists x(P_v(x) \wedge P_{v'}(x)) \wedge \neg \exists x(\mathsf{Int}_v(x) \wedge \mathsf{Int}_{v'}(x))$$

  where $\mathsf{Int}_v(x) := P_v(x) \wedge \forall y(xRy \to P_v(y))$ describes the interior points of P_v (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 $\mathsf{RCC8}$ relations.

Now let N be an infinite 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 corresponding fragment of N. By Theorem 5, 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 $P_v^{\mathfrak{M}} := \pi_M(v)$ for all $v \in V$. 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, $M' = (\mathfrak{A}, R^{\mathfrak{N}}, \{v \mapsto P_v^{\mathfrak{N}}\})$ is a fork model. It is readily checked that the topology $\mathfrak{T}_{M'}$ is a model of N. ❑

2.5 Properties of Allen

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

Lemma 10. $\mathsf{Allen}_{\mathbb{Q}}$ *has the patchwork property.*

Proof. Let N and M be finite complete Allen-networks that are satisfiable in $\mathsf{Allen}_{\mathbb{Q}}$ and whose intersection parts $I_{N,M}$ and $I_{M,N}$ (defined as in Definition 2) are identical. We have to prove that $N \cup M$ is also satisfiable. Satisfiability of N means that there exists a mapping $\tau_N : V_N \to \mathsf{Int}_{\mathbb{Q}}$ such that $(v \, r \, v') \in N$ implies $(\tau_N(v), \tau_N(v')) \in \mathsf{r}^{\mathbb{Q}}$, and an analogous mapping τ_M for M. Define

$$\begin{array}{rcl} S_N & := & \{(v, \ell, q) \mid v \in V_{I_{N,M}} \text{ and } \tau_N(v) = [q, q'] \text{ for some } q' \in \mathbb{Q}\} \cup \\ & & \{(v, r, q) \mid v \in V_{I_{N,M}} \text{ and } \tau_N(v) = [q', q] \text{ for some } q' \in \mathbb{Q}\} \end{array}$$

Now arrange the elements of S_N into a sequence $(v_0, d_0, q_0), \ldots, (v_k, d_k, q_k)$ such that $i < j$ implies $q_i \leq q_j$. Define a corresponding sequence $(v_0, d_0, q_0'), \ldots, (v_k, d_k, q_k')$ for M by setting, for $i \leq k$,

$$q_i' := \begin{cases} q & \text{if } d_i = \ell \text{ and } \tau_M(v_i) = (q, q') \text{ for some } q' \in \mathbb{Q} \\ q & \text{if } d_i = r \text{ and } \tau_M(v_i) = (q', q) \text{ for some } q' \in \mathbb{Q}. \end{cases}$$

Since $I_{N,M} = I_{M,N}$, we have that $i < j$ implies $q_i' \leq q_j'$. Fix, for each $i < k$, a bijection π_i from the interval $[q_i', q_{i+1}')$ to the interval $[q_i, q_{i+1})$ that is an isomorphism w.r.t. "<". Moreover, fix additional isomorphisms $\pi^* : (-\infty, q_0')$ to $(-\infty, q_0)$ and $\pi^\dagger : [q_k', \infty)$ to $[q_k, \infty)$. For $q \in \mathbb{Q}$, set

$$\pi(q) := \begin{cases} \pi^*(q) & \text{if } q < q_0' \\ \pi_i(q) & \text{if } q_i \leq q < q_{i+1}' \\ \pi^\dagger(q) & \text{if } q \geq q_k \end{cases}$$

Now define a mapping $\tau_M' : V_M \to \mathsf{Int}_{\mathbb{Q}}$ by setting $\tau_M'(v) := [\pi(q), \pi(q')]$ if $\tau_M(v) = [q, q']$. 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 $\mathsf{Allen}_{\mathbb{Q}}$. ❑

Lemma 11. $\mathsf{Allen}_{\mathbb{Q}}$ *has the compactness property.*

Proof. As in the case of $\mathsf{RCC8}$, 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 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 $\mathbb{Q}$, and constants $(b_v)_{v \in \mathsf{Var}}$ and $(e_v)_{v \in \mathsf{Var}}$ 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., $(v \mathrel{\mathsf{m}} v')$ translates to $e_v = b_{v'}$;
- for each $v \in V_N$, a sentence ensuring the correct ordering of endpoints: $b_v < e_v$.

It is easily seen that each finite or infinite Allen-network N is satisfiable in $\mathsf{Allen}_{\mathbb{Q}}$ iff $\Gamma(N)$ is satisfiable in a structure $(\mathbb{Q}, <, P_1^{\mathfrak{M}}, P_2^{\mathfrak{M}}, \dots)$. Thus, compactness of first-order logic on structures $(\mathbb{Q}, <, c_1^{\mathfrak{M}}, c_2^{\mathfrak{M}}, \dots)$ implies that $\mathsf{Allen}_{\mathbb{Q}}$ has the compactness property. ❑

3 Syntax and Semantics

In this section we formaly introduce the syntax and semantics of the description logic $\mathcal{ALC}(\mathcal{C})$. We start by defining $\mathcal{ALC}(\mathcal{C})$-concepts and TBoxes, based on a constraint system $\mathcal{C} = (\mathsf{Rel}, \mathfrak{M})$.

