Adding Numbers to the $\mathcal{SHIQ}$ Description Logic—First Results

Carsten Lutz
LuFG Theoretical Computer Science, RWTH Aachen
Ahornstr. 55, 52074 Aachen, Germany

Abstract

Recently, the Description Logic (DL) $\mathcal{SHIQ}$ has found a large number of applications. This success is due to the fact that $\mathcal{SHIQ}$ combines a rich expressivity with efficient reasoning. One weakness of $\mathcal{SHIQ}$, however, limits its usability in several application areas: numerical knowledge such as knowledge about the age, weight, or temperature of real-world entities cannot be adequately represented. In this paper, we present $\mathbb{Q}$-$\mathcal{SHIQ}$, an extension of $\mathcal{SHIQ}$ that aims at closing this gap, and show that reasoning with the extended DL is ExpTime-complete.

1 Motivation

Description Logics (DLs) are a family of knowledge representation formalisms, which are—apart from their classical application in KR—nowadays used in various application areas such as reasoning about entity relationship (ER) diagrams and providing a formal basis for the so-called semantic web [4; 6]. One of the most influential DLs proposed during the last years is the $\mathcal{SHIQ}$ Description Logic, whose success is based mainly on the following two facts: first, $\mathcal{SHIQ}$ is a very expressive DL providing for, e.g., transitive roles, inverse roles, and number restrictions, but its reasoning problems are nevertheless decidable in ExpTime [13; 22]. Second, $\mathcal{SHIQ}$ has been implemented in efficient DL systems such as FaCT and RACER, which can, despite the high worst-case complexity of reasoning with $\mathcal{SHIQ}$, deal surprisingly well even with huge knowledge bases [10; 7].

Although, as we just argued, $\mathcal{SHIQ}$'s expressive power is one of the main reasons for its success, there is still room for improvement. In particular, $\mathcal{SHIQ}$ cannot adequately represent numerical knowledge such as knowledge about the age, weight, or temperature of real-world entities, which, as we will later discuss in more detail, is crucial for many important applications [2; 12; 6; 17]. In this paper, we extend $\mathcal{SHIQ}$ with a set of concept constructors that belong to the so-called concrete domain family of constructors and allow a straightforward representation of numerical knowledge. Let us view a concrete example of knowledge representation with the resulting DL, which is called $\mathbb{Q}$-$\mathcal{SHIQ}$: the concept

$$\mathsf{Grandfather} \sqcap \exists \mathsf{age}.{=_{91}} \sqcap (\geqslant 20\ \mathsf{relatives}\ \mathsf{Human}) \sqcap \forall \mathsf{relatives}\ \mathsf{age}, \mathsf{age}.{<}$$

describes Grandfathers who are 91 years old, have at least 20 relatives (such constraints are called "qualified number restrictions"), and are older than all of these relatives. Note that we can refer to rational numbers such as "91" and also compare numbers using predicates such as "<". We argue that the additional expressivity provided by $\mathbb{Q}$-$\mathcal{SHIQ}$ is rather useful in many application areas. Let us briefly review three examples:

(1) As described in [5; 4], reasoning about ER diagrams is an important application area of Description Logics. One shortcoming of the standard way to encode ER diagrams, which is to use a fragment of the $\mathcal{SHIQ}$ Description Logic, can be described as follows: ER diagrams make use of so-called attributes to represent non-relational data such as numbers and strings to be stored in the database. If $\mathcal{SHIQ}$ is used for representing ER diagrams, constraints concerning the values of attributes cannot be expressed. To give a simple example, if there exists an entity Employee having two attributes Birthday and Employment-date, then it cannot be expressed that employees should be born before they are hired. If $\mathbb{Q}$-$\mathcal{SHIQ}$ is used for representing ER diagrams, such *numerical* data constraints on attributes can easily be handled. This topic is discussed in more detail in [19].

(2) In [17; 16], the Description Logic $\mathcal{TDL}$ is motivated as a valuable tool for the representation of temporal conceptual knowledge. $\mathcal{TDL}$ can be obtained from the well-known DL $\mathcal{ALC}$ [20] by adding general TBoxes and concrete domain style concept constructors that allow to represent relations between rational numbers such as "=" and "<". Indeed, it is not hard to see that $\mathcal{TDL}$ is a proper fragment of $\mathbb{Q}$-$\mathcal{SHIQ}$. Thus, $\mathbb{Q}$-$\mathcal{SHIQ}$ is also well-suited for reasoning about temporal conceptual knowledge as described in [17; 16]. Moreover, $\mathbb{Q}$-$\mathcal{SHIQ}$ significantly extends the expressive power provided by $\mathcal{TDL}$, even in the temporal/numerical component of the logic. For example, if $\mathbb{Q}$-$\mathcal{SHIQ}$ is used for temporal reasoning, then one can refer to *concrete* time points and time intervals such as 4 or $[1, 12]$. This is not possible in $\mathcal{TDL}$.

(3) A rapidly developing application area of DLs is their use as an ontology language for the semantic web [6]. As noted in [6; 9], the representation of "concrete datatypes" such as numbers is an important task in this context. However, in DLs such as OIL and DAML+OIL, which have been proposed in this application area, appropriate expressivity is either not provided or not taken into account for reasoning, which is done by a translation into $\mathcal{SHIQ}$ or related DLs. In [12], Horrocks and Sattler propose to extend $\mathcal{SHOQ}$, a close relative of $\mathcal{SHIQ}$, with so-called unary concrete domains in order to integrate concrete datatypes. However, this solution is not really satisfying since, as is explained in more detail in [16], unary concrete domains are of very limited expressivity. If $\mathbb{Q}$-$\mathcal{SHIQ}$ is used as the target logic in translations of OIL and DAML+OIL, a rather powerful means for describing *numerical* concrete datatypes becomes available.

As the main result of this paper, we prove reasoning with $\mathbb{Q}$-$\mathcal{SHIQ}$ to be decidable in ExpTime by devising an automata-based decision procedure. Thus, $\mathbb{Q}$-$\mathcal{SHIQ}$ sensibly enhances the expressive power of $\mathcal{SHIQ}$ without increasing the worst-case complexity of reasoning. This paper is accompanied by a technical report that contains more details and full proofs [14].

2 Syntax and Semantics

In this section, we introduce the Description Logic $\mathbb{Q}$-$\mathcal{SHIQ}$ in detail. We first give the syntax and semantics of $\mathbb{Q}$-$\mathcal{SHIQ}$-roles, then introduce some useful abbreviations, and finally define syntax and semantics of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts.

Definition 1. Let $\mathsf{N_{rR}}$, $\mathsf{N_{tR}}$, and $\mathsf{N_{aF}}$ be countably infinite and mutually disjoint sets of *regular role names*, *transitive role names*, and *abstract features*, respectively. Moreover, let $\mathsf{N_R} = \mathsf{N_{rR}} \uplus \mathsf{N_{tR}} \uplus \mathsf{N_{aF}}$. The set of *$\mathbb{Q}$-$\mathcal{SHIQ}$-roles* ROL is $\mathsf{N_R} \cup \{R^- \mid R \in \mathsf{N_R}\}$. A *role inclusion* is of the form $R \sqsubseteq S$, for $R, S \in \mathsf{ROL}$. A *role hierarchy* is a set of role inclusions.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$ which maps every role $R \in \mathsf{ROL}$ to a subset $R^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that, for $R \in \mathsf{N_R}$, $S \in \mathsf{N_{tR}}$, and $f \in \mathsf{N_{aF}}$, we have

- $(x, y) \in R^\mathcal{I}$ iff $(y, x) \in R^{-\mathcal{I}}$,
- if $(x, y) \in S^\mathcal{I}$ and $(y, z) \in S^\mathcal{I}$, then $(x, z) \in S^\mathcal{I}$,
- $f^\mathcal{I}$ is functional.

An interpretation $\mathcal{I}$ is a *model* of a role hierarchy $\mathcal{R}$ iff $R^\mathcal{I} \subseteq S^\mathcal{I}$ for each $R \sqsubseteq S \in \mathcal{R}$.

