Reasoning in the $\mathcal{SHOQ(D_n)}$ Description Logic

Jeff Z. Pan and Ian Horrocks

Information Management Group
Department of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL, UK
{pan, horrocks}@cs.man.ac.uk

Abstract

Description Logics (DLs) are of crucial importance to the development of the Semantic Web, where their role is to provide formal underpinnings and automated reasoning services for Semantic Web ontology languages such as DAML+OIL. In this paper\(^1\) we show how the description logic $\mathcal{SHOQ(D)}$, which has been designed to provide such services, can be extended with n-ary datatype predicates as well as datatype number restrictions, to give $\mathcal{SHOQ(D_n)}$, and we present an algorithm for deciding the satisfiability of $\mathcal{SHOQ(D_n)}$ concepts, along with a proof of its soundness and completeness. The work is motivated by the requirement for n-ary datatype predicates in relation to "real world" properties such as size, weight and duration, in the Semantic Web applications.

1 Introduction

Description Logics (DLs) are of crucial importance to the development of the so-called "Semantic Web" [2], where their role is to provide formal underpinnings and automated reasoning services for Semantic Web ontology languages \([3, 10]\), such as DAML+OIL \([6]\). Significant effort has already been devoted to the investigation of suitable DLs—in particular, Horrocks and Sattler \([7]\) have presented the $\mathcal{SHOQ(D)}$ DL, along with a sound and complete algorithm for deciding concept satisfiability, a basic reasoning service for DLs and ontologies. A key feature of $\mathcal{SHOQ(D)}$ is that, like DAML+OIL, it supports datatypes \([1]\) (e.g., string, integer) as well as the usual abstract concepts (e.g., animal, plant).

$\mathcal{SHOQ(D)}$, however, supports a very restricted form of datatypes, i.e., it can only deal with unary datatype predicates. While this is enough for the current version of the DAML+OIL language, it may not be adequate for some Semantic Web applications and for possible future extensions of DAML+OIL. E.g., ontologies used in e-commerce may want to classify different items according to their sizes, and to reason that an item which has height less than 5cm and the sum of its length and width less than 10cm is a kind of item for which no shipping costs are charged. Here "less than 5cm(height)" is a unary datatype predicate, and "the sum less than 10cm(length,width)" is a binary predicate (see also the according $\mathcal{SHOQ(D_n)}$-concept in Section 2). As shown above, unary predicates are not enough in the example, while n-ary datatype predicates are often necessary for "real world" properties, such as size, weight, duration etc., in the ontology applications.

\(^1\)Also available at \texttt{http://www.cs.man.ac.uk/~pan/zhilin/download/Paper/Pan-Horrocks-shoqdn-2002.pdf}
An approach of extending DL with datatypes was first introduced by Baader and Hanschke [1], who described a datatype (D) extension of the well known ACC DL. Baader and Hanschke [1] have shown that although the satisfiability of ACC(D) is decidable, if ACC(D) is extended with transitive closure of features, the satisfiability problem is undecidable. Lutz [8] proved that reasoning with ACC(D) and general TBoxes is undecidable. In order to extend expressive DLs with concrete domains, Horrocks and Sattler [7] proposed a simplified approach on concrete domain and applied this approach on the SHOQ(D) DL. Pan [9] investigated the simplifying constraints of SHOQ(D) w.r.t. datatypes, and showed how these could be relaxed in order to extend SHOQ(D) with n-ary datatype predicates. We should mention that, similar to Baader and Hanschke [1]'s approach, Haarslev et al. [4] extended the SH(N) DL with restricted concert domain (D) and gave the SH(N(D) DL, which supports n-ary datatype.

In this paper, we extend our work in [9] and add datatype number restrictions to give the SHOQ(D_n) DL, and present a sound and complete decision procedure for concept satisfiability and subsumption. The rest of the paper is organized as follows. In Section 2, we give the definition of the SHOQ(D_n) DL. In Section 3, we define a tableau for SHOQ(D_n). In Section 4, we give an algorithm and its decidability proof. Section 5 is a brief discussion on future works of SHOQ(D_n).

2 SHOQ(D_n)

Definition 1 \( \Delta_D \) is the datatype domain covering all concrete datatypes.

Definition 2 Let \( C, R = R_A \sqcup R_D, I \) be disjoint sets of concept, abstract and concrete role and individual names. For \( R \) and \( S \) roles, a role axiom is either a role inclusion, which is of the form \( R \sqsubseteq S \) for \( R, S \in R_A \) or \( R, S \in R_D \), or a transitivity axiom, which is of the form \( \text{Trans}(R) \) for \( R \in R_A \). A role box \( R \) is a finite set of role axioms. A role \( R \) is called simple if, for \( \sqsubseteq \) the transitive reflexive closure of \( \sqsubseteq \) on \( R \) and for each role \( S, S \in R \) implies \( \text{Trans}(S) \notin R \).

The set of concept terms of \( \text{SHOQ}(D_n) \) is inductively defined. As a starting point of the induction, any element of \( C \) is a concept term (atomic terms). Now let \( C \) and \( D \) be concept terms, \( o \) be an individual, \( R \) be a abstract role name, \( T_1, \ldots, T_n \) be concrete role names, \( P \) be an n-ary predicate name. Then the following expressions are also concept terms:

1. \( \top \) (universal concept) and \( \top_D \) (universal datatype),
2. \( C \sqcap D \) (disjunction), \( C \sqcap D \) (conjunction), \( \neg C \) (negation), and \{o\} (nominals),
3. \( \exists R.C \) (exists-in restriction) and \( \forall R.C \) (value restriction),
4. \( \geq m R.C \) (atleast restriction) and \( \leq m R.C \) (atmost restriction),
5. \( \exists T_1, \ldots, T_n.P_n \) (datatype exists) and \( \forall T_1, \ldots, T_n.P_n \) (datatype value),
6. \( \geq m T_1, \ldots, T_n.P_n \) (datatype atleast) and \( \leq m T_1, \ldots, T_n.P_n \) (datatype atmost),
7. \( \geq m T, \leq m T \) (number restrictions on concrete roles).