Definition 12 ($\mathcal{ALC}(\mathcal{C})$-concepts). 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 $\mathsf{N_{rR}}$. The elements of $\mathsf{N_{aF}}$ are called *abstract features* and the elements of $\mathsf{N_{rR}}$ *regular roles.* A *path* is a sequence $R_1 \cdots R_k g$ consisting of roles $R_1, \dots, R_k \in \mathsf{N_R}$ and a concrete feature $g \in \mathsf{N_{cF}}$. A path $R_1 \cdots R_k g$ with $\{R_1, \dots, R_k\} \subseteq \mathsf{N_{aF}}$ is called *feature path.* The set of $\mathcal{ALC}(\mathcal{C})$-concepts is the smallest set such that

1. 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 $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, and $\exists R.C$ are concepts,
3. if u_1 and u_2 are feature paths and $r \in \mathsf{Rel}$, then $\exists u_1, u_2.r$ and $\forall u_1, u_2.r$ are concepts,
4. if $R \in \mathsf{N_{rR}}$, g_1 and g_2 are concrete features, and $r \in \mathsf{Rel}$, then $\exists R g_1, g_2.r$, $\exists g_1, R g_2.r$, $\forall R g_1, g_2.r$, and $\forall g_1, R g_2.r$ are concepts,

A *general concept inclusion axiom (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are concepts. A finite set of GCIs is called *TBox*. △

The TBox formalism introduced in Definition 12 is often called *general TBox* since it subsumes several other, much weaker variants [5, 11]. Throughout this paper, we use $\top$ as abbreviation for an arbitrary propositional tautology and $C \rightarrow D$ for $\neg C \sqcup D$.

As most description logics, the semantics of $\mathcal{ALC}(\mathcal{C})$ is defined in terms of interpretations.

Definition 13 ($\mathcal{ALC}(\mathcal{C})$ 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 $\Delta_\mathcal{I}$, and
- each concrete feature g to a partial function $g^\mathcal{I}$ from $\Delta_\mathcal{I}$ to the set of variables $V_{M_\mathcal{I}}$ of $M_\mathcal{I}$.

The interpretation function is extended to arbitrary concepts as follows:

$$\begin{array}{rcl}
(C \sqcap D)^\mathcal{I} & := & C^\mathcal{I} \cap D^\mathcal{I}, \\
(C \sqcup D)^\mathcal{I} & := & C^\mathcal{I} \cup D^\mathcal{I}, \\
\neg C^\mathcal{I} & := & \Delta^\mathcal{I} \setminus C^\mathcal{I}, \\
(\exists R.C)^\mathcal{I} & := & \{d \in \Delta^\mathcal{I} \mid \text{there is some } e \in \Delta^\mathcal{I} \text{ with } (d,e) \in R^\mathcal{I} \text{ and } e \in C^\mathcal{I}\}, \\
(\forall R.C)^\mathcal{I} & := & \{d \in \Delta^\mathcal{I} \mid \text{for all } e \in \Delta^\mathcal{I}, \text{ if } (d,e) \in R^\mathcal{I}, \text{ then } e \in C^\mathcal{I}\}, \\
(\exists U_1, U_2.r)^\mathcal{I} & := & \{d \in \Delta^\mathcal{I} \mid \text{there are } x_1 \in U_1^\mathcal{I}(d) \text{ and } x_2 \in U_2^\mathcal{I}(d) \text{ with } (x_1 r x_2) \in M_\mathcal{I} \ \} \\
(\forall U_1, U_2.r)^\mathcal{I} & := & \{d \in \Delta^\mathcal{I} \mid \text{for all } x_1 \in U_1^\mathcal{I}(d) \text{ and } x_2 \in U_2^\mathcal{I}(d), \text{ we have } (x_1 r x_2) \in M_\mathcal{I}\}
\end{array}$$

where U_1 and U_2 denote paths, and, for every path $U = R_1 \cdots R_k g$ and $d \in \Delta_\mathcal{I}$, $U^\mathcal{I}(d)$ is defined as

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

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

C is called *satisfiable with respect to a TBox* $\mathcal{T}$ iff there exists a 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 $\mathcal{T}$. △

4 Tableau Algorithm

In this section we present a tableau algorithm for $\mathcal{ALC}(\mathcal{C})$ with general TBoxes. In order to ensure the correctness of the algorithm, we have to require the underlying constraint system $\mathcal{C}$ to be ω-admissible:

Definition 14 (ω-admissible). Let $\mathcal{C} = (\mathsf{Rel}, \mathfrak{M})$ be a constraint system. We say that $\mathcal{C}$ is *ω-admissible* iff the following holds:

1. satisfiability in $\mathcal{C}$ is decidable;
2. $\mathcal{C}$ has the patchwork property (c.f. Definition 2);
3. $\mathcal{C}$ has the compactness property (c.f. Definition 3).

$\triangle$

In Section 2, we have introduced two constraint systems: (i) $\mathsf{Allen}_{\mathbb{Q}}$ and (ii) $\mathsf{RCC8}_{\mathbb{R}^2}$ and shown that they have patchwork and compactness property. Moreover, satisfiability is decidable in both of them, and, more precisely, it is NP-complete. The proofs of NP-completeness can be found in [20] for $\mathsf{Allen}_{\mathbb{Q}}$ and in [19] for $\mathsf{RCC8}_{\mathbb{R}^2}$. Thus, we obtain that that these constraint systems are ω-admissible.

Before presenting the tableau algorithm, we need some more prerequisites. More specifically, we assume concepts and TBoxes to be in negation normal form (NNF) and, more importantly, restrict the length of paths, which will turn out to be rather convenient for constructing tableau algorithm. 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 assuming NNF is not a restriction. A notable difference to usual NNF transformations is that we preserve only (un)satisfiability, but not equivalence. Since relations from Rel are JEPD, we assume w.l.o.g. that Rel contains an equality predicate.