We introduce some notation to make the following considerations easier:

(1) The function Inv yields the inverse of a role. More precisely, for $R \in \mathsf{ROL}$, we set

$$\mathsf{Inv}(R) := \begin{cases} R^- & \text{if } R \text{ is a role name,} \\ S & \text{if } R = S^- \text{ for a role name } S. \end{cases}$$

(2) Since set inclusion is transitive and $R^\mathcal{I} \subseteq S^\mathcal{I}$ implies $\mathsf{Inv}(R)^\mathcal{I} \subseteq \mathsf{Inv}(S)^\mathcal{I}$, for a role hierarchy $\mathcal{R}$, we introduce $\underline{\circledast}_\mathcal{R}$ as the reflexive-transitive closure of $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$.

(3) We call a role $R \in \mathsf{ROL}$ *transitive* with respect to a role hierarchy $\mathcal{R}$ iff R is interpreted in a transitive relation in every model of $\mathcal{R}$. It is not hard to see that this is the case iff the following predicate evaluates to true:

$$\mathsf{Trans}_\mathcal{R}(R) := \begin{cases} \mathsf{true} & \text{if there exists a role } S \in \mathsf{N_{tR}} \\ & \text{s.t. } S' \underline{\circledast}_\mathcal{R} R \text{ and } R \underline{\circledast}_\mathcal{R} S'' \\ & \text{for some } S', S'' \in \{S, \mathsf{Inv}(S)\} \\ \mathsf{false} & \text{otherwise.} \end{cases}$$

(4) A role $R \in \mathsf{ROL}$ is called *simple* with respect to a role hierarchy $\mathcal{R}$ iff $\mathsf{Trans}_\mathcal{R}(S)$ does not hold for any $S \in \mathsf{ROL}$ with $S \underline{\circledast} R$.

For both "$\underline{\circledast}_\mathcal{R}$" and $\mathsf{Trans}_\mathcal{R}$, we omit the index if clear from the context. Note that no transitive role is simple since $\underline{\circledast}$ is defined as the *reflexive*-transitive closure. For the same reason, we have $\mathsf{Trans}(R)$ for all $R \in \mathsf{N_{tR}}$. However, roles must obviously not be in $\mathsf{N_{tR}}$ in order to be transitive. For example, if $R \in \mathsf{N_{tR}}$, then R^- is also transitive. Similarly, if $S \in \mathsf{N_{tR}}$, $R \notin \mathsf{N_{tR}}$, $S^- \sqsubseteq R$, $R \sqsubseteq S^-$, then R is transitive.

$$
\begin{array}{rcl}
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \quad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
(\exists R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{There is some } y \in \Delta^{\mathcal{I}} \text{ with } (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}, \\
(\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{For all } y \in \Delta^{\mathcal{I}}, \text{ if } (x,y) \in R^{\mathcal{I}}, \text{ then } y \in C^{\mathcal{I}}\}, \\
(\leqslant n\ R\ C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leqslant n\}, \\
(\geqslant n\ R\ C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geqslant n\}, \\
(\exists U_1, U_2.P)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{There are } q_1 \in U_1^{\mathcal{I}} \text{ and } q_2 \in U_2^{\mathcal{I}} \text{ with } q_1\, P\, q_2\} \\
(\forall U_1, U_2.P)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \text{For all } q_1 \in U_1^{\mathcal{I}} \text{ and } q_2 \in U_2^{\mathcal{I}}, \text{ we have } q_1\, P\, q_2\} \\
(\exists g.P_q)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid g^{\mathcal{I}}(x) \text{ is defined and } g^{\mathcal{I}}(x)\ P\ q\}
\end{array}
$$

Figure 1: $\mathbb{Q}$-$\mathcal{SHIQ}$ concept semantics.

We are now ready to define $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts and their semantics.

Definition 2. Let $\mathsf{N_C}$ and $\mathsf{N_{cF}}$ be countably infinite sets of *concept names* and *concrete features*, respectively, such that $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_{cF}}$ are mutually disjoint. A *path* is a sequence $R_1 \cdots R_k g$ consisting of roles $R_1, \dots, R_k \in \mathsf{ROL}$ and a concrete feature $g \in \mathsf{N_{cF}}$. A path $R_1 \cdots R_k g$ in which $R_1, \dots, R_k$ are abstract features (i.e., from $\mathsf{N_{aF}}$) is called *feature path.* The set of $\mathbb{Q}$-$\mathcal{SHIQ}$*-concepts* is the smallest set such that

1. every concept name $C \in \mathsf{N_C}$ is a concept,
2. if C and D are concepts and $R \in \mathsf{ROL}$, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, and $\exists R.C$ are concepts,
3. if C is a concept, $R \in \mathsf{ROL}$ is simple, and $n \in \mathbb{N}$, then $(\leqslant n\ R\ C)$ and $(\geqslant n\ R\ C)$ are concepts,
4. if u_1 and u_2 are feature paths and P is a predicate from the set $\{<, \leq, =, \neq, \geq, >\}$, then $\exists u_1, u_2.P$ and $\forall u_1, u_2.P$ are concepts,
5. if $R \in \mathsf{ROL}$ is simple, g_1 and g_2 are concrete features, and $P \in \{<, \leq, =, \neq, \geq, >\}$, then $\exists R g_1, g_2.P$ and $\forall R g_1, g_2.P$ are concepts, and
6. if g is a concrete feature, $P \in \{<, \leq, =, \neq, \geq, >\}$, and $q \in \mathbb{Q}$, then $\exists g.P_q$ is a concept.

We use $\top$ as an abbreviation for $A \sqcup \neg A$ (for some fixed $A \in \mathsf{N_C}$). The interpretation function $\cdot^{\mathcal{I}}$ of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept C to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and every concrete feature g to a partial function $g^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to the set of rational numbers $\mathbb{Q}$ such that the equations in Figure 1 are satisfied, where U_1 and U_2 denote paths, $\sharp S$ denotes the cardinality of the set S, and, for every path $U = R_1 \cdots R_k g$, $U^{\mathcal{I}}$ is defined as

$$\{(x,q) \subseteq \Delta_{\mathcal{I}} \times \mathbb{Q} \mid \exists y_1, \dots, y_{k+1} : x = y_1, \\ (y_i, y_{i+1}) \in R_i^{\mathcal{I}} \text{ for } 1 \leq i \leq k, \text{ and } g^{\mathcal{I}}(y_{k+1}) = q\}.$$

An interpretation $\mathcal{I}$ is a *model* of a concept C iff $C^{\mathcal{I}} \neq \emptyset$. C is called *satisfiable with respect to a role hierarchy* $\mathcal{R}$ iff there exists a model of C and $\mathcal{R}$. A concept D *subsumes* a concept C with respect to $\mathcal{R}$ (written $C \sqsubseteq_{\mathcal{R}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{R}$.

Throughout this paper, we denote concept names by A and B, concepts by C, D, and E, roles by P, R, and S, abstract features by f, concrete features by g, paths by U, feature paths by u, and predicates by P.

In the following sections, we show that $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability is decidable in deterministic exponential time. This also yields decidability and an ExpTime upper complexity bound for concept subsumption: we have $C \sqsubseteq_{\mathcal{R}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{R}$.

Most modern Description Logics do not only consist of a concept language but also provide for a TBox component. Formally, a TBox is a finite set of concept equations $C \doteq D$, where C and D are concepts, and an interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all $(C \doteq D) \in \mathcal{T}$. In the presence of TBoxes, one is usually interested in the satisfiability of concepts w.r.t. TBoxes and role hierarchies, i.e., in whether there exists a model $\mathcal{I}$ of C, $\mathcal{T}$, and $\mathcal{R}$. However, as shown in [11], in the presence of role hierarchies and transitive roles, it is possible to polynomially reduce concept satisfiability w.r.t. TBoxes and role hierarchies to concept satisfiability w.r.t. role hierarchies, only. Hence, we do not explicitly consider TBoxes in what follows.