\( \text{SHOQ}(D_n) \) extends \( \text{SHOQ}(D) \) by supporting n-ary datatype predicates \( P_n \), the interpretation \( P_n^D \) of which is

\[
P_n^D \subseteq d_1^D \times \cdots \times d_n^D \subseteq \Delta_D
\]

where \( d_i^D \in \Delta_D \) are datatypes. The interpretation of the projection of \( P_n \) is defined as

\[
P_n(i)^D \subseteq d_i^D \subseteq \Delta_D
\]
and note that when we say

\[ \langle y_1, \ldots, y_n \rangle \in P_n^D \]

we mean: (i) \( y_i \in P_n(i)^D \) for \( 1 \leq i \leq n \), and (ii) \( y_1, \ldots, y_n \) satisfy datatype predicate \( P_n \). The interpretations of other datatype constructs are listed in Figure 1. Note that concrete role at least (atmost) is only a special form of datatype at least (atmost, respectively) where \( n = 1 \) and \( P_n = \top_0 \).

To illustrate the use of \( SHO\Omega(D_n) \)-concept, let’s go back to the example we used in Section 1. Items with height less than \( 5 \text{cm} \), and the sum of their length and width less that \( 10 \text{cm} \) can be defined as a \( SHO\Omega(D_n) \)-concept

\[ \text{item} \cap 1\text{height} < 5\text{cm} \cap 1\text{length} \cap 1\text{width} \cap \forall \text{length, width. sum} < 10\text{cm} \]

where “\( =1 \)” is a shortcut for “\( =1 \cap \geq 1 \)”, and \( \text{height, length and width} \) are concrete roles, \( < 5\text{cm} \) is a unary datatype predicate and \( \text{sum} < 10\text{cm} \) is a binary predicate. Note that \( < 5\text{cm} \) and \( \text{sum} < 10\text{cm} \) are datatype predicates, rather than datatype number restrictions.

Datatypes and predicates are considered to be already sufficiently structured by the type system, which may includes a derivation mechanism and built-in ordering relations, so that it can be used to define datatypes \( d \) and predicates \( P_n \), as well as negation of predicates \( \neg P_n \), it can be used to check if a tuple of values \( t_1, \ldots, t_n \) satisfy a predicate \( P_n \), if a set of tuples of values satisfy a set of predicates simultaneously etc. With the type system, we can deal with an arbitrary conforming set of datatypes and predicates without compromising the compactness of the concept language or the soundness and completeness of our decision procedure [7].

3 A Tableau for \( SHO\Omega(D_n) \)

In this section, we define a tableau for \( SHO\Omega(D_n) \). For ease of presentation, we assume all concepts to be in negation normal form (NNF). We use \( C \) to denote the NNF of \( \neg C \). Moreover, for a concept \( D \), we use \( cl(D) \) to denote the set of all sub-concepts of \( D \), the NNF of these sub-concepts, and the (possibly negated) datatypes occurring in these (NNF of) sub-concepts.

**Definition 3** If \( D \) is a \( SHO\Omega(D_n) \)-concept in NNF, \( R \) a role box, and \( R_n^D, R_n^D \) are the sets of abstract and concrete roles occurring in \( D \) or \( R \), a tableau \( T \) for \( D \) w.r.t. \( R \) is defined as a quadruple \( (S, L, E_A, E_D) \) such that: \( S \) is a set of individuals, \( L : S \rightarrow 2^{(D)} \) maps each individual to a set of concepts which is a subset of \( cl(D) \), \( E_A : R_n^D \rightarrow 2^{S \times S} \) maps each abstract
role in $\mathbf{R}_D^0$ to a set of pairs of individuals, $\mathcal{E}_D : \mathbf{R}_D^0 \to \mathbf{2}^{S \times \Delta_D}$ maps each concrete role in $\mathbf{R}_D^0$ to a set of pairs of individuals and concrete values, and there is some individual $s \in S$ such that $D \in \mathcal{L}(s)$. For all $s, t \in S$, $C, C_1, C_2 \in \mathcal{C}(D)$, $R, S \in \mathbf{R}_A^0, T, T', T_1, \ldots, T_n \in \mathbf{R}_D^0, n$-ary predicate $P_n$ and

$$
S^T(s, C) := \{t \in S \mid \langle s, t \rangle \in \mathcal{E}_A(S) \text{ and } C \in \mathcal{L}(t)\},
$$

$$
T_1 T_2 \ldots T_n(s, P_n) := \{(y_1, \ldots, y_n) \in P_n \mid \langle s, y_i \rangle \in \mathcal{E}_D(T_i) \text{ for } 1 \leq i \leq n\},
$$

$$
DC^T(s, T_1, \ldots, T_n, y_1, \ldots, y_n, P_n) :=
\begin{cases}
  \text{true} & \text{if } \langle s, y_i \rangle \in \mathcal{E}_D(T_i) \text{ for } 1 \leq i \leq n \text{ and } \langle s, y_i \rangle \in \mathcal{E}_D(T_i) \\
  \text{false} & \text{otherwise}
\end{cases}
$$

it holds that:

(P1) if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,

(P2) if $C_1 \cap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,

(P3) if $C_1 \cup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,

(P4) if $\langle s, t \rangle \in \mathcal{E}_A(R)$ and $R \subseteq S$, then $\langle s, t \rangle \in \mathcal{E}_A(S)$,