Lemma 15 (NNF Conversion). *Exhaustive application of the following rewrite rules translates $\mathcal{ALC}(\mathcal{C})$-concepts to eqi-satisfiable ones in NNF.*

$$\begin{array}{rcll}
\neg\neg C & \rightsquigarrow & C & \\
\neg(C \sqcap D) & \rightsquigarrow & \neg C \sqcup \neg D & \neg(C \sqcup D) \rightsquigarrow \neg C \sqcap \neg D \\
\neg(\exists R.C) & \rightsquigarrow & (\forall R.\neg C) & \neg(\forall R.C) \rightsquigarrow (\exists R.\neg C) \\
\neg(\forall U_1, U_2.r) & \rightsquigarrow & \bigsqcup_{r' \in \mathsf{Rel}, r' \neq r} \exists U_1, U_2.r' & \\
\neg(\exists u_1, u_2.r) & \rightsquigarrow & \bigsqcup_{r' \in \mathsf{Rel}, r' \neq r} \forall u_1, u_2.r' & \textit{where } u_i \textit{ are feature paths} \\
\neg(\exists R g_1, g_2.r) & \rightsquigarrow & (\forall R g^*, g_2. =) \sqcap \bigsqcup_{r' \in \mathsf{Rel}, r' \neq r} \forall R.(\forall g_1, g^*.r') & \\
& & \multicolumn{2}{r}{\textit{where } R \in \mathsf{N_{rR}} \textit{ and } g^* \textit{ is a fresh concrete feature}}
\end{array}$$

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

We now introduce path normal form for $\mathcal{ALC}(\mathcal{C})$-concepts and TBoxes. Path normal form was first considered in [15] in the context of the description logic $\mathcal{TDL}$ and in [12] in the context of $\mathbb{Q}$-$\mathcal{SHIQ}$.

Definition 16 (Path Normal Form). An $\mathcal{ALC}(\mathcal{C})$-concept C is in *path normal form (PNF)* iff it is in NNF and, for all subconcepts $\exists U_1, U_2.r$ and $\forall U_1, U_2.r$ of C, we have one of the following:

1. $U_1 = g_1$ and $U_2 = g_2$ for some $g_1, g_2 \in \mathsf{N_{cF}}$
2. $U_1 = Rg_1$ and $U_2 = g_2$ for some $R \in \mathsf{N_{aF}} \cup \mathsf{N_{rR}}$ and $g_1, g_2 \in \mathsf{N_{cF}}$
3. $U_1 = g_1$ and $U_2 = Rg_2$ for some $R \in \mathsf{N_{aF}} \cup \mathsf{N_{rR}}$ and $g_1, g_2 \in \mathsf{N_{cF}}$.

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

$\triangle$

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

Lemma 17. *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. For every feature path $u = f_1 \cdots f_n g$ used in C, we assume that $[g], [f_n g], \ldots, [f_1 \cdots f_n g]$ are concrete features not used in C. We inductively define a mapping λ from feature paths u in C to concepts as follows:

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

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

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

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

  where $\mathsf{r}^{\neq} \in \mathsf{Rel} \setminus \{=\}$
- and then replacing all subconcepts $\exists u_1, u_2.r$ with $\exists [u_1], [u_2].r \sqcap \lambda(u_1) \sqcap \lambda(u_2)$.

We extend the mapping ρ to TBoxes in the obvious way, i.e., if

$$\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_k \sqsubseteq D_k\},$$

then

$$\rho(\mathcal{T}) = \{\rho(C_1) \sqsubseteq \rho(D_1), \ldots, \rho(C_k) \sqsubseteq \rho(D_k)\}.$$

Now let C be a $\mathcal{ALC}(\mathcal{C})$-concept and $\mathcal{T}$ a $\mathcal{ALC}(\mathcal{C})$-TBox. Using the rewrite rules from Lemma 15, we can convert C into an equivalent concept C' in NNF and $\mathcal{T}$ into an equivalent TBox $\mathcal{T}'$ in NNF. It is then easy to check that C' is satisfiable w.r.t. a TBox $\mathcal{T}'$ iff $\rho(C')$ is satisfiable w.r.t. $\rho(\mathcal{T}')$. Moreover, $\rho(C')$ and $\rho(\mathcal{T}')$ are clearly in PNF and the translation can be done in polynomial time. ❏

Intuitively, Lemma 17 states that Variant (i) of the binary concrete domain constructors discussed in the previous section can be reduced to the forms $\exists f g_1, g_2.r$ and $\exists g_1, g_2.P$. Variant (ii) of the binary concrete domain constructors does not need to be manipulated in order to fit into the PNF scheme. In what follows, we generally assume that all concepts and TBoxes are in path normal form, as well as that constraint systems are ω-admissible. Moreover, we will often refer to TBoxes $\mathcal{T}$ in their *concept form* $C_\mathcal{T}$ which is defined as follows:

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

Finally, we will define the set of subconcepts of C_0 and $\mathcal{T}$ as:

$$\mathsf{sub}(C_0, \mathcal{T}) = \mathsf{sub}(C_0) \cup \mathsf{sub}(C_\mathcal{T})$$

Now we introduce the underlying data structure of the tableau algorithm:

Definition 18 (Completion system). Let $\mathsf{O_a}$ and $\mathsf{O_c}$ be disjoint and countably infinite sets of *abstract* and *concrete nodes*. A *completion tree* for an $\mathcal{ALC}(\mathcal{C})$-concept C and a Tbox $\mathcal{T}$ is a finite, labelled tree $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ with nodes $\mathsf{V_a} \cup \mathsf{V_c}$, such that $\mathsf{V_a} \subseteq \mathsf{O_a}$, $\mathsf{V_c} \subseteq \mathsf{O_c}$, and all nodes from $\mathsf{V_c}$ are leaves. The tree is labelled as follows:

1. each node $a \in \mathsf{V_a}$ is labelled with a subset $\mathcal{L}(a)$ of $\mathsf{sub}(C, \mathcal{T})$,
2. each edge $(a, b) \in E$ with $a, b \in \mathsf{V_a}$ is labelled with a role name $\mathcal{L}(a, b)$ occurring in C or $\mathcal{T}$;
3. each edge $(a, x) \in E$ with $a \in V_a$ and $x \in \mathsf{V_c}$ is labelled with a concrete feature $\mathcal{L}(a, x)$ occurring in C or $\mathcal{T}$.