Let us discuss the $\mathbb{Q}$-$\mathcal{SHIQ}$-concept language in some more detail. Since exhaustive information on $\mathcal{SHIQ}$ can be found in, e.g., [13], we concentrate on the additional concept constructors $\exists U_1, U_2.P$, $\forall U_1, U_2.P$, and $\exists g.P_q$, which, as has already been noted, are often called "concrete domain constructors". Concrete domains have been introduced by Baader and Hanschke as a means for representing "concrete knowledge" such as knowledge about numbers, strings, and spatial extensions [1]. More precisely, Baader and Hanschke extend the basic propositionally closed DL $\mathcal{ALC}$ with concrete domains, where a concrete domain $\mathcal{D}$ is comprised of a set called the domain and a set of predicates with a fixed extension on this domain. It is important to note that Baader and Hanschke do not commit themselves to a particular concrete domain, but rather view the concrete domain $\mathcal{D}$ as a parameter to their logic, which they call $\mathcal{ALC}(\mathcal{D})$. From the concrete domain perspective, $\mathbb{Q}$-$\mathcal{SHIQ}$ can be viewed as being equipped with one particular concrete domain, whose domain are the rationals and which is equipped with binary predicates $<, \leq, =, \neq, \geq, >$ and with (infinitely many) unary predicates P_q, where $q \in \mathbb{Q}$ and $P \in \{<, \leq, =, \neq, \geq, >\}$.

The paths U_1 and U_2 that may appear inside $\mathbb{Q}$-$\mathcal{SHIQ}$'s binary concrete domain constructors $\exists U_1, U_2.P$ and $\forall U_1, U_2.P$ are of a rather special form: either (i) U_1 and U_2 are feature paths or (ii) U_1 has the form Rg_1 and U_2 has the form g_2. Let us illustrate the expressive power of these two variants of the same constructors: using Variant (i), we can, e.g., describe people whose mother's spouse earns more than their father (we use parentheses for better readability):

$$\exists(\mathsf{mother\ spouse\ wage}), (\mathsf{father\ wage}).>$$

The example illustrates the main advantage of Variant (i): we can talk about *sequences* of features. This variant of $\mathbb{Q}$-$\mathcal{SHIQ}$'s binary concrete domain constructors are precisely the concrete domain constructors offered by $\mathcal{ALC}(\mathcal{D})$ and the temporal DL $\mathcal{TDL}$ mentioned in the introduction. The main disadvantage of Variant (i) is that, inside paths, we may only use abstract features but no roles from $\mathsf{N_{rR}}$. For example, if we want to describe people having an older neighbor by the concept

$$\exists(\mathsf{neighbor}, \mathsf{age}), (\mathsf{age}).>,$$

then "neighbor" should clearly be from $\mathsf{N_{rR}}$ rather than from $\mathsf{N_{aF}}$, since otherwise we would enforce that the described persons have at most a single neighbor. Therefore, we need Variant (ii) of the binary concrete domain constructors to define this concept. Note that Variant (ii) is neither provided by $\mathcal{ALC}(\mathcal{D})$ nor by $\mathcal{TDL}$, but rather is a restricted version of the concrete domain constructors defined in [8].

It is not hard to see that we could also have admitted variants $\exists g_1, Rg_2.P$ and $\forall g_1, Rg_2.P$ of the binary concrete constructors since this variant is just syntactic sugar: $\exists g_1, Rg_2.P$ is equivalent to $\exists Rg_2, g_1.\widetilde{P}$ and $\forall g_1, Rg_2.P$ is equivalent to $\forall Rg_2, g_1.\widetilde{P}$, where $\widetilde{P}$ denotes the inverse of the predicate P—for example, "$\widetilde{<}$" is "$>$" and "$\widetilde{=}$" is "$=$". Obviously, the most general approach would be to allow arbitrary paths inside the binary concrete domain constructors.[1] The resulting logic, however, cannot easily be handled by the ExpTime decision procedure for concept satisfiability presented in the remainder of this paper.

Only simple roles are allowed in Variant (ii) of the binary concrete domain constructors. Similarly, roles used inside number restrictions are also required to be simple. As proved in [13], the latter restriction is crucial since admitting non-simple roles inside number restrictions yields undecidable reasoning problems. Non-simple roles inside the binary concrete domain constructors cannot be handled by the ExpTime decision procedure presented in this paper. However, it is as of now unknown whether admitting them yields undecidability of reasoning.

As a last comment concerning the binary concrete domain constructors, note that there exist existential and universal versions of the binary concrete domain constructors but only an existential version of the unary concrete domain constructor. It is not hard to see that we could also have admitted a universal version since $\forall g.P_q$ (with the obvious semantics) is clearly equivalent to $\forall g, g.\neq \sqcup \exists g.P_q$, where $\forall g, g.\neq$ simply expresses that there exists no successor for the concrete feature g. Similarly, the universal version of Variant (i) of the binary concrete domain constructors can be expressed in terms of the existential version of Variant (i) of this constructor. This does, however, not hold for Variant (ii) of the binary concrete domain constructors since it accepts non-functional roles as arguments. For this reason, we have chosen to include universal versions of both Variant (i) and (ii) for uniformity.

It may look strange at first sight that $\mathbb{Q}$-$\mathcal{SHIQ}$ provides for both abstract features and number restrictions since, as is well-known, number restrictions, transitive roles, and role hierarchies can be used to enforce that a role R_f from $\mathsf{N_{rR}}$ is interpreted functionally: just use the concept $\forall R.(\leqslant 1\ R_f\ \top)$, where $R \in \mathsf{N_{tR}}$, and employ the role hierarchy to ensure that $S \sqsubseteq^* R$ for every "relevant" role S (i.e. for the roles occurring in the concept and role hierarchy whose satisfiability is to be decided). The reason for this redundancy is that number restrictions are, in principle, a strictly more general means of expressivity than abstract features, but having abstract features explicitly available allows for a straightforward definition of Variant (i) of the concrete domain constructors.

[1] Admitting arbitrary paths inside the *unary* concrete domain constructor is not an issue since the concept $\exists R_1 \cdots R_k g.P_q$ (with the obvious semantics) can be written as $\exists R_1. \cdots .\exists R_k.\exists g.P_q$.

3 Preliminaries

Decidability and the ExpTime upper complexity bound for $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability is established by devising an automata-based decision procedure. The general idea behind this procedure is to define, for a given concept C and role hierarchy $\mathcal{R}$, a looping tree-automaton $\mathcal{A}_{C,\mathcal{R}}$ that accepts exactly the so-called Hintikka-trees for C and $\mathcal{R}$. These Hintikka-trees are abstractions of models of C and $\mathcal{R}$, i.e., C and $\mathcal{R}$ have a model if and only if C and $\mathcal{R}$ have a Hintikka-tree. The obvious advantage of Hintikka-trees over models is that they are trees and thus amenable to tree automata techniques. Once the automaton $\mathcal{A}_{C,\mathcal{R}}$ is defined, it remains to apply the standard emptiness test for tree automata: clearly, the language accepted by the constructed automaton is empty iff C is satisfiable w.r.t. $\mathcal{R}$.

In this section, we introduce the basic notions underlying the decision procedure sketched above. We start with developing a useful normal form (called *path normal form*) for $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts, and then introduce looping tree-automata. Finally, we define constraint graphs, which will play an important role in representing the "numerical part" of $\mathbb{Q}$-$\mathcal{SHIQ}$-interpretations in Hintikka-trees.

3.1 Normal Forms

We start with formulating a property of role hierarchies that we will generally assume to be satisfied in what follows:

> A role hierarchy $\mathcal{R}$ is called *admissible* iff all $f \in \mathsf{N_{aF}}$ are simple w.r.t. $\mathcal{R}$.

