Ontology Reasoning in the $\mathcal{SHOQ}(\mathrm{D})$ Description Logic

Ian Horrocks
Department of Computer Science
University of Manchester, UK
horrocks@cs.man.ac.uk

Ulrike Sattler
LuFg Theoretical Computer Science
RWTH Aachen, Germany
sattler@informatik.rwth-aachen.de

Abstract

Ontologies are set to play a key rôle in the "Semantic Web" by providing a source of shared and precisely defined terms that can be used in descriptions of web resources. Reasoning over such descriptions will be essential if web resources are to be more accessible to automated processes. $\mathcal{SHOQ}(\mathbf{D})$ is an expressive description logic equipped with named individuals and concrete datatypes which has almost exactly the same expressive power as the latest web ontology languages (e.g., OIL and DAML). We present sound and complete reasoning services for this logic.

1 Introduction

The recent explosion of interest in the World Wide Web has also fuelled interest in ontologies.[1] This is due both to the use of ontologies in existing Web based applications and to their likely rôle in the future development of the Web [van Heijst *et al.*, 1997; McGuinness, 1998; Uschold and Grüninger, 1996]. In particular, it has been predicted that ontologies will play a pivotal rôle in the *Semantic Web*—the World Wide Web Consortium's vision of a "second generation" Web in which Web resources will be more readily accessible to automated processes [Berners-Lee, 1999].

A key component of the Semantic Web will be the annotation of web resources with meta-data that describes their content, with ontologies providing a source of shared and precisely defined terms that can be used in such meta-data. This requirement has led to the extension of Web markup languages in order to facilitate content description and the development of Web based ontologies, e.g., XML Schema, RDF (Resource Description Framework), and RDF Schema [Decker *et al.*, 2000]. RDF Schema (RDFS) in particular is recognisable as an ontology/knowledge representation language: it talks about classes and properties (binary relations), range and domain constraints (on properties), and subclass and subproperty (subsumption) relations. However, RDFS is a very primitive language (the above is an almost complete description of its functionality), and more expressive power would clearly be necessary/desirable in order to describe resources in sufficient detail. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes.

These considerations have led to the development of OIL [Fensel *et al.*, 2000] and DAML [Hendler and McGuinness, 2001], two ontology languages that extend RDFS with a much richer set of modelling primitives. Both languages have been designed in such a way that they can be mapped onto a very expressive description logic (DL).[2] This mapping provides them with a formal semantics, a clear understanding of the characteristics of various reasoning problems (e.g., subsumption/satisfiability), and the possibility of exploiting existing decision procedures. OIL, in particular, was designed so that reasoning services could be provided, via a mapping to the $\mathcal{SHIQ}$ DL, by the FaCT system [Horrocks *et al.*, 1999; Horrocks, 2000].

Unfortunately, these mappings are currently incomplete in two important respects. Firstly, any practical ontology language will need to deal with *concrete datatypes* (numbers, strings, etc.) [Baader and Hanschke, 1991]. E.g., ontologies used in e-commerce may want to classify items according to weight, and to reason that an item weighing more than 50 kilogrammes is a kind of item that requires special shipping arrangements. OIL already supports the use of integers and strings in class descriptions, and it is anticipated that DAML+OIL, a new language developed from a merging of the DAML and OIL efforts, will support (most of) the datatypes defined or definable by XML Schema. However, the $\mathcal{SHIQ}$ logic implemented in the FaCT system does not include any concrete datatypes, so there is no mechanism for reasoning with this part of the language.

Secondly, realistic ontologies typically contain references to named individuals within class descriptions. E.g., "Italians" might be described as persons who are citizens of "Italy", where Italy is a named individual [Schaerf, 1994]. The required functionality can be partially simulated by treating such individuals as pairwise disjoint atomic classes (this is the approach taken in the existing OIL $\longrightarrow$ FaCT mapping), but this can result in incorrect inferences.

In this paper we will present a new DL that overcomes

[1] The word ontology has been used—some would say abused—in a wide range of contexts. In this paper it will be taken to mean a formally defined model of (part of) the domain of interest.