A node $b \in \mathsf{V_a}$ is an R-successor of a node $a \in \mathsf{V_a}$ if $(a, b) \in E$ and $\mathcal{L}(a, b) = R$, while an $x \in \mathsf{V_c}$ is an 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 tuple $S = (T, \mathcal{N})$ where

- $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ is a completion tree for C and $\mathcal{T}$,
- $\mathcal{N}$ is a Rel-network with $V_{\mathcal{N}} = \mathsf{V_c}$.

△

To decide the satisfiability of an $\mathcal{ALC}(\mathcal{C})$-concept C_0 w.r.t. a TBox box $\mathcal{T}$ (both in PNF), the tableau algorithm is started with the initial completion system

$$S_{C_0} = (T_{C_0}, \emptyset)$$

with the initial completion tree

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

The algorithm applies completion rules to the completion system until an obvious inconsistency (clash) is detected or no completion rule is applicable any more. We will define completion rules for $\mathcal{ALC}(\mathcal{C})$ after some prerequisites. Let us first introduce an operation that is used by completion rules to add new nodes to completion trees. The operation respects the functionality of abstract and concrete features.

Definition 19 ($\oplus$ Operation)). An abstract or concrete node is called *fresh* w.r.t. a completion tree T if it does not appear in T. Let $S = (T, \mathcal{N})$ be a completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. We use the following operations:

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

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

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

△

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

- For $a \in \mathsf{V_a}$, we define:

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

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

Definition 20 (Blocking). Let $S = (T, \mathcal{N})$ be a completion system for a concept C_0 and a Tbox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. Let $a, b \in \mathsf{V_a}$. We say that $a \in \mathsf{V_a}$ is *potentially blocked by* b if the following holds:

- b is an ancestor of a in T,
- $\mathcal{L}(a) \subseteq \mathcal{L}(b)$,
- $\mathsf{cs}(a) = \mathsf{cs}(b)$.

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

- a is potentially blocked by b,
- $\mathcal{N}(a)$ and $\mathcal{N}(b)$ are complete, and
- $\mathcal{N}(a) = \mathcal{N}(b)$.

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

We are now ready to define the completion rules, which are given in Figure 3. If no completion rule is applicable to a completion system S, S is called *complete*.

All rules except $R\mathsf{net}$ and $R\mathsf{net}'$ are rather standard. The purpose of these additional rules is to resolve potential blocking situations into actual blocking situations or non-blocking situations by completing the parts of the network $\mathcal{N}$ that correspond to the "blocked" and "blocking" node. To ensure an appropriate interplay between $R\mathsf{net}/R\mathsf{net}'$, and the blocking condition and thus to guarantee termination, we apply these rules with highest precedence.

Note that the blocking mechanism obtained in this way is *dynamic* in the sense that blocking situations can be broken again after they have been established.

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

Figure 3: The Completion Rules.

Also note that the conditions $\mathcal{L}(a) \subseteq \mathcal{L}(b)$ and $\mathsf{cs}(a) = \mathsf{cs}(b)$ can be viewed as a refinement of pairwise blocking as known from [10]: due to path normal form, pairwise blocking is a strictly sharper condition than these two.

The algorithm applies completion rules until no more rules are applicable or a clash is encountered.

Definition 21 (Clash). Let $S = (T, \mathcal{N})$ be a completion system for a concept C and a Tbox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_a}, E, \mathcal{L})$. S is said to contain a *clash* iff

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

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

We present now the tableau algorithm in pseudo-code notation. It is started with $\mathsf{sat}(S_{C_0})$.

Algorithm:

```
procedure sat(S)
    if $S$ contains a clash then return unsatisfiable
    if is complete then return satisfiable
    if $R \in \{R\mathsf{net}, R\mathsf{net}'\}$ is applicable
        then $S' :=$ application of $R$ to $S$
        else $S' :=$ application of any applicable completion rule to $S$
    return $\mathsf{sat}(S')$
```

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

4.1 Correctness

Here we prove the termination, soundness and completeness of the presented tableau algorithm. Let us first introduce a few notions. With $|M|$ we 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, roles, and concrete features, respectively, that occur in a concept C_0 and a TBox $\mathcal{T}$.

Lemma 22 (Termination). *The tableau algorithm terminates for every input $\mathcal{ALC}(\mathcal{C})$-concept C_0 and input TBox $\mathcal{T}$, both in PNF.*

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 $S_i = (T_i, \mathcal{N}_i)$. Moreover, let $n = |\mathsf{sub}(C_0, \mathcal{T})|$. We first show the following:

(a) For all $i \geq 0$, the out-degree of T_i is bounded by n:
New nodes are created exclusively due to applications of the rules $R\exists$ and $R\exists_c$. The rule $R\exists$ generates at most one successor per node for each $\exists R.C \in \mathsf{sub}(C_0, \mathcal{T})$, and, since the concepts in the node labels are in PNF, the rule $R\exists_c$ generates at most one abstract successor per node for every $\exists U_1, U_2.r \in \mathsf{sub}(C_0, \mathcal{T})$. Moreover, $R\exists_c$ generates at most one concrete successor for every concrete feature appearing in some $\exists U_1, U_2.r \in \mathsf{sub}(C_0, \mathcal{T})$. Hence, the number of successors of a node is bounded by $n = |\mathsf{sub}(C_0, \mathcal{T})|$.