Demanding admissibility of role hierarchies is closely related to requiring roles R that appear inside number restrictions $(\leqslant n\ R\ C)$ and $(\geqslant n\ R\ C)$ to be simple: since abstract features are interpreted in functional relations, they are "inherently number restricted", i.e., for each $f \in \mathsf{N_{aF}}$, $(\leqslant 1\ f\ \top)$ is satisfied by every domain element in every interpretation. However, it seems that, in contrast to admitting arbitrary roles inside number restrictions, dropping admissibility of role hierarchies does not seem to lead to undecidability of reasoning. Indeed, we claim that the decision procedure presented in this paper can, in principle, be extended to also deal with non-admissible role hierarchies. We nevertheless restrict ourselves to admissible role hierarchies since (i) this eliminates several case distinctions in the proofs, and (ii) we agree with Horrocks and Sattler [11] who argue that non-simple features are rather unnatural: if $f \in \mathsf{N_{aF}}$ is non-simple, then there exists a role $R \in \mathsf{N_R}$ such that $\mathsf{Trans}(R)$ and $R \sqsubseteq^* f$. Hence, R is both functional and transitive which produces strange effects: for any interpretation $\mathcal{I}$, $R^\mathcal{I}$ may not contain any acyclic paths of length greater 1. Hence, the concept $\exists R.\exists R.\top$ is satisfiable only in models that contain either (i) a domain element a which is its own R-successor or (ii) two domain elements a and b, where b is R-successor of a and of itself (the same holds for the concept $\exists R.\exists R.\exists R.\top$). To avoid such effects, which do not seem to promote writing understandable knowledge bases, we generally require role hierarchies to be admissible.

Let us now turn our attention towards the path normal form for $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts, which was first described in [17] in the context of the Description Logic $\mathcal{TDL}$.

Definition 3. A $\mathbb{Q}$-$\mathcal{SHIQ}$-concept C is in *negation normal form* (NNF) if negation occurs only in front of concept names. Moreover, C is in *path normal form* (PNF) iff it is in NNF and, for all subconcepts $\exists U_1, U_2.P$ and $\forall U_1, U_2.P$ of C, we have either

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

It is not hard to see (c.f. [14]) that every $\mathbb{Q}$-$\mathcal{SHIQ}$-concept can be converted into an equivalent one in NNF. In what follows, we use $\sim C$ to denote the result of converting $\neg C$ to NNF. The following lemma shows that we can even assume $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts to be in PNF.

Lemma 4. *Satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts can be polynomially reduced to satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts in PNF.*

Proof We first define an auxiliary mapping and then use this mapping to translate $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts into equivalent ones in PNF. Let C be a $\mathbb{Q}$-$\mathcal{SHIQ}$-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 concrete paths u in C to concepts as follows:

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

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

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

$$\begin{array}{l} \forall f_1^{(1)}.\cdots\forall f_{k_1}^{(1)}.\forall g_1, g_1.\neq \\ \sqcup\, \forall f_1^{(2)}.\cdots\forall f_{k_2}^{(2)}.\forall g_2, g_2.\neq\ \sqcup\ \exists u_1, u_2.P \end{array}$$

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

Now let C be a $\mathbb{Q}$-$\mathcal{SHIQ}$-concept. Using the rewriting rules from [14], we can convert C into an equivalent concept C' in NNF. It is then easy to check that C' is satisfiable iff $\rho(C')$ is satisfiable. Moreover, $\rho(C')$ is clearly in PNF and the translation can be done in polynomial time. ❑

Intuitively, Lemma 4 states that Variant (i) of the binary concrete domain constructors discussed in the previous section can be reduced to the forms $\exists f g_1, g_2.P$ 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. Let us remark that our algorithm's need for PNF is the reason why we cannot handle arbitrary paths inside the binary concrete domain constructors: it is an interesting exercise to check that the constructor $\forall U_1, U_2.P$ with $U_1 = R_1 \cdots R_n g$ and $U_2 = S_1 \cdots S_m g'.P$ can be reduced to the forms $\exists R g_1, g_2.P$ and $\exists g_1, g_2.P$ if $P \in \{<, \leq, =, \geq, >\}$ but *not* if P is "$\neq$".

3.2 Automata and Constraint Graphs

At the core of the decision procedure to be developed are so-called looping tree-automata, i.e., finite automata on infinite trees for which every run is accepting [23; 21].

Definition 5. Let M be a set and $k \geq 1$. A *k-ary M-tree* is a mapping $T : \{1, \ldots, k\}^* \to M$ that labels each node $\alpha \in \{1, \ldots, k\}^*$ with $T(\alpha) \in M$. Intuitively, the node αi is the i-th child of α. We use ϵ to denote the empty word (corresponding to the root of the tree).

A *looping automaton* $\mathcal{A} = (Q, M, I, \Delta)$ for k-ary M-trees is defined by a finite set Q of *states*, a finite alphabet M, a subset $I \subseteq Q$ of *initial states*, and a *transition relation* $\Delta \subseteq Q \times M \times Q^k$.

A *run* of $\mathcal{A}$ on an M-tree T is a mapping $r : \{1, \ldots, k\}^* \to Q$ with $r(\epsilon) \in I$ and

$$(r(\alpha), T(\alpha), r(\alpha 1), \ldots, r(\alpha k)) \in \Delta$$

for each $\alpha \in \{1, \ldots, k\}^*$. The language $L(\mathcal{A})$ of M-trees *accepted* by $\mathcal{A}$ is

$$L(\mathcal{A}) := \{T \mid \text{there is a run of } \mathcal{A} \text{ on } T\}.$$

Vardi and Wolper [23] show that the emptiness problem for looping automata, i.e., the problem to decide whether $L(\mathcal{A}) = \emptyset$ for a given looping automaton $\mathcal{A}$, is decidable in polynomial time.

We now introduce constraint graphs. As already noted, such graphs will be used to represent the "numerical part" of $\mathbb{Q}$-$\mathcal{SHIQ}$-interpretations in Hintikka-trees.

Definition 6. A *constraint graph* is a directed graph $G = (V, E, \tau)$, where V is a countable set of *nodes*,

$$E \subseteq V \times V \times \{<, \leq, =, \neq, \geq, >\}$$

is a set of *labeled edges*, and

$$\tau \subseteq V \times \{P_q \mid P \in \{<, \leq, =, \neq, \geq, >\} \text{ and } q \in \mathbb{Q}\}$$

is a node labeling relation. In what follows, we sometimes write $\tau(v)$ for $\{P_q \mid (v, P_q) \in \tau\}$.

A constraint graph $G = (V, E, \tau)$ is called *satisfiable over S*—where S is a set equipped with a total ordering $<$—iff there exists a total mapping δ from V to S such that

1. $\delta(v)\, P\, q$ for all $P_q \in \tau(v)$ and
2. $\delta(v_1)\, P\, \delta(v_2)$ for all $(v_1, v_2, P) \in E$.

Such a mapping δ is called a *solution* for G.

We will see later that every Hintikka-tree T induces a constraint graph which represents the "numerical part" of the canonical interpretation described by T. As should be intuitively clear, these induced constraint graphs have to be satisfiable in order for Hintikka-trees to be proper abstractions of interpretations. Since, later on, we must define looping automata which accept exactly the Hintikka-trees for a concept C and role hierarchy $\mathcal{R}$, such automata should be able to verify the satisfiability of (induced) constraint graphs. This check is the main problem to be solved when developing an automata-based decision procedure for $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability: the induced constraint graph and its satisfiability are "global" notions while automata work "locally". This problem can be overcome as follows: first, we define Hintikka trees such that their induced constraint graphs have a certain form (we will call such constraint graphs *normal*); second, we formulate an adequate criterion for the satisfiability of normal constraint graphs; and third, we show how this criterion can be verified by "local tests" that can be performed by automata. Let us start with introducing normal constraint graphs and the criterion for their satisfiability, which is called *consistency.*

Definition 7. Let $G = (V, E, \tau)$ be a constraint graph. G is called *normal* if it satisfies the following conditions:

1. $(v_1, v_2, P) \in E$ implies $P \in \{<, =\}$,

2. $(v, P_q) \in \tau$ implies $P \in \{<, =, >\}$,

3. for each rational number q appearing in τ and each node $v \in V$, we have $(v, P_q) \in \tau$ for some $P \in \{<, =, >\}$.