(P5) if $\forall R.C \in \mathcal{L}(s)$ and $(s, t) \in \mathcal{E}_A(R)$, then $C \in \mathcal{L}(t)$,

(P6) if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in S$ such that $\langle s, t \rangle \in \mathcal{E}_A(R)$ and $C \in \mathcal{L}(t)$,

(P7) if $\forall S.C \in \mathcal{L}(s)$ and $(s, t) \in \mathcal{E}_A(R)$ for some $R \subseteq S$ with $\text{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,

(P8) if $\exists n S.C \in \mathcal{L}(s)$, then $\exists S^T(s, C) \leq n$,

(P9) if $\forall n S.C \in \mathcal{L}(s)$, then $\forall S^T(s, C) \geq n$,

(P10) if $\{s, t \in \mathcal{E}_A(S) \mid \exists n S.C \in \mathcal{L}(s) \cap \mathcal{L}(t) \neq \emptyset \}$ and $\langle s, t \rangle \in \mathcal{E}_A(S)$, then $\{C, \neg C \} \cap \mathcal{L}(t) = \emptyset$,

(P11) if $\{s \in \mathcal{L}(s) \cap \mathcal{L}(t) \mid \neg s = t\}$, then $s = t$,

(P12) if $\forall T_1, \ldots, T_n \in \mathcal{L}(s)$ and $(s, t_1) \in \mathcal{E}_D(T_1), \ldots, (s, t_n) \in \mathcal{E}_D(T_n)$, then $\exists T_1 T_2 \ldots T_n(s, P_n) = \text{true}$,

(P13) if $\exists T_1, \ldots, T_n \in \mathcal{L}(s)$, then there is some $t_1, \ldots, t_n \in \Delta_D$ such that $\langle s, t_1 \rangle \in \mathcal{E}_D(T_1), \ldots, \langle s, t_n \rangle \in \mathcal{E}_D(T_n)$, $\exists T_1 T_2 \ldots T_n(s, P_n) = \text{true}$,

(P14) if $\exists m T_1, \ldots, T_n \in \mathcal{L}(s)$, then $\exists T_1 T_2 \ldots T_n(s, P_n) \geq m$,

(P15) if $\exists m T_1, \ldots, T_n \in \mathcal{L}(s)$, then $\exists T_1 T_2 \ldots T_n(s, P_n) \leq m$,

(P16) if $\{x \in \mathcal{E}_A(R) \mid \exists n T_1, \ldots, T_n \in \mathcal{L}(s) \cap \mathcal{L}(t) \neq \emptyset \}$ and $\langle s, x \rangle \in \mathcal{E}_A(S)$, then for $1 \leq i \leq n$, we have either $\exists T_1 T_2 \ldots T_n(s, P_n) = \text{true}$, or $\exists T_1 T_2 \ldots T_n(s, P_n) = \text{true}$.

Lemma 4 A $S^H\OQ(D_n)$-conceput $D$ in NNF is satisfiable w.r.t. a role box $\mathcal{R}$ iff $D$ has a tableau w.r.t. $\mathcal{R}$.

Proof: For the if direction, if $\mathcal{T} = (\mathbf{S}, \mathcal{L}, \mathcal{E}_A, \mathcal{E}_D)$ is a tableau for $D$, a model $\mathcal{T} = (\Delta_T, \mathcal{T})$ of $D$ can be defined as: $\Delta_T = S, \mathcal{CN}_T = \{s \mid s \in \mathcal{L}(s)\}$ for all concept names $CN$ in $\mathcal{C}(D)$, if $R \in \mathbf{R}_A, \mathbf{R}_A^+ = \mathcal{E}_A(R)^+$, otherwise $\mathbf{R}_A^+ = \mathcal{E}_A(R) \cup \bigcup_{F \in \mathcal{R}_F} \mathbf{R}_F^+$. $\mathbf{R}_D^+ = \mathcal{E}_D(R)$, where $\mathcal{E}_A(R)^+$ denotes the transitive closure of $\mathcal{E}_A(R)$. $D^+ \neq \emptyset$ because $s_0 \in D^+$. Here we only concentrate on (P14) to (P15); the remainder is similar to the proofs found in [9].

Note that in this paper, we mainly focus on the proof of the number restriction on concrete roles, the remainder is similar to the proofs found in [9].
1. $E = \geq m T_1, \ldots, T_n, P_n$. According to (P14), $E \in L(s)$ implies that $\exists T_1 T_2 \ldots T_n P_n \geq m$. By the definition of $T_1 T_2 \ldots T_n P_n$, we have $s \in \{ t \in \Delta^2 | \exists \{ t_1, \ldots, t_n \} \in E_D(T_i) \wedge \{ t_1, \ldots, t_n \} \in P_n \} \geq m$. Since $E_D(T_i) = T_i^2$, we have $s \in \{ \geq m t_1, \ldots, T_n, P_n \}^2$. Similarly, if $E = \leq m T_1, \ldots, T_n, P_n$, we have $s \in \{ \leq m t_1, \ldots, T_n, P_n \}^2$.

For the converse, if $I = (\Delta^2, \cdot)$ is a model of $D$, then a tableau $T = (S, L, E_A, E_D)$ for $D$ can be defined as: $S = \Delta^2$, $E_A(R) = R_A^2$, $E_D(R) = R_D^2$, $L(s) = \{ C \in c_1(D) | s \in C^2 \}$. It only remains to demonstrate that $T$ is a tableau for $D$: $T$ satisfies (P14) to (P16) as a direct consequence of the semantics of datatype constructs.

4 A Tableau Algorithm for $SHOQ(D_n)$

Form Lemma 4, an algorithm which constructs a tableau for a $SHOQ(D_n)$-concept $D$ can be used as a decision procedure for the satisfiability of $D$ with respect to a role box $R$.