(b) For $i \geq 0$, the depth of T_i is bounded by $\ell = 2^{2n} \cdot |\mathsf{Rel}|^{n^2} + 2$:
Assume, to the contrary of what is to be shown, that there is an $i \in \mathbb{N}$ such that the depth of T_i exceeds ℓ. Moreover, let i be smallest with this

property. This means that T_i has been obtained from T_{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 $R\mathsf{net}$ and $R\mathsf{net}'$ are applied with highest precedence, the latter two rules are 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, we have that $\mathcal{N}_{i-1}(a)$ and $\mathcal{N}_{i-1}(b)$ are complete. Let us define a binary relation $\approx$ on $\mathsf{V_a}$ as follows:

$$a \approx b \qquad \text{iff} \qquad \mathcal{L}(a) = \mathcal{L}(b) \text{ and } \mathsf{cs}(a) = \mathsf{cs}(b) \text{ and } \mathcal{N}_{i-1}(a) = \mathcal{N}_{i-1}(b)$$

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 let $m = |\mathsf{N}_{\mathsf{cF}}^{C_0,\mathcal{T}}|$. Since $\mathcal{L}(a) \subseteq \mathsf{sub}(C_0, \mathcal{T})$, and $\mathcal{N}_{i-1}(a)$ is a complete Rel-network with $V_{\mathcal{N}_{i-1}(a)} = \mathsf{cs}(a)$, it is not difficult to verify that

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

Since $m \leq n$, we obtain that $|\mathsf{V_a}/_\approx| \leq 2^n \cdot 2^n \cdot |\mathsf{Rel}|^{n^2} = 2^{2n} \cdot |\mathsf{Rel}|^{n^2}$.

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 that $a_i \approx a_j$ 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.

The tableau algorithm terminates due to the following reasons:

- It constructs a finitely labelled completion tree T of a bounded out-degree and depth (by (a) and (b)) in a monotonic way, i.e., no nodes are removed from T, no concepts are removed from node labels, and no constraints are removed from $\mathcal{N}$;

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

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

❏

Lemma 23 (Soundness). *If the tableau algorithm returns* $\mathsf{satisfiable}$*, 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 $\mathcal{T}$. Let $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, and let $\mathsf{root} \in \mathsf{V_a}$ denote the root of T. Our aim is to define a model $\mathcal{I}$ for C_0 and $\mathcal{T}$. We proceed in several steps.

Let blocks be a function that for every directly blocked $b \in \mathsf{V_a}$, returns an unblocked $a \in \mathsf{V_a}$ such that b is blocked by a in S. It can easliy seen that, by definition of blocking, such 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 $1 \leq i < n$, b_{i+1} is a successor of a_i in T and

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

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

$$\begin{aligned}
\Delta_{\mathcal{I}} &:= \{p \in \mathsf{Paths} \mid p \text{ non-empty and } \mathsf{head}(p) = \frac{\mathsf{root}}{\mathsf{root}}\} \\
A^{\mathcal{I}} &:= \{p \in \Delta_{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a}{b} \text{ and } A \in \mathcal{L}(a)\},\ A \in \mathsf{N_C^{C_0,\mathcal{T}}} \\
R^{\mathcal{I}} &:= \{(p, p \cdot \frac{a}{b}) \in \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a'}{b'} \text{ and } b \text{ is } R\text{-successor of } a' \text{ in } T\ \},\ R \in \mathsf{N_R^{C_0,\mathcal{T}}}
\end{aligned}$$

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 $f \in \mathsf{N_{aF}}$, which is ensured by "$\oplus$" operation which generates at most one f-successor per abstract node, and by definition of Paths, where, in the case of several nodes blocking the same node, only one of them is chosen and put into a path.

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

- p is called a *hook* if $p = \frac{\mathsf{root}}{\mathsf{root}}$ or $\mathsf{tail}(p) = \frac{a}{b}$ with $a \neq b$ (and thus b blocked by a). We use Hooks to denote the set of all hooks.

- we call p a *q-companion* if q is a hook and there exists $q' \in \mathsf{Paths}$ such that $p = qq'$ and all nodes $\frac{a}{b}$ in q' satisfy $a = b$, with the possible exception of $\mathsf{tail}(q')$.

Intuitively, the hooks, which are induced by blocking situations in T, are the points where we patch together parts of T. The part of T patched at a hook p with $\mathsf{tail}(p) = \frac{a}{b}$ is comprised of (copies of) all the nodes c in T that are reachable from a, except indirectly blocked ones. Formally, the part of $\mathcal{I}$ 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 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 technical results. Since S is clash-free, $\mathcal{N}$ is satisfiable. Therefore, 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). Note that this naming scheme is well-defined since the "$\oplus$" operation ensures that every abstract node a has at most one g-successor, for every $g \in \mathsf{N_{cF}}$. 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:

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

It is not hard to verify that $V_{\mathbf{N}} = \{(p, g) \mid p \in \mathsf{Paths},\ \mathsf{tail}(p) = \frac{a}{b},\ g \in \mathsf{cs}(a)\}$: this is an easy consequence of the definition of $\mathbf{N}$, the naming scheme used in $\mathcal{N}^c$, and the fact that the blocking condition ensures $\mathsf{cs}(a) = \mathsf{cs}(b)$ if a is blocked by b.

We have to show that $\mathbf{N}$ is satisfiable. To this end, we first show that $\mathbf{N}$ is patched together from (copies of) parts of $\mathcal{N}$: every hook p gives rise to a part of $\mathbf{N}$ as follows:

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

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 $N(p)|_{V_{N(p)} \cap V_{N(q)}} = N(q)|_{V_{N(p)} \cap V_{N(q)}}$;

(d) for every $p \in \mathsf{Hooks}$, $N(p)$ is finite, complete, and satisfiable.

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 $q^* \in P(q) \cap P(q')$, 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 $q^* = pq'$ and (ii), 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 $p, q \in \mathsf{Hooks}$.

(d) Let $p \in \mathsf{Hooks}$. First for finiteness of $N(p)$. Since the completion tree T is finite, for every hook p we have that $P(p)$ is finite. Moreover, for every $q \in P(p)$, the set $\{(q,g) \mid g \in \mathsf{N}_{\mathsf{cF}}^{C_0,\mathcal{T}}\} \supseteq V_{N(p)}$ is clearly finite.