Let $\oplus_n$ denote addition modulo n. A $<$-cycle O in a normal constraint graph G is a a finite non-empty sequence of nodes $v_0, \ldots, v_{k-1} \in V$ such that (i) for all $i < k$, there exists a P such that $(v_i, v_{i \oplus_k 1}, P) \in E$ and (ii) there exists an $i < k$ such that $(v_i, v_{i \oplus_k 1}, <) \in E$.

A normal constraint graph G is *consistent* iff it satisfies the following conditions:

1. G contains no $<$-cycle,

2. for all $v \in V$, there exists a $q \in \mathbb{Q}$ such that qPq' for all $P_{q'} \in \tau(v)$,

3. for all $(v_1, v_2, P) \in V$, there exist $q_1, q_2 \in \mathbb{Q}$ such that
   - $q_1 \, P \, q_2$,
   - $q_1 \, P' \, q$ for all $P'_q \in \tau(v_1)$, and
   - $q_2 \, P' \, q$ for all $P'_q \in \tau(v_2)$.

It may appear that Property 3 of consistency is too weak since it only demands the existence of rationals q_1, q_2 for *each* edge between v_1 and v_2 separately instead of for *all* such edges simultaneously: a normal constraint graph with set of edges $\{(v_1, v_2, <), (v_1, v_2, =)\}$ is clearly unsatisfiable, but does not violate Property 3. This, however, is compensated by Property 1 which is violated in this example.

One can show that consistency is indeed an adequate criterion for the satisfiability of normal constraint graphs.

Theorem 8. *A normal constraint graph G is satisfiable over $\mathbb{Q}$ iff G is consistent.*

It is interesting to note that Theorem 8 also holds if satisfiability over $\mathbb{R}$ is considered instead of satisfiability over $\mathbb{Q}$ (the same proof works). However, as noted in [17], Theorem 8 does *not* hold if satisfiability over non-dense structures such as $\mathbb{N}$ is considered.

Intuitively, every constraint graph $G = (V, E, \tau)$ can be converted into a normal one (called a *normalization* of G) by first specializing the relations in E and τ such that Conditions 1 and 2 of normality are satisfied and then augmenting τ such that Condition 3 holds.

Definition 9 (Normalization). A constraint graph $G = (V, E, \tau)$ is a *normalization* of the constraint graph $G' = (V, E', \tau')$ iff it is normal and the following conditions are satisfied:

1. $(v_1, v_2, P) \in E'$ with $P \in \{<, =\}$ implies $(v_1, v_2, P) \in E$,
2. $(v_1, v_2, >) \in E'$ implies $(v_2, v_1, <) \in E$,
3. $(v_1, v_2, \leq) \in E'$ implies $\{(v_1, v_2, <), (v_1, v_2, =)\} \cap E \neq \emptyset$,
4. $(v_1, v_2, \geq) \in E'$ implies $\{(v_2, v_1, <), (v_1, v_2, =)\} \cap E \neq \emptyset$,
5. $(v_1, v_2, \neq) \in E'$ implies $\{(v_1, v_2, <), (v_2, v_1, <)\} \cap E \neq \emptyset$,
6. if $(v, P_q) \in \tau$, then there exists a $v' \in V$ and a P' such that $(v', P'_q) \in \tau'$,
7. $(v, P_q) \in \tau'$ with $P \in \{<, =, >\}$ implies $(v, P_q) \in \tau$,
8. $(v, \leq_q) \in \tau'$ implies $\{(v, <_q), (v, =_q)\} \cap \tau \neq \emptyset$,
9. $(v, \geq_q) \in \tau'$ implies $\{(v, >_q), (v, =_q)\} \cap \tau \neq \emptyset$, and
10. $(v, \neq_q) \in \tau'$ implies $\{(v, <_q), (v, >_q)\} \cap \tau \neq \emptyset$.

Due to Theorem 8 and the obvious fact that a constraint graph is satisfiable iff one of its normalizations is satisfiable, a constraint graph G is satisfiable iff it has a consistent normalization. This property will play an important role for establishing the correspondence between Hintikka-trees and canonical interpretations.

4 Defining Hintikka-trees

In this section, we define Hintikka-trees, which are, as has already been noted, abstractions of canonical (tree-shaped) interpretations. Let us start with defining, for each concept C (in PNF) and role hierarchy $\mathcal{R}$, the set of concepts $\mathsf{cl}(C, \mathcal{R})$ that are "relevant" for deciding whether a given interpretation is a model of C and $\mathcal{R}$: for a given concept C and role hierarchy $\mathcal{R}$, we use $\mathsf{cl}(C, \mathcal{R})$ to denote the smallest set such that

1. $C \in \mathsf{cl}(C, \mathcal{R})$,
2. $\top \in \mathsf{cl}(C, \mathcal{R})$,
3. if $\forall R.D \in \mathsf{cl}(C, \mathcal{R})$, $\mathsf{Trans}(S)$, and $S \sqsubseteq^* R$, then $\forall S.D \in \mathsf{cl}(C, \mathcal{R})$, and
4. $\mathsf{cl}(C, \mathcal{R})$ is closed under subformulas and $\sim$ (c.f. the remark below Definition 3).

Note that $\sharp\mathsf{cl}(C, \mathcal{R})$ is linear in the length of C and the number of role inclusions in $\mathcal{R}$.

Hintikka-trees are defined in several steps. We start with introducing Hintikka-sets, which form the basis for the definition of so-called Hintikka-labels. As the name indicates, Hintikka-labels are used as node labels in Hintikka-trees. We then define Hintikka-tuples, which are tuples of Hintikka-labels that describe a valid label configuration for a node and its direct successors in a Hintikka-tree (Hintikka-tuples will also be rather convenient for defining looping automata that accept Hintikka-trees). Eventually, we use Hintikka-labels and Hintikka-tuples to define Hintikka-trees.

Intuitively, each node α of a Hintikka-tree T describes a domain element x of the corresponding canonical model $\mathcal{I}$. The node label of α consists of several parts, one of them a Hintikka-set. This Hintikka-set contains all concepts D from $\mathsf{cl}(C, \mathcal{R})$ such that $x \in D^{\mathcal{I}}$.

Definition 10 (Hintikka-set). Let C be a concept in PNF and $\mathcal{R}$ a role hierarchy. A set $\Psi \subseteq \mathsf{cl}(C, \mathcal{R})$ is a *Hintikka-set for C and $\mathcal{R}$* iff it satisfies the following conditions:

(S1) if $C_1 \sqcap C_2 \in \Psi$, then $\{C_1, C_2\} \subseteq \Psi$,

(S2) if $C_1 \sqcup C_2 \in \Psi$, then $\{C_1, C_2\} \cap \Psi \neq \emptyset$,

(S3) $\{A, \neg A\} \not\subseteq \Psi$ for all concept names A,

(S4) if $f \in \mathsf{N_{aF}}$ is used in C or $\mathcal{R}$, then $(\leqslant 1\ f\ \top) \in \Psi$,

(S5) if $(\leqslant n\ R\ D) \in \mathsf{cl}(C, \mathcal{R})$, then $\{D, {\sim}D\} \cap \Psi \neq \emptyset$,

(S6) $\top \in \Psi$

Concepts of the form $\exists R.D$, $(\geqslant n\ R\ D)$, $(\leqslant n\ R\ D)$, and $\exists R g_1, g_2.P$ may appear either *marked* or *unmarked* in Ψ.