[2] In fact they can be viewed as syntactic variants of such a logic.

both of the above deficiencies by taking the logic $\mathcal{SHQ}$ and extending it with individuals ($\mathcal{O}$) and concrete datatypes ($\mathbf{D}$) to give $\mathcal{SHOQ}(\mathbf{D})$. The starting point for these extensions is $\mathcal{SHQ}$ rather than $\mathcal{SHIQ}$ (i.e., without inverse roles), because reasoning with inverse roles is known to be difficult and/or highly intractable when combined with either concrete datatypes or named individuals: the concept satisfiability problem is know to be NExpTime hard even for the basic DL $\mathcal{ALC}$ augmented with inverse roles and either concrete datatypes or named individuals [Lutz, 2000; Tobies, 2000]. This hardness result for concrete datatypes is not yet directly applicable to $\mathcal{SHOQ}(\mathbf{D})$ as it depends on comparisons of concrete values (binary predicates), but the addition of such comparisons would be a natural future extension to $\mathcal{SHOQ}(\mathbf{D})$. Moreover, the presence of nominals in any DL leads to the loss of the tree/forest model property, which becomes particularly problematical in the presence of inverse roles, number restrictions, and general axioms. As a result, to the best of our knowledge, there is no (practicable) decision procedure for $\mathcal{SHIQ}$ with nominals or converse-DPDL with nominals, the latter being a close relative of $\mathcal{SHIQ}$ from dynamic logics [Streett, 1982]. Finally, since individuals and concrete datatypes are much more widely used in ontologies than inverse roles [Corcho and Pérez, 2000], $\mathcal{SHOQ}(\mathbf{D})$ is a very useful addition to our reasoning armoury.

2 Preliminaries

In this section, we will describe our choice of concrete datatypes and named individuals, and introduce the syntax and semantics of $\mathcal{SHOQ}(\mathbf{D})$.

Concrete Datatypes Concrete datatypes are used to represent literal values such as numbers and strings. A type system typically defines a set of "primitive" datatypes, such as *string* or *integer*, and provides a mechanism for deriving new datatypes from existing ones. For example, in the XML schema type system the *nonNegativeInteger* datatype is derived from the *integer* datatype by constraining values of *nonNegativeInteger* to be greater than or equal to zero [Biron and Malhorta, 2000].

In order to represent concepts such as "persons whose age is at least 21", we can extend our concept language with a set $\mathbf{D}$ of concrete datatypes and concepts of the form $\exists R.d$ and $\forall R.d$, where $d \in \mathbf{D}$. To be more precise, we assume that we have a set of datatypes $\mathbf{D}$, and, with each $d \in \mathbf{D}$, a set $d^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$ is associated, where $\Delta_{\mathbf{D}}$ is the domain of all datatypes. We will assume that:

1. the domain of interpretation of all concrete datatypes $\Delta_{\mathbf{D}}$ (the *concrete domain*) is disjoint from the domain of interpretation of our concept language (the *abstract domain*), and
2. there exists a sound and complete decision procedure for the emptiness of an expression of the form $d_1^{\mathbf{D}} \cap \ldots \cap d_n^{\mathbf{D}}$, where d_i is a (possibly negated) concrete datatype from $\mathbf{D}$ (where $\neg d$ is interpreted as $\Delta_{\mathbf{D}} \setminus d^{\mathbf{D}}$).

We will say that a set of datatypes is *conforming* if it satisfies the above criteria.

The disjointness of the abstract and concrete domains is motivated by both philosophical and pragmatic considerations. On the one hand, concrete datatypes are considered to be already sufficiently structured by the type system, which may include a derivation mechanism and built-in ordering relations; therefore, we do not need the DL mechanism to form new datatypes as in [Baader and Hanschke, 1991]. On the other hand, it allows us to deal with an arbitrary conforming set of datatypes without compromising the compactness of our concept language or the soundness and completeness of our decision procedure.

This scheme can be trivially extended to include boolean combinations of datatypes and number restrictions qualified with data types, but to simplify the presentation we will only consider (possibly negated) atomic datatypes and exists/value restrictions. The type system can be as complex as that defined for XML schema, or as simple as the one defined in the OIL ontology