We now 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 $(q,g), (q',g') \in V_{N(p)}$. Then $q, q' \in P(p)$ and $\mathsf{rep}_p(q)$, $\mathsf{rep}_p(q')$ are defined. Since $N(p) \subseteq \mathbf{N}$, the variables $(q,g), (q',g')$ occur in $\mathbf{N}$ and, by definition of $\mathbf{N}$, $(\mathsf{rep}_p(q), g)$ and $(\mathsf{rep}_p(q'), g')$ occur in $\mathcal{N}^c$. 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 $\mathbf{N}$, it follows that $((q,g)\, r\, (q',g')) \in \mathbf{N}$. By definition of $N(p)$, $((q,g)\, r\, (q',g')) \in N(p)$.

For (ii), assume that $((q,g)\, r\, (q',g')) \in N(p)$, for each $r \in \{r_1, r_2\}$. Since $N(p) \subseteq \mathbf{N}$, it follows that $((q,g)\, r\, (q',g')) \in \mathbf{N}$, for each $r \in \{r_1, r_2\}$. Thus, there are $q_1, q_2 \in \mathsf{Hooks}$ such that $q, q' \in P(q_i)$ and $((\mathsf{rep}_{q_i}(q), g)\, r_i\, (\mathsf{rep}_{q_i}(q'), g') \in \mathcal{N}^c$, for each $i \in \{1, 2\}$. We distinguish two cases:

- $q_1 = q_2$. Then completeness of $\mathcal{N}^c$ implies that $r_1 = r_2$.
- $q_1 \neq q_2$. By definition of q-companions and since $q, q' \in P(q_1) \cap P(q_2)$, this is only possible if $q = q' = q_i$ and q_i is a successor-hook of $q_{\bar{i}}$, for some $i \in \{1, 2\}$, where $\bar{i}$ denotes 2 if $i = 1$ and 1 if $i = 2$. W.l.o.g., assume that $i = 1$. Let $\mathsf{tail}(q) = \frac{a}{b}$. Since $q = q_1$ and q_1 is a hook, we have $a \neq b$, and thus b is blocked by a in T. By definition of rep, we have $\mathsf{rep}_{q_2}(q) = b$ and $\mathsf{rep}_{q_1}(q) = a$. Thus, $((a, g)\, r_1\, (a, g')) \in \mathcal{N}^c$ and $((b, g)\, r_2\, (b, g')) \in \mathcal{N}^c$. Since b is blocked by a, the blocking condition implies $r_1 = r_2$.

Finally, we show satisfiability of $N(p)$. To this end, it suffices to prove that $((q, g)\, r\, (q', g')) \in N(p)$ implies $((\mathsf{rep}_p(q), g)\, r\, (\mathsf{rep}_p(q'), g') \in \mathcal{N}^c$: together with the satisfiability of $\mathcal{N}^c$, this yields satisfiability of $N(p)$. Thus, let $((q, g)\, r\, (q', g')) \in N(p)$. Since $N(p) \subseteq \mathbf{N}$, the definition of $\mathbf{N}$ yields $q, q' \in P(p)$ and, as required, $((\mathsf{rep}_p(q), g)\, r\, (\mathsf{rep}_p(q'), g') \in \mathcal{N}^c$.

Claim 2: $\mathbf{N}$ is satisfiable.

Proof: We distinguish two cases:

(1) There are no blocked nodes in S. In that case, $\mathsf{Hooks} = \{\frac{\mathsf{root}}{\mathsf{root}}\}$. Then, by Claim 1(a), we have that $\mathbf{N} = N(\frac{\mathsf{root}}{\mathsf{root}})$, and by Claim 1(d) we obtain that $\mathbf{N}$ is satisfiable.

(2) There are blocked nodes in S. In that case, since $\mathsf{V_a}$ is finite (c.f. Lemma 22), 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 out-degree of the completion tree T, and thus finite by what was shown in the proof of Lemma 22. 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 $i > j$.

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

- $k = 0$: $\mathbf{N}_0 = N(p_0)$ is satisfiable by Claim 1(d).
- $k = n + 1,\ n \geq 0$: We have that $\mathbf{N}_{n+1} = \mathbf{N}_n \cup N(p_{n+1})$. By induction, $\mathbf{N}_n$ is satisfiable. Let $\mathbf{N}_n^c$ be a completion of $\mathbf{N}_n$ and let $\mathbf{N}'_{n+1} = \mathbf{N}_n^c \cup N(p_{n+1})$. There exists a unique $p_l \in \mathsf{Hooks}$, $l \leq n$, such that p_{n+1} is a successor of p_l. By the definition of $\mathbf{N}_n$ and Claim 1(b), and since $V_{\mathbf{N}_n^c} = V_{\mathbf{N}_n}$, we have that $V_{\mathbf{N}_n^c} \cap V_{N(p_{n+1})} = V_{N(p_l)} \cap V_{N(p_{n+1})}$. Moreover, by Claim 1(d), $N(p_l)$

is complete, and thus, $\mathbf{N}^c_n|_{V_{N(p_l)} \cap V_{N(p_{n+1})}} = N(p_l)|_{V_{N(p_l)} \cap V_{N(p_{n+1})}}$. Thus, we obtain:

$$\begin{aligned}
\mathbf{N}^c_n|_{V_{\mathbf{N}^c_n} \cap V_{N(p_{n+1})}} &= \mathbf{N}^c_n|_{V_{N(p_l)} \cap V_{N(p_{n+1})}} \\
&= N(p_l)|_{V_{N(p_l)} \cap V_{N(p_{n+1})}} \\
&= N(p_{n+1})|_{V_{N(p_l)} \cap V_{N(p_{n+1})}}
\end{aligned}$$

Here, the last equality follows from Claim 1(c). Moreover, by Claim 1(d), we have that $N(p_{n+1})$ is finite, complete and satisfiable, and $\mathbf{N}^c_n$ is finite. Thus, $\mathbf{N}^c_n$ and and $N(p_{n+1})$ are finite complete satisfiable networks, whose intersection parts are identical. Since the completion system $\mathcal{C}$ has the patchwork property, we obtain that $\mathbf{N}'_{n+1}$ is satisfiable, and, since $\mathbf{N}_{n+1} \subseteq \mathbf{N}'_{n+1}$, we have that $\mathbf{N}_{n+1}$ is satisfiable.