The marking of concepts is a technical trick that allows us to deal with the inverse role constructor. Intuitively, edges of a Hintikka-tree T describe role successor relationships of the corresponding canonical model $\mathcal{I}$. If $\exists R.D$ occurs in the Hintikka-set of a node β, then there has to exist a "witness" for this concept: either (i) there exists a successor γ of β such that the edge from β to γ represents an R role relationship and D is in the Hintikka-set of γ, or (ii) β has a predecessor α, the edge from α to β represents an $\mathsf{Inv}(R)$ role relationship, and D occurs in the Hintikka-set of α. The marking of concepts is used for bookkeeping of these two possibilities: if $\exists R.D$ occurs *marked* in the Hintikka-set of β, then its predecessor is a "witness" for $\exists R.D$ and we do not need to enforce the existence of a witness among β's successors. The marking of $(\geqslant n\ R\ D)$, $(\leqslant n\ R\ D)$, and $\exists R g_1, g_2.P$ concepts can be explained similarly. Hintikka-sets are one of the components of Hintikka-labels:

Definition 11 (Hintikka-label). Let C be a concept in PNF and $\mathcal{R}$ a role hierarchy. A *Hintikka-label* $(\Psi, \omega, V, E, \tau)$ *for C and $\mathcal{R}$* consists of

1. a Hintikka-set Ψ for C and $\mathcal{R}$,
2. a set $\omega \subseteq \mathsf{ROL}$ of roles occurring in C or $\mathcal{R}$, and
3. a constraint graph (V, E, τ) where $V \subseteq \mathsf{N_{cF}}$, every $g \in V$ occurs in C, and every q appearing in τ occurs in C.

such that

(N1) if $\exists g_1, g_2.P \in \Psi$, then $g_1, g_2 \in V$ and $(g_1, g_2, P) \in E$,

(N2) if $\forall g_1, g_2.P \in \Psi$ and $g_1, g_2 \in V$, then $(g_1, g_2, P) \in E$,

(N3) if $\exists g.P_q \in \Psi$, then $g \in V$ and $(g, P_q) \in \tau$,

(N4) $R \in \omega$ and $R \sqsubseteq\!\!\!* \, S$ implies $S \in \omega$,

(N5) if $g_1, g_2 \in V$, then $\{(g_1, g_2, <), (g_1, g_2, =), (g_2, g_1, <)\} \cap E \neq \emptyset$,

(N6) if q appears in C, then, for each $g' \in V$, there exists a $P' \in \{<, =, >\}$ such that $(g', P'_q) \in \tau$.

The set of all Hintikka-labels for C and $\mathcal{R}$ is denoted by $\Gamma_{C,\mathcal{R}}$.

Let us explain the intuition behind Hintikka-labels. If α is a node in a Hintikka-tree T, $\mathcal{I}$ the canonical model corresponding to T, and $x \in \Delta_{\mathcal{I}}$ the domain element associated with α, then the Hintikka-label $L = (\Psi, \omega, V, E, \tau)$ of α is a description of x in $\mathcal{I}$. More precisely, (i) the Hintikka-set Ψ is the set of concepts $D \in \mathsf{cl}(C, \mathcal{R})$ such that $x \in D^{\mathcal{I}}$ (ii) ω is the set of roles $R \in \mathsf{ROL}$ such that $(y, x) \in R^{\mathcal{I}}$, where y is the domain element corresponding to the precessor β of α in T; and (iii) the constraint graph (V, E, τ) describes the numerical successors of x and their relationships: if, for some $g \in \mathsf{N_{cF}}$, we have $g \in V$, then $g^{\mathcal{I}}(x)$ is defined. By **(N5)** and **(N6)**, (V, E, τ) fixes the relationship between any two numerical successors of x as well as the relationship between any numerical successor of x and any rational number q appearing in the input concept. By **(N1)**, **(N2)**, and **(N3)**, the relationships stated by E and τ are "consistent" with the Hintikka-set Ψ.

It is rather important that the constraint graph (V, E, τ) fixes the relationship between *any* two nodes of V: as already noted in Section 3, every Hintikka-tree T induces a (normal) constraint graph $G(T)$ that describes the "numerical part" of the canonical interpretation corresponding to T, and should thus be satisfiable. Since $G(T)$ is normal, by Theorem 8 it suffices to demand that $G(T)$ should be consistent. The *complete* determination of the relationships between nodes of the constraint graphs (V, E, τ) in Hintikka-labels will allow us to ensure the consistency of $G(T)$

using a *local* condition which can be verified by looping automata. This condition is part of the definition of Hintikka-tuples, which are introduced next.

Definition 12 (Tuple-graph, Hintikka-tuple). Let C be a concept in PNF and $\mathcal{R}$ a role hierarchy. With $\mathsf{b}_{C,\mathcal{R}}$, we denote

$$\sharp\{D \in \mathsf{cl}(C,\mathcal{R}) \mid D = \exists R.E \text{ or } D = \exists R g_1, g_2.P\} + \sum_{(\geqslant n\ R\ C) \in \mathsf{cl}(C,\mathcal{R})} n.$$

Let $\chi = (L_0, \ldots, L_{\mathsf{b}_{C,\mathcal{R}}})$ be an $\mathsf{b}_{C,\mathcal{R}} + 1$-tuple of Hintikka-labels with $L_i = (\Psi_i, \omega_i, V_i, E_i, \tau_i)$ for $i \le \mathsf{b}_{C,\mathcal{R}}$. A constraint graph $G = (V, E, \tau)$ is a *tuple-graph* for χ if

$$\begin{array}{rcl}
V & = & V_0 \cup \{ig \mid 1 \le i \le \mathsf{b}_{C,\mathcal{R}} \text{ and } g \in V_i\} \\
E & \supseteq & E_0 \cup \{(ig_1, ig_2, P) \mid 1 \le i \le \mathsf{b}_{C,\mathcal{R}} \\
 & & \qquad \text{and } (g_1, g_2, P) \in E_i\} \\
\tau & = & \tau_0 \cup \{(ig, P_q) \mid 1 \le i \le \mathsf{b}_{C,\mathcal{R}} \text{ and } (g, P_q) \in \tau_i\}
\end{array}$$

such that

(G1) if $\exists R g, g'.P$ is unmarked in Ψ_0, then there exists an i with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$ such that $ig, g' \in V$, $R \in \omega_i$, and $(ig, g', P) \in E$,

(G2) if $\exists R g, g'.P$ is marked in Ψ_i with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $g, ig' \in V$, $\mathsf{Inv}(R) \in \omega_i$, and $(g, ig', P) \in E$,

(G3) if $\forall R g, g'.P \in \Psi_0$, $R \in \omega_i$, $g \in V_i$, and $g' \in V_0$ for some i with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $(ig, g', P) \in E$,

(G4) if $\forall R g, g'.P \in \Psi_i$, $\mathsf{Inv}(R) \in \omega_i$, $g \in V_0$, and $g' \in V_i$ for some i with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $(g, ig', P) \in E$.

The tuple χ is a *Hintikka-tuple* iff the following conditions are satisfied:

(M1) if $\exists R.D$ is unmarked in Ψ_0, then there exists an i with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$ such that $R \in \omega_i$ and $D \in \Psi_i$,

(M2) if $(\geqslant n\ R\ D) \in \Psi_0$, then either

- $(\geqslant n\ R\ D)$ is unmarked in Ψ_0 and there exists a set $I \subseteq \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}$ of cardinality n such that, for each $i \in I$, we have $R \in \omega_i$ and $D \in \Psi_i$ or
- $(\geqslant n\ R\ D)$ is marked in Ψ_0 and there exists a set $I \subseteq \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}$ of cardinality $n-1$ such that, for each $i \in I$, we have $R \in \omega_i$ and $D \in \Psi_i$,

(M3) if $\exists R.D$ or $(\geqslant n\ R\ D)$ is marked in Ψ_i with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $\mathsf{Inv}(R) \in \omega_i$ and $D \in \Psi_0$,

(M4) if $\forall R.D \in \Psi_0$ and $R \in \omega_i$ with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $D \in \Psi_i$,

(M5) if $\forall R.D \in \Psi_i$ and $\mathsf{Inv}(R) \in \omega_i$ with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $D \in \Psi_0$,