Definition 5 Let $R$ be a role box, $D$ a $SHOQ(D_n)$-concept in NNF, $R_A^D$ the set of abstract roles occurring in $D$ or $R$, and $R_D$ the set of nominal occurring in $D$. A tableau algorithm works on a completion forest for $D$ w.r.t. $R$, which is a set of trees $F$. Each node $x$ of the forest is labelled with a set $L(x) \subseteq c_1(D) \cup \{ \uparrow (R, \{ o \}) | R \in R_A^D \text{ and } \{ o \} \in R_D \}$, and each edge $(x, y)$ is labelled with a set of role names $L((x, y))$ containing roles occurring in $c_1(D)$ or $R$. Concrete values are represented by concrete nodes, which are always leaves of $F$. Additionally, we keep track of inequalities between nodes of the tree with a symmetric binary relation $\neq$ between the nodes of $F$. For each $\{ o \} \in R_D$ there is a distinguished node $x_{\{ o \}}$ in $F$ such that $\{ o \} \in L(x)$ for some node $x$ or by adding new leaf nodes.

Given a completion forest, a node $y$ is called an $R$-successor of a node $x$ if, for some $R'$ with $R' \sqsubseteq R$, either $y$ is a successor of $x$ or $R' \in L((x, y))$, or $\uparrow (R', \{ o \}) \in L(x)$ and $y = x_{\{ o \}}$. Ancestors and roots are defined as usual. For an abstract role $S$ and a node $x$ in $F$ we define $S^S(x, C)$ by $S^S(x, C) := \{ y | y \text{ is an } S\text{-successor of } x \text{ and } C \in L(y) \}$.

Given a completion forest, concrete nodes $t_1, \ldots, t_n$ are called $T_1 T_2 \ldots T_n$-successors of a node $x$ if, for some concrete roles $T'_1, \ldots, T'_n$ with $T'_1 \sqsubseteq T_1, \ldots, t_n$ are successors of $x$ and $T'_i \in L((x, t_i)), 1 \leq i \leq n$. For a node $x$, its $T_1 T_2 \ldots T_n$-successors $\langle t_1, \ldots, t_n \rangle$, $n$-ary datatype predicate $P_n$, we define a set $DC^S$ by $DC^S = \{ \langle DC\text{element} > \}$ where $DC\text{element}$ is a set of $DC\text{elements}$, which have the form $\langle DC\text{element} > = \{ x, \langle T_1, \ldots, T_n \rangle, \langle t_1, \ldots, t_n \rangle, P_n \}$ $DC^S$ is initialised as an empty set. $DC^S$ is satisfied iff. there exists value : $N_C \rightarrow \Delta_D$, where $N_C$ is the set of all concrete nodes, s.t. for all $x, \langle T_1, \ldots, T_n \rangle, \langle t_1, \ldots, t_n \rangle, P_n \in DC^S$, $(value(t_1), \ldots, value(t_n)) \in P_n$ are true. In order to retrieve the set of all the $T_1 T_2 \ldots T_n$-successors of $x$, which satisfy a certain predicate $P_n$, we define $DC\text{successor}^S(x, P_n)$ by $DC\text{successor}^S(x, T_1, \ldots, T_n, P_n) := \{ \langle t_1, \ldots, t_n \rangle \in \{ x, \langle T_1, \ldots, T_n \rangle, \langle t_1, \ldots, t_n \rangle, P_n \} \in DC^S \}$

In order to retrieve the set of datatype predicates, which are satisfied by $T_1 T_2 \ldots T_n$-successors $t_1, \ldots, t_n$ of $x$, we define $DC\text{predicates}^S(x, T_1, \ldots, T_n, t_1, \ldots, t_n)$ by $DC\text{predicates}^S(x, T_1, \ldots, T_n, t_1, \ldots, t_n) := \{ P_n \in DC^S \}$
∀P-rule: if 1.∀T₁,⋯,Tₙ, Pᵢ ∈ \( L(x) \), x is not blocked, and  
  2. there are \( T₁T₂ \ldots Tₙ \)-successors \( (t₁, \ldots, tₙ) \) of \( x \),  
  with \( Pᵢ \not\in DCPredicates\( x,T₁,\ldots,Tₙ,t₁,\ldots,tₙ \),  
  then \( DC^ℙ \rightarrow DC^ℙ \cup \{x, (T₁,\ldots,Tₙ),(t₁,\ldots,tₙ), Pᵢ\} \).  

∃P-rule: if 1.∃T₁,⋯,Tₙ, Pᵢ ∈ \( L(x) \), x is not blocked, and  
  2. there are no \( T₁T₂ \ldots Tₙ \)-successors \( (t₁,\ldots,tₙ) \) of \( x \),  
  with \( Pᵢ \in DCPredicates\( x,T₁,\ldots,Tₙ,t₁,\ldots,tₙ \),  
  then 1. create \( T₁T₂ \ldots Tₙ \)-successors \( (t₁,\ldots,tₙ) \) with \( L((x,tᵢ)) = \{Tᵢ\} \)  
  for \( 1 \leq i \leq n \), and  
  2. \( DC^ℙ \rightarrow DC^ℙ \cup \{x, (T₁,\ldots,Tₙ),(t₁,\ldots,tₙ), Pᵢ\}\)  
  and  
  3. set \( (t₁,\ldots,tₙ) \neq (t₁,\ldots,tₙ) \), for all \( 1 \leq i \leq n \), \( 1 \leq j < k \leq m \).  