Thus, we have shown that $\mathbf{N}_k$ is satisfiable for all $k \geq 0$.

To the contrary to what is to be shown, let us now assume that $\mathbf{N}$ is not satisfiable. We consider two cases:

(i) $\mathbf{N}$ is finite. Then, there exists an $m \in \mathbb{N}$, such that $\mathbf{N} = \mathbf{N}_m$. Hence, we have that $\mathbf{N}_m$ is unsatisfiable, which is a contradiction.

(ii) $\mathbf{N}$ is infinite. Since $\mathcal{C}$ has the compactness property, we have that $\mathbf{N}$ has a finite sub-network M such that M is unsatisfiable. But then there exists an $m \in \mathbb{N}$ such that $\mathbf{N}_m \supseteq M$, and thus $\mathbf{N}_m$ is unsatisfiable, which is a contradiction.

This finishes the proof of Claim 2. We are now ready to define the concrete part of the model $\mathcal{I}$. Since $\mathbf{N}$ is a satisfiable, there is an $M_\mathcal{I} \in \mathfrak{M}$ and a mapping $\tau : V_\mathbf{N} \to V_{M_\mathcal{I}}$ such that $(v\ r\ v') \in \mathbf{N}$ implies $(\tau(v)\ r\ \tau(v')) \in V_{M_\mathcal{I}}$. 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:

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

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

- C is a concept name. By construction of $\mathcal{I}$.

- $C = \neg D$. 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 $s \in (\neg D)^{\mathcal{I}}$.

- $C = D \sqcap E$. 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 $s \in (D \sqcap E)^{\mathcal{I}}$.

- $C = D \sqcup E$. 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 $s \in (D \sqcup E)^{\mathcal{I}}$.

- $C = \exists R.D$. 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 $\mathcal{L}(c) \subseteq \mathcal{L}(d)$ (both if $c = d$ and if c is blocked by d), 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 $s \in C^{\mathcal{I}}$.

- $C = \forall R.D$. 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 previuos 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 $s \in C^{\mathcal{I}}$.

- $C = \exists U_1, U_2.r$. As C is in PNF, it suffices to consider the case $U_1 = Rg_1$, $U_2 = g_2$ (the symmetric case and the one when U_i are concrete features are analogus or easier). Since the $R\exists_c$ rule is not applicable, there exists an R-successor c of a, a g_1-successor y_1 of c, and a g_2-successor y_2 of a such that $(y_1 \; r \; y_2) \in \mathcal{N}$. Then $((c, g_1) \; r \; (a, g_2)) \in \mathcal{N}^c$. Moreover, there is a $t = s \cdot \frac{d}{c} \in \Delta_{\mathcal{I}}$ such that $c = d$ or c is blocked by d. By definition of $R^{\mathcal{I}}$, we have that $(s, t) \in R^{\mathcal{I}}$. Moreover, there is a $p \in \mathsf{Hooks}$ such that s and t are p-companions and $\mathsf{rep}_p(s) = a$, $\mathsf{rep}_p(t) = c$. Thus, by definition of $\mathbf{N}$ we obtain $((t, g_1) \; r \; (s, g_2)) \in \mathbf{N}$, implying $(\tau(t, g_1) \; r \; \tau(s, g_2)) \in M_{\mathcal{I}}$. Since $g_1^{\mathcal{I}}(t) = \tau(t, g_1)$ and $g_2^{\mathcal{I}}(s) = \tau(s, g_2)$, we obtain that $s \in C^{\mathcal{I}}$.

- $C = \forall U_1, U_2.r$. As in the previous case, we will assume that U_1 and U_2 are of the form $U_1 = Rg_1$, $U_2 = g_2$. Let t, x_1, x_2 be such that $(s, t) \in R^{\mathcal{I}}$, $g_1^{\mathcal{I}}(s) = x_1 = \tau(s, g_1)$, and $g_2^{\mathcal{I}}(s) = x_2 = \tau(t, g_2)$. By definition of $\mathcal{I}$, we have that $t = s \cdot \frac{d}{c} \in \Delta_{\mathcal{I}}$ such that c is an R-successor of a. Moreover, there is a g_1-successor y_1 of c and a g_2-successor y_2 of a. Since $R\forall_c$ is inapplicable, $\forall U_1, U_2.r \in \mathcal{L}(a)$ implies that $(y_1 \; r \; y_2) \in \mathcal{N}$. Thus, $((c, g_1) \; r \; (a, g_2)) \in \mathcal{N}^c$. Moreover, there is a $p \in \mathsf{Hooks}$ such that s and t are p-companions and $\mathsf{rep}_p(s) = a$, $\mathsf{rep}_p(t) = c$. Thus, by definition of $\mathbf{N}$, $((t, g_1) \; r \; (s, g_2)) \in \mathbf{N}$, which implies $(\tau(t, g_1) \; r \; \tau(s, g_2)) \in M_{\mathcal{I}}$.

This finishes the proof of Claim 3. Since $C_0 \in \mathcal{L}(\mathsf{root})$ and $\frac{\mathsf{root}}{\mathsf{root}} \in \Delta_\mathcal{I}$, Claim 3 implies that $\mathcal{I}$ is a model of C_0.