(M6) if $\forall R.D \in \Psi_0$, $S \in \omega_i$ with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, $\mathsf{Trans}(S)$, and $S \mathrel{\underline{\circledast}} R$, then $\forall S.D \in \Psi_i$,

(M7) if $\forall R.D \in \Psi_i$ and $\mathsf{Inv}(S) \in \omega_i$ with $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, $\mathsf{Trans}(S)$, and $S \mathrel{\underline{\circledast}} R$, then $\forall S.D \in \Psi_0$,

(M8) if $(\leqslant n\ R\ D) \in \Psi_0$, then either

- $(\leqslant n\ R\ D)$ is unmarked in Ψ_0 and the cardinality of the set $\{i \mid 1 \le i \le \mathsf{b}_{C,\mathcal{R}}, R \in \omega_i \text{ and } D \in \Psi_i\}$ is at most n or
- $(\leqslant n\ R\ D)$ is marked in Ψ_0 and the cardinality of the set $\{i \mid 1 \le i \le \mathsf{b}_{C,\mathcal{R}}, R \in \omega_i \text{ and } D \in \Psi_i\}$ is at most $n-1$,

(M9) if $D \in \Psi_0$, $\mathsf{Inv}(R) \in \omega_i$, and $(\leqslant n\ R\ D) \in \Psi_i$ for $1 \le i \le \mathsf{b}_{C,\mathcal{R}}$, then $(\leqslant n\ R\ D)$ is marked in Ψ_i,

(M10) there exists a tuple-graph for χ that has a consistent normalization.

Except for **(M10)**, which refers to tuple-graphs and is the aforementioned local condition enforcing consistency of induced constraint graphs, the properties of Hintikka-tuples should be quite easy to understand. Before we discuss tuple graphs and **(M10)** in more detail, let us introduce Hintikka-trees.

Definition 13 (Hintikka-tree). An $\mathsf{b}_{C,\mathcal{R}}$-ary $\Gamma_{C,\mathcal{R}}$-tree T with $T(\epsilon) = (\Psi_\epsilon, \omega_\epsilon, V_\epsilon, E_\epsilon, \tau_\epsilon)$ is a *Hintikka-tree for C and $\mathcal{R}$* iff it satisfies the following conditions:

(T1) $C \in \Psi_\epsilon$,

(T2) all concepts in Ψ_ϵ are unmarked, and

(T3) for all $\alpha \in \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}^*$, the tuple $(T(\alpha), T(\alpha 1), \ldots, T(\alpha \mathsf{b}_{C,\mathcal{R}}))$ is a Hintikka-tuple.

Let T be a Hintikka-tree, $\alpha \in \{1, \ldots, \mathsf{b}_{C,\mathcal{R}}\}^*$ a node in T, and $T(\alpha) = (\Psi, \omega, V, E, \tau)$. We use $\Psi_T(\alpha)$ to denote Ψ and ω_T to denote ω.

We can now return to the discussion of Property **(M10)**. As is apparent from their definition, tuple-graphs are built by taking the union of all the constraint graphs that appear as a part of the Hintikka-labels in a Hintikka-tuple. The constraint graph $G(T)$ induced by a Hintikka-tree T, in turn, is constructed from tuple-graphs: by **(T3)**, for each node α of T, the tuple

$$\chi_T(\alpha) := (T(\alpha), T(\alpha 1), \ldots, T(\alpha \mathsf{b}_{C,\mathcal{R}}))$$

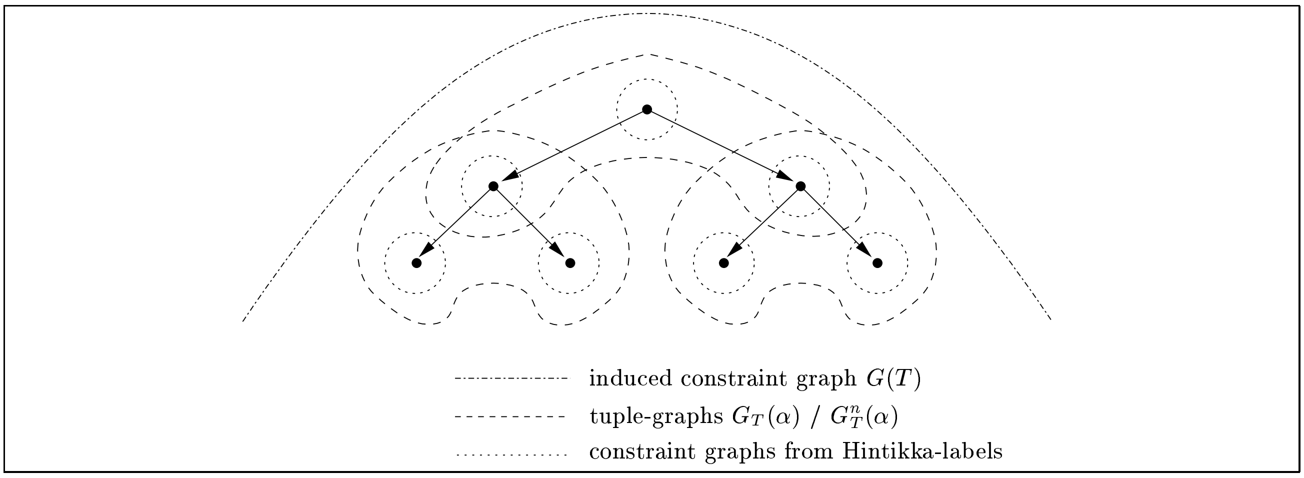
induced constraint graph $G(T)$
tuple-graphs $G_T(\alpha)$ / $G_T^n(\alpha)$
constraint graphs from Hintikka-labels

Figure 2: Hintikka-trees and constraint graphs.

is a Hintikka-tuple. By **(M10)**, there exists a tuple-graph $G_T(\alpha)$ for $\chi_T(\alpha)$ which has a consistent normalization $G_T^n(\alpha)$. Modulo some technical details, the constraint graph $G(T)$ induced by T can be viewed as the union of the constraint graphs $G_T^n(\alpha)$ for all nodes α of T. Figure 2 illustrates the relationship between the various constraint graphs involved. In [14], we prove that the consistency of the normalizations $G_T^n(\alpha)$ enforced by **(M10)** implies consistency of the constraint graph $G(T)$ (which is necessary for T to be a proper abstraction of a $\mathbb{Q}$-$\mathcal{SHIQ}$-interpretation). The hardest part of this proof is to show that $G(T)$ satisfies Property 1 of consistency, i.e., that it contains no $<$-cycle: for this proof, it is crucial that

1. the tuple-graph $G_T(\alpha)$ overlaps with the tuple-graph $G_T(\beta)$ if β is a successor of α in T, and
2. the constraint graphs (V, E, τ), which are part of Hintikka-tuples and thus used to build of tuple-graphs, fix the relationship between *any* two elements of V as discussed above.

Using the fact that the constraint graph induced by Hintikka-trees is consistent, the following, central lemma can be established:

Lemma 14. *A concept C in PNF and a role hierarchy $\mathcal{R}$ have a model iff they have a Hintikka-tree.*

5 Defining Looping Automata

To prove decidability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability, it remains to define a looping automaton $\mathcal{A}_{C,\mathcal{R}}$ for each concept C and role hierarchy $\mathcal{R}$ such that $\mathcal{A}_{C,\mathcal{R}}$ accepts exactly the Hintikka-trees for C and $\mathcal{R}$. Using the notion of Hintikka-tuples from Definition 12, this is rather straightforward.

Definition 15. Let C be a concept in PNF, $\mathcal{R}$ a role hierarchy, and $\mathsf{b}_{C,\mathcal{R}}$ as in Definition 12. The looping automaton $\mathcal{A}_{C,\mathcal{R}} = (Q, M, \Delta, I)$ is defined as follows:

- $Q := M := \Gamma_{C,\mathcal{R}}$
- $I := \{(\Psi, \omega, V, E, \tau) \in Q \mid C \in \Psi$ and all concepts in Ψ are unmarked $\}$.
- $\Delta \subseteq Q^{\mathsf{b}_{C,\mathcal{R}}+2}$ such that $(L, L', L_1, \ldots, L_{\mathsf{b}_{C,\mathcal{R}}}) \in \Delta$ iff
  - $L = L'$ and
  - $(L, L_1, \ldots, L_{\mathsf{b}_{C,\mathcal{R}}})$ is a Hintikka-tuple.