≤ₚ-rule: if 1. ≤ₚ T₁,⋯,Tₙ, Pᵢ ∈ \( L(x) \), x is not blocked, and  
  2. \( DC\text{Successor}^{\exists}(x,T₁,\ldots,Tₙ,Pᵢ) > m \)  
  and  
  3. there exist \( j \neq k \), s.t. \( (t₁,\ldots,tₙ),(t₁,\ldots,tₙ) \in DC\text{Successor}^{\exists}(x, \)  
  \( T₁,\ldots,Tₙ,Pᵢ) \) but not \( (t₁,\ldots,tₙ),(t₁,\ldots,tₙ) \neq (tₖ,\ldots,tₖ) \), \( 1 \leq j < k \leq m \),  
  then 1. \( L((x,tᵢ)) \rightarrow L((x,tᵢ)) \cup L((x,tᵢ)) \),  
  2. \( DC^ℙ \rightarrow DC^ℙ \cup \{x, (T₁,\ldots,Tₙ),(t₁,\ldots,tₙ), Pᵢ\}\),  
  and  
  3. add \( u \neq (t₁,\ldots,tₙ) \) for each tuple \( u \) with \( u \neq (t₁,\ldots,tₙ) \),  
  and  
  4. remove all \( tᵢ \) where \( tᵢ \) isn’t in any tuples of \( DC\text{Successor}^{\exists}(x, *, *) \)  
  and  
  remove all edges leading to these \( tᵢ \) from \( F \).  

chooseP-rule: if 1. \( \{≤ₚ T₁,\ldots,Tₙ, Pᵢ \} \cup L(x) \neq ∅ \), x is not blocked, and  
  2. \( (t₁,\ldots,tₙ) \) are \( T₁T₂ \ldots Tₙ \)-successors of \( x \), and  
  then either  
  1. \( DC^ℙ \rightarrow DC^ℙ \cup \{x, (T₁,\ldots,Tₙ),(t₁,\ldots,tₙ), Pᵢ\}\),  
  or  
  2. \( DC^ℙ \rightarrow DC^ℙ \cup \{x, (T₁,\ldots,Tₙ),(t₁,\ldots,tₙ), Pᵢ\}\).  

Note that we can use * as parameter in \( DC\text{Successor}^{\exists} \) and \( DC\text{Predicates}^{\exists} \), e.g. \( DC\text{Successor}^{\exists} \)  
or \( ^{\exists}(x, *, *) \) means all the concrete successors of node \( x \).  

A node \( x \) is directly blocked if none of its ancestors are blocked, and it has an ancestor \( x' \)  
that is not distinguished such that \( L(x) \subseteq L(x' \). We call \( x' \) blocks \( x \). A node is blocked if it  
is directly blocks or if its predecessor is blocked.  

If \( \{v₁,\ldots,vₙ\} \) are all individuals occurring in \( D \), the algorithm initialises the completion  
forest \( F \) to contain \( l+1 \) root nodes \( x₀, x_{v₁}, \ldots, x_{vₙ} \) with \( C ∈ L(x₀) \) for each \( 0 ≤ i ≤ n \) and \( yᵢ \neq yⱼ \) for each \( 0 ≤ i < j ≤ n \), or  

3. \( DC^ℙ \) isn’t satisfied;  

4. for some concrete roles \( T₁,\ldots,Tₙ, \) n-ary datatype predicate \( Pᵢ \), \( ≤ₚ T₁,\ldots,Tₙ, Pᵢ ∈ \)  
\( L(x) \), we have \( \#(T₁T₂ \ldots Tₙ(x, Pᵢ)) ≥ m \), or  

\*Figure 2 only lists the rules about datatypes, other rules can be found in [9].
5. for some \( \{ o \} \in \mathcal{L}(x), x \neq x_{\{ o \}} \).

The completion forest is \textit{complete} when, for some node \( x \), \( \mathcal{L}(x) \) contains a clash, or when none of the expansion rules is applicable. If the expansion rules can be applied in such a way that they yield a complete, clash-free completion forest, then the algorithm returns “\( D \) is \textbf{satisfiable} w.r.t. \( \mathcal{R} \)”, and “\( D \) is \textbf{unsatisfiable} w.r.t. \( \mathcal{R} \)” otherwise.

\textbf{Lemma 6 (Termination)} When started with a \( \mathcal{SHCQ}(D_n) \)-concept \( D \) in NNF, the tableau algorithm terminates.

\textbf{Proof:} Let \( d = |c(D)|, k = |R^D_A|, n_{\text{max}} \) the maximal number in at least number restrictions as well as datatype at least, and \( \ell = |T^D| \). Here we mainly concentrate on rules about number restriction on concrete roles. Termination is a consequence of the following properties of the expansion rules:

1. Each rule but the \( \leq \), \( \leq^p \)- or the \( \mathbf{O} \)-rule strictly extends the completion forest, by extending node labels or adding nodes, while removing neither nodes nor elements from node.