Finally, let us show that $\mathcal{I}$ is a model of the input TBox $\mathcal{T}$. Let $C \sqsubseteq D \in \mathcal{T}$ and $s \in \Delta_\mathcal{I}$. Let $\mathsf{tail}(s) = \frac{a}{b}$. Since S is complete, $R\mathsf{gci}$ is not applicable, and thus $C_\mathcal{T} \in \mathcal{L}(a)$. By Claim 3 we have that $s \in C_\mathcal{T}^\mathcal{I}$, and, consequently, $s \in \bigcap_{C \sqsubseteq D \in \mathcal{T}} (C \to D)^\mathcal{I}$. Thus, we obtain $C^\mathcal{I}(s) \subseteq D^\mathcal{I}(s)$. ❏

Lemma 24 (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) $(x\, r\, y) \in \mathcal{N} \Rightarrow (\tau(x)\, r\, \tau(y)) \in M_\mathcal{I}$

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 $b \in C^\mathcal{I}$. 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) re-uses 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 $\exists U_1, U_2.r \in \mathcal{L}(a)$. We assume that $U_1 = Rg_1$ and $U_2 = g_2$. The symmetric case is analogous and the case where one or both of U_1, U_2 are only concrete features is similar, but easier. The rule application generates a new abstract node b and concrete nodes x and y (or re-uses existing ones and renames them) such that:

- b is an R-successor of a,
- x is a g_1-successor of b, and
- y is a g_2-successor of a

and adds $(x\, r\, y)$ to $\mathcal{N}$. By (Ca), we have $\pi(a) \in (\exists U_1, U_2.r)^{\mathcal{I}}$. Thus, there exist $d \in \Delta_{\mathcal{I}}$ and $z, w \in V_{M_{\mathcal{I}}}$ such that:

- $(\pi(a), d) \in R^{\mathcal{I}}$,
- $g_1^{\mathcal{I}}(d) = z$,
- $g_2^{\mathcal{I}}(\pi(a)) = w$, and
- $(z\, r\, w) \in M_{\mathcal{I}}$

Extend π by setting $\pi(b) := d$, and extend τ by setting $\tau(x) := z$ and $\tau(y) := w$. It is easily seen that the resulting completion system is $\mathcal{I}$-compatible via the extended π and τ.

$R\forall_c$ The rule application is applied to an abstract node a with $\forall U_1, U_2.r \in \mathcal{L}(a)$ such that there are $x_1, x_2 \in \mathsf{V_c}$ such that x_i if a U_i-successor of a for $i = 1, 2$. It adds $(x_1\ r\ x_2)$ to $\mathcal{N}$. By (Ca), $\pi(a) \in (\forall U_1, U_2.r)^{\mathcal{I}}$. 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 $(\tau(x_1)\, r\, \tau(x_2)) \in M_{\mathcal{I}}$. Thus, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

Rnet The rule application guesses a completion $\mathcal{N}'$ of $\mathcal{N}'(a)$, for some $a \in \mathsf{V_a}$, and sets $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$. Define

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

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.

Rnet$'$ Analogously to the previous case.

Rgci 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 22, 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 22, 23 and 24, we get the following theorem:

Theorem 25. *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 Conclusion

We have proved decidability of $\mathcal{ALC}$ with ω-admissible constraint systems and GCIs. We conjecture that, by mixing the techniques from the current paper with those from [15, 12], it is possible to prove ExpTime-completeness of satisfiability in $\mathcal{ALC}(\mathcal{C})$ provided that satisfiability in $\mathcal{C}$ can be decided in ExpTime. Various language extensions, both on the logical and concrete side, should also be possible in a straightforward way.

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

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

However, a complication is posed by the fact that path normal form introduces additional concrete features. Simply requiring, as an additional restriction, that only concepts and TBoxes in PNF are allowed is rather severe: it can be seen that, then, satisfiability in $\mathcal{ALC}(\mathcal{C})$ instantiated with the RCC8 and Allen constraint systems can be decided by adding some simple clash conditions. In particular, there is no need to use an external reasoner for the constraint system at all. It is more interesting to find a tableau algorithm for $\mathcal{ALC}(\mathcal{C})$ with only one concrete feature that does not rely on PNF. As future work, we will try to identify such an algorithm. We believe that, in this, it is neither necessary to add any non-determinism apart from the $R\sqcup$ rule, nor to perform a *global* satisfiability check of $\mathcal{N}$.

References

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

[2] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.

[3] P. Balbiani and J.-F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *Frontiers of Combining Systems (FroCoS 2002)*, number 2309 in LNAI, pages 162–176. Springer, 2002.

[4] B. Bennett. Modal logics for qualitative spatial reasoning. *Journal of the Interest Group in Pure and Applied Logic*, 4(1), 1997.

[5] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 303–307, 1996.

[6] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[7] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Number 148 in Studies in Logic and the Foundations of Mathematics. Elsevier, 2003.

[8] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 701–705. Springer-Verlag, 2001.

[9] I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR98)*, pages 636–647, 1998.

[10] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[11] C. Lutz. Complexity of terminological reasoning revisited. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 181–200. Springer-Verlag, 1999.

[12] C. Lutz. 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)*, pages 191–202. Morgan Kaufman, 2002.

[13] C. Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. In I. Horrocks and S. Tessaris, editors, *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*, number 53 in CEUR-WS (http://ceur-ws.org/), pages 185–194, 2002.

[14] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003.

[15] C. Lutz. Combining interval-based temporal reasoning with general tboxes. *Artificial Intelligence*, 152(2):235–274, 2004.

[16] C. Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[17] C. Lutz and F. Wolter. Modal logics of topological relations. In *Proceedings of Advances in Modal Logics 2004*, 2004.

[18] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. Morgan Kaufman, 1992.

[19] J. Renz and B. Nebel. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1–2):69–123, 1999.

[20] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: a revised report. In *Readings in qualitative reasoning about physical systems*, pages 373–381. Morgan Kaufmann, San Francisco, CA, USA, 1990.