As a consequence of the following lemma and Lemma 14, we can reduce satisfiability of concepts (in PNF) w.r.t. role hierarchies to the emptiness of the language accepted by looping automata.

Lemma 16. *T is a Hintikka-tree for C and $\mathcal{R}$ iff $T \in L(\mathcal{A}_{C,\mathcal{R}})$.*

It is an immediate consequence of Lemmas 4, 14, and 16 and the decidability of the emptiness problem of looping automata [23] that satisfiability of $\mathbb{Q}$-$\mathcal{SHIQ}$-concepts w.r.t. role hierarchies is decidable. However, the presented automata-based algorithm additionally provides us with a tight complexity bound if the numbers inside number restrictions are assumed to be encoded unarily: an ExpTime upper bound is obtained by showing that the size of $\mathcal{A}_{C,\mathcal{R}}$ is at most exponential in the size of C and $\mathcal{R}$ and noting that the emptiness problem for looping automata is in PTime [23]. The ExpTime lower bound is an immediate consequence of the fact that $\mathcal{SHIQ}$-concept satisfiability is already ExpTime-hard [22].

Theorem 17. *If numbers inside number restrictions are encoded unarily, then satisfiability of* $\mathbb{Q}$-$\mathcal{SHIQ}$*-concepts w.r.t. role hierarchies is* ExpTime*-complete.*

Since subsumption can be reduced to (un)satisfiability, $\mathbb{Q}$-$\mathcal{SHIQ}$-concept subsumption w.r.t. role hierarchies is also ExpTime-complete.

6 Future Work

In this paper, we have presented the Description Logic $\mathbb{Q}$-$\mathcal{SHIQ}$, which extends the well-known DL $\mathcal{SHIQ}$ with several concrete domain concept constructors that allow the adequate representation of numerical knowledge. As argued in the introduction, $\mathbb{Q}$-$\mathcal{SHIQ}$ is a contribution to several interesting application areas. However, we regard the work presented in this paper only as a first step. As discussed in [14], there exist many possible future research problems connected with the $\mathbb{Q}$-$\mathcal{SHIQ}$ Description Logic. Let us highlight three of them:

(1) To make $\mathbb{Q}$-$\mathcal{SHIQ}$ available for use in applications, modern DL systems like FaCT and RACER, which are implementations of the $\mathcal{SHIQ}$ Description Logic, need to be extended to $\mathbb{Q}$-$\mathcal{SHIQ}$. Unfortunately, the results presented in this paper cannot immediately be used for this task: the aforementioned DL systems are based on tableau-style algorithms while the decision procedure described in this paper is automata-based. Hence, it would be interesting to devise a tableau-based algorithm for $\mathbb{Q}$-$\mathcal{SHIQ}$-concept satisfiability. As discussed in [15] in the context of $\mathcal{TDL}$, the automata-based algorithm presented in this paper can provide important information (i.e., a "regular model property") for this task.

(2) If $\mathbb{Q}$-$\mathcal{SHIQ}$ is to be used for reasoning about ER diagrams as sketched in the introduction, one is usually not interested in the satisfiability of concepts in arbitrary models, but rather in the satisfiability in finite models [3]. These two problems do not coincide since $\mathcal{SHIQ}$, and hence also $\mathbb{Q}$-$\mathcal{SHIQ}$, lacks the finite model property [11]. Thus, it is worthwhile to investigate the decidability and complexity of finite model reasoning with $\mathbb{Q}$-$\mathcal{SHIQ}$.

(3) For some applications, it is desirable to refer to natural numbers instead of rational numbers. As a simple example, consider the concept

$$\begin{aligned}
&\exists(\mathsf{left\text{-}neighbor\ numchild}).=_2\\
&\sqcap\ \exists(\mathsf{left\text{-}neighbor\ numchild}),(\mathsf{numchild}).<\\
&\sqcap\ \exists(\mathsf{right\text{-}neighbor\ numchild}).=_3\\
&\sqcap\ \exists(\mathsf{numchild}),(\mathsf{right\text{-}neighbor\ numchild}).<,
\end{aligned}$$

where numchild is a concrete feature representing the number of children. Clearly, the above concept should be unsatisfiable. In $\mathbb{Q}$-$\mathcal{SHIQ}$, however, this concept is satisfiable since, in a model, the number of children of the described person may e.g. be 2.5. It would thus be interesting to add a concept constructor $\exists g.\mathsf{nat}$ to $\mathbb{Q}$-$\mathcal{SHIQ}$ expressing that the filler of the concrete feature g is a natural number. If the extended logic should be decidable at all, then at least it seems to require some serious modifications of the presented decision procedure: as noted in [17] in the context of $\mathcal{TDL}$, Theorem 8 does *not* hold if the satisfiability of constraint graphs over non-dense structures such as $\mathbb{N}$ is considered. However, if $\mathbb{Q}$-$\mathcal{SHIQ}$ is extended with an $\exists g.\mathsf{nat}$ constructor, then concepts of the resulting logic can clearly be used to describe constraint graphs all of whose nodes are labeled with the nat predicates. This means that, effectively, we have to decide satisfiability of these constraint graphs over $\mathbb{N}$.

We should like to note that, in many aspects, $\mathbb{Q}$-$\mathcal{SHIQ}$ is already on the border to undecidability: for example, it seems rather unlikely that any kind of arithmetics can be added to $\mathbb{Q}$-$\mathcal{SHIQ}$ without losing decidability. More precisely, it follows from results in [18; 16] that the addition of a concept constructor expressing the addition of two numbers already yields undecidability of reasoning.

Acknowledgements

I would like to thank Ulrike Sattler for helpful comments.

References

[1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, Australia, 1991.

[2] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proceedings of the 16th German AI-Conference (GWAI-92)*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143. Springer-Verlag, 1992.

[3] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* Dottorato di ricerca in informatica, Università degli Studi di Roma "La Sapienza", Italia, 1996.

[4] 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.

[5] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal for Intelligent and Cooperative Information Systems*, 2(4):375–399, 1993.

[6] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proceedings of the European Knowledge Acquisition Conference (EKAW 2000)*, volume 1937 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, 2000.

[7] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 161–166. Morgan-Kaufmann, 2001.

[8] P. Hanschke. Specifying role interaction in concept languages. In W. Nebel, Bernhard; Rich, Charles; Swartout, editor, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 318–329. Morgan Kaufmann, 1992.

[9] I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In C. Goble, D. L. McGuinness, R. Möller, and P. F. Patel-Schneider, editors, *Proceedings of the International Workshop in Description Logics 2001 (DL2001)*, number 49 in CEUR-WS (http://ceur-ws.org/), pages 30–35, 2001.

[10] I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[11] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3), 1999.

[12] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(D)$ description logic. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 199–204. Morgan-Kaufmann, 2001.

[13] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[14] C. Lutz. Adding numbers to the $\mathcal{SHIQ}$ description logic—first results. LTCS-Report 01-07, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[15] C. Lutz. Interval-based temporal reasoning with general TBoxes. LTCS-Report 00-06, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[16] C. Lutz. *The Complexity of Reasoning with Concrete Domains.* PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

[17] C. Lutz. Interval-based temporal reasoning with general TBoxes. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 89–94. Morgan-Kaufmann, 2001.

[18] C. Lutz. NExpTime-complete description logics with concrete domains. 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 45–60. Springer-Verlag, 2001.

[19] C. Lutz. Reasoning about entity relationship diagrams with constraints on attributes. Submitted to the 2002 Workshop on Description Logics, 2002.

[20] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[21] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.

[22] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, RWTH Aachen, 2001.

[23] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.