2. New nodes are only generated by the \( \exists \), \( \exists^p \), \( \geq \)-rule or the \( \geq_p \)-rule as successors of a node \( x \) for concepts of the form \( \exists R.C, \exists T_1, \ldots, T_n.P_n, \geq m T_1, \ldots, T_n.P_n \in \mathcal{L}(x) \). For a node \( x \), each of these concepts can trigger the generation of successors at most once—even though the node[s] generated was later removed by either the \( \leq \), \( \leq^p \)- or the \( \mathbf{O} \)-rule. For the \( \geq \)-rule: If \( T_1 T_2 \ldots T_n \)-successors \( \{ t_{i_1}, \ldots, t_{i_m} \} \) were generated by an application of the \( \geq_p \)-rule for a concept \( \geq m T_1, \ldots, T_n.P_n \), then \( \{ t_{j_1}, \ldots, t_{j_n} \} \neq \{ t_{i_1}, \ldots, t_{i_m} \} \) holds for all \( 1 \leq i \leq n \) and \( 1 \leq j < k \leq m \). This implies there will always be \( m \) \( T_1 T_2 \ldots T_n \)-successors \( \{ t_{i_1}, \ldots, t_{i_m} \} \) of \( x \) with \( P_n(i) \in \mathcal{L}(t_i) \) and \( \{ t_{j_1}, \ldots, t_{j_n} \} \neq \{ t_{i_1}, \ldots, t_{i_m} \} \) holds for all \( 1 \leq i \leq n \) and \( 1 \leq j < k \leq m \), since the \( \leq \), \( \mathbf{O} \)- and \( \leq^p \)-rule can never merge them, and whenever an application of the \( \leq_p \)-rule sets some \( \mathcal{L}(t_{j_i}) \) to \( \emptyset \), then there will be some \( T_1 T_2 \ldots T_n \)-successors \( \{ t_{i_1}, \ldots, t_{i_m} \} \) of \( x \) with \( P_n(i) \in \mathcal{L}(t_i) \) and \( \{ t_{j_1}, \ldots, t_{j_n} \} \) “inherits” all inequalities from \( \{ t_{j_1}, \ldots, t_{j_n} \} \). Hence the out-degree of the forest is bounded by \( d \cdot n_{\text{max}} \).

3. Nodes are labelled with subsets of \( c(D) \cup \{ \mathbf{t}(R, \{ o \}) \} \mid R \in R^D_A \) and \( \{ o \} \in T^D \), and the concrete value nodes are always leaves, so there are at most \( 2^{2^{|c(D)|} \cdot |T^D|} \) different node labelings. Therefore, if a path \( p \) is of length at least \( 2^{2^{|c(D)|} \cdot |T^D|} \), then, from the blocking condition above, there are two nodes \( x, y \) on \( p \) such that \( x \) is directly blocked by \( y \). Hence paths are of length at most \( 2^{2^{|c(D)|} \cdot |T^D|} \).

\textbf{■}

\textbf{Lemma 7 (Soundness)} If the expansion rules can be applied to a \( \mathcal{SHCQ}(D_n) \)-concept \( D \) in NNF and a role box \( \mathcal{R} \) such that they yield a complete and clash-free completion forest, then \( D \) has a tableau w.r.t. \( \mathcal{R} \).

\textbf{Proof:} Let \( \mathcal{F} \) be the complete and clash-free completion forest constructed by the tableau algorithm for \( D \). To cope with cycle, an individual in \( S \) corresponds to a \textit{path} in \( \mathcal{F} \). Due to qualifying number restrictions, we must distinguish different nodes that are blocked by the same node. We refer the readers to [9] for the definitions of path and related concepts. We can define a tableau \( \mathcal{T} = \{ S, \mathcal{L}, \mathcal{E}_A, \mathcal{E}_D \} \) with: \( S = \text{Paths}(\mathcal{F}) \), \( \mathcal{L}(p) = \mathcal{L}(\text{Tail}(p)) \), \( \mathcal{E}_A(R_A) = \{ (p, q) \in S \times S \mid q = [p(x, x')] \) and \( x' \) is an \( R_A \)-successor of \( \text{Tail}(p) \} \), \( \mathcal{E}_D(\mathcal{R}_D) = \{ (p, t) \in S \times \Delta_D \mid t \) is an \( \mathcal{R}_D \)-successor of \( \text{Tail}(p) \} \).

We have to show that \( \mathcal{T} \) satisfies (P14) to (P17) from Definition 3.

\begin{itemize}
  \item (P14): Assume \( \geq m T_1, \ldots, T_n.P_n \in \mathcal{L}(p) \). This implies that in \( \mathcal{F} \) there exist \( m \) \( T_1 T_2 \ldots T_n \)-successors \( \{ t_{i_1}, \ldots, t_{i_m} \} \), \( \{ t_{j_1}, \ldots, t_{j_m} \} \) of \( \text{Tail}(p) \) and \( P_n(i) \in \mathcal{L}(t_i) \) for all \( 1 \leq i \leq n, 1 \leq j \leq m \). We claim that, for each of these concrete nodes, according to the construction of \( \mathcal{E}_D \) above, we have \( \langle p, t_{i_j} \rangle \in \mathcal{E}_D(T_i) \), and \( \langle t_{i_1}, \ldots, t_{i_m} \rangle \neq \langle t_{j_1}, \ldots, t_{j_m} \rangle \) and \( \langle p, \{ t_{j_1}, \ldots, T_n \}, \{ t_{j_1}, \ldots, t_{j_m}, \} \rangle \in \mathcal{D}^D \) for all \( 1 \leq i \leq n \) and \( 1 \leq j < k \leq m \)(otherwise, \( \geq_p \)-rule was still applicable). According to the definition of \( \mathcal{D}^D \) and \( T_1 T_2 \ldots T_n (p, P_n) \), this implies \( \mathcal{T} T_1 T_2 \ldots T_n (p, P_n) \geq m \).
\end{itemize}
• (P15): Assume (P15) doesn’t hold. Hence there is some \( p \in S \) with \( (\leq m T_1, \ldots, T_n, P_n) \in \mathcal{L}(p) \) and \( \exists \mathcal{T}_1 T_2 \ldots T_n^T (p, P_n) > m \). According to the definition of \( T_1 T_2 \ldots T_n^T (p, P_n) \), let \( \text{value}(t_j) \) be the value of node \( t_j \); this implies that there exist \( (t_1, \ldots, t_n), \ldots, (t_{m+1}, \ldots, t_{m+n}) \) such that \( (p, \text{value}(t_j)) \in E_D(T_1), \ldots, (p, (T_1, \ldots, T_n), (t_{j+1}, \ldots, t_{j+n}), P_n) \in D_C^x \), for all \( 1 \leq i \leq m \) and \( 1 \leq j < k \leq m + 1 \). Therefore the \( \leq \)-rule is still applicable, which is a contradiction to the completeness of \( F \). Thus the assumption \( \exists \mathcal{T}_1 T_2 \ldots T_n^T (p, P_n) > m \) is false. So we have \( \exists \mathcal{T}_1 T_2 \ldots T_n^T (p, P_n) \leq m \).

• (P16): Assume \( \{ \leq m T_1, \ldots, T_n, P_n, \geq m T_1, \ldots, T_n, P_n \} \cap \mathcal{L}(p) \neq \emptyset, (p, t_i) \in E_D(T_1), 1 \leq i \leq n, \) thus \( (t_1, \ldots, t_n) \) is a \( T_1 T_2 \ldots T_n \)-successors of \( \text{Tail}(p) \). Let \( \text{value}(t_i) \) be the value of \( t_i \): (1) if \( \{ (p, (T_1, \ldots, T_n), (t_1, \ldots, t_n), P_n) \} \in D_C^x \), we have \( D_C^x (p, (T_1, \ldots, T_n), \text{value}(t_i), \ldots, \text{value}(t_n), P_n) = \text{true} \); (2) if \( \{ (p, (T_1, \ldots, T_n), (t_1, \ldots, t_n), P_n) \} \in D_C^x \), we have \( D_C^x (p, (T_1, \ldots, T_n), \text{value}(t_i), \ldots, \text{value}(t_n), P_n) = \text{true} \).

Lemma 8 (Completeness) If a \( SHOQ(D,n) \)-concept \( D \) in NNF has a tableau w.r.t. \( P \), then the expansion rules can be applied to \( D \) and \( P \) such that they yield a complete, clash-free completion forest.

Proof: Let \( T = (S, \mathcal{L}, E_A, E_D) \) be a tableau for \( D \) w.r.t. a role box \( R \). We use \( T \) to guide the application of the non-deterministic rules. We define a function \( \pi \), mapping the nodes of the forest \( F \) to \( S \cup \Delta_P \) such that \( \mathcal{L}(x) \subseteq \mathcal{L}(\pi(x)) \): \( (\pi(x), \pi(y)) \in E_A \) if: 1. \( \pi(y) \in S \) and \( y \) is an \( R_A \)-successor of \( x \), or 2. \( \uparrow (R, \{ y \}) \in \mathcal{L}(x) \) and \( y = x_{\{ y \}}(\pi(x), \pi(y)) \in E_P \) if \( \pi(y) \not\in S \) and \( y \) is an \( R_D \)-successor of \( x \): \( x \neq y \) implies \( \pi(x) \neq \pi(y) \); \( \langle y_1, \ldots, y_n \rangle \not\in \langle \pi(y_1), \ldots, \pi(y_m) \rangle \) for \( y_1, \ldots, y_n, \pi(y_1), \ldots, \pi(y_m) \in S \). (*)

We only have to consider the various rules about number restriction on concrete roles.

• The \( \geq \)-rule: If \( \geq m T_1, \ldots, T_n, P_n \in \mathcal{L}(x) \), then \( \geq m T_1, \ldots, T_n, P_n \in \mathcal{L}(\pi(x)) \). Since \( T \) is a tableau, (P14) of Definition 3 implies \( T_1 T_2 \ldots T_n^T (\pi(x), P_n) \geq m \). Hence there are \( m \) tuples \( (t_1, \ldots, t_n), \ldots, (t_{m+1}, \ldots, t_{m+n}) \), such that \( (\pi(x), t_j) \in E_D, (t_{j+1}, \ldots, t_n) \not\in \langle t_1, \ldots, t_m \rangle \), and \( D_C^x (\pi(x), t_1, \ldots, T_n, t_{j+1}, \ldots, t_n, P_n) = \text{true} \), for \( 1 \leq i \leq m \) and \( 1 \leq j < k \leq m \). The \( \geq \)-rule generates \( m+1 \) \( T_1 T_2 \ldots T_n \)-successors \( (y_1, \ldots, y_n), \ldots, (y_{m+1}, \ldots, y_{m+n}) \). By setting \( \pi(x) \mapsto \pi(y_i) \mapsto t_j \), \( 1 \leq i \leq m, 1 \leq j < k \leq m \), one obtains a function \( \pi(x) \) that satisfies (*) for the modified forest.

• The \( \leq \)-rule: If \( \leq m T_1, \ldots, T_n, P_n \in \mathcal{L}(x) \), then \( \leq m T_1, \ldots, T_n, P_n \in \mathcal{L}(\pi(x)) \). Since \( T \) is a tableau, (P15) of Definition 3 implies \( \leq m T_1 T_2 \ldots T_n^T (\pi(x), P_n) \leq m \). If the \( \leq \)-rule is applicable, we have \( D_C^x (\pi(x), T_1, \ldots, T_n, P_n) > m \), which implies that there are at least \( m+1 \) \( T_1 T_2 \ldots T_n \)-successors \( (y_1, \ldots, y_n), \ldots, (y_{m+1}, \ldots, y_{m+n}) \) such that \( \{ (x, (T_1, \ldots, T_n), (y_{j+1}, \ldots, y_{j+n}), P_n) \} \in D_C^x \), for \( 1 \leq j \leq m + 1 \). Thus, there must be two \( y_{j_1}, y_{j_2} \) among the \( m+1 \) \( T_1 T_2 \ldots T_n \)-successors such that \( (\pi(y_j), \ldots, \pi(y_{j+n})) = (\pi(y_{j_1}), \ldots, \pi(y_{j+n})) \) otherwise \( \pi(T_1, \ldots, T_n^T (\pi(x), P_n) \geq m \) would hold). This implies \( (y_1, \ldots, y_n) \not\in \langle \pi(x), \pi(y) \rangle \) cannot hold because of (*). Hence the \( \leq \)-rule can be applied without violating (*).

• The \( \text{choose} \)-rule: If \( \{ \leq m T_1, \ldots, T_n, P_n, \geq m T_1, \ldots, T_n, P_n \} \cap \mathcal{L}(x) \neq \emptyset \), we have \( \{ \leq m T_1, \ldots, T_n, P_n, \geq m T_1, \ldots, T_n, P_n \} \cap \mathcal{L}(\pi(x)) \neq \emptyset \), and if there are \( T_1 T_2 \ldots T_n \)-successors \( (y_1, \ldots, y_n) \) of \( x \), then \( (\pi(x), \pi(y)) \in E_D, 1 \leq i \leq n \), due to (*). Since \( T \) is a tableau, (P16) of Definition 3 implies either \( D_C^x (\pi(x), T_1, \ldots, T_n, \pi(y_1), \ldots, \pi(y_n), P_n) = \text{true} \), or \( D_C^x (\pi(x), T_1, \ldots, T_n, \pi(y), \ldots, \pi(y_{j+n}), P_n) = \text{true} \). Hence the \( \text{choose} \)-rule can accordingly either set \( D_C^x \rightarrow D_C^x \cup \{ x, (T_1, \ldots, T_n), (y_1, \ldots, y_n), P_n \} \) or set \( D_C^x \rightarrow D_C^x \cup \{ x, (T_1, \ldots, T_n), (y_1, \ldots, y_n), (\pi(y), \ldots, \pi(y_{j+n})), P_n \} \).

Whenever a rule is applicable to \( F \), it can be applied in a way that maintains (*), and, from Lemma 6, we have that any sequence of rule applications must terminate. Since (*), holds, any forest generated by these rule-applications must be clash-free. This can be seen from the condition described in [7] plus the following:

• If \( F \) does not satisfy \( D_C^x \), there must be some concrete nodes from which no values mapping satisfies all the relevant predicates, and therefore there can be no values satisfying all of properties (P12) to (P16).
• F cannot contain a node $x$ with $\preceq mT_1, \ldots, T_n, P_n \in L(x)$, and $m + 1 \ T_1, \ldots, T_n$-successors $(t_{i_1}, \ldots, t_{i_1}), \ldots, (t_{m+1}, \ldots, t_{m+1})$ of $x$ with $P_n(t_i) \in L(t_i)$, $(t_{j_1}, \ldots, t_{j_m}) \neq (t_{k_1}, \ldots, t_{k_n})$ and $DCF(x, t_{j_1}, \ldots, t_{j_m}, P_n) = true$, for all $1 \leq i \leq n$, $1 \leq k \leq m + 1$, and, since $(t_{j_1}, \ldots, t_{j_m}) \neq (t_{k_1}, \ldots, t_{k_n})$ implies $(\pi(t_{j_1}), \ldots, \pi(t_{j_m})) \neq (\pi(t_{k_1}), \ldots, \pi(t_{k_n}))$, $T \cap T_1, \ldots, T_n (\pi(x), P_n) > n$ would hold which contradicts (P10) of Definition 3.

As an immediate consequence of Lemmas 2,4,5 and 6, the completion algorithm always terminates, and answers with "D is satisfiable w.r.t. $\mathcal{R}$" iff. $D$ has a tableau $T$. Next, subsumption can be reduced to (un)satisfiability. Finally, $SHOQ(D_n)$ can internalise general concept inclusion axioms [5]. However, in the presence of nominals, we must also add $\exists O.o_1 \cap \ldots \cap \exists O.o_i$ to the concept internalising the general concept inclusion axioms to make sure that the universal role $O$ indeed reaches all nominals $O_i$ occurring in the input concept and terminology. Thus, we can decide these inference problems also w.r.t. terminologies.

**Theorem 9** The tableau algorithm presented in Definition 5 is a decision procedure for satisfiability and subsumption of $SHOQ(D_n)$-concepts w.r.t. terminologies.

5 Discussion

As we have seen, unary datatype predicates are usually not enough, while n-ary datatype predicates are often necessary in modelling the "concrete properties" of real world entities. Furthermore, datatype number restrictions are very expressive that e.g., with them, we can define single/multiple-value datatype attributes. Therefore, we have extended $SHOQ(D)$ with n-ary datatype predicates and datatype number restrictions to give the $SHOQ(D_n)$ DL.

We have shown that the decision procedure for concept satisfiability and subsumption is still decidable in $SHOQ(D_n)$. An implementation based on the FaCT system is planned, and will be used to test empirical performance.

With its support for both nominals and n-ary datatype predicates with datatype number restrictions, $SHOQ(D_n)$ is well suited to provide reasoning support for ontology languages in general, and Semantic Web ontology languages in particular. As future work, it would be interesting to study the datatype number restrictions in the Semantic Web applications. It is also important to extend current optimisation techniques to cope with nominals used in the logic. The $SHOQ(D_n)$ DL decision procedure is similar to those of the $SHIQ$ DL implemented in the successful FaCT system, and should be amenable to a similar range of performance enhancing optimisations. Thirdly, ABox reasoning and query answering in $SHOQ(D_n)$ are also very interesting, since these efforts will make more reasoning services available, e.g., querying services, to the Web ontology languages, such as DAML+OIL.

Acknowledgements

We would like to thank Ulrike Sattler, since the work presented here extends the original work on $SHOQ(D)$. Thanks are also due to Carsten Lutz and Volker Haarslev for their helpful discussion on concrete datatypes.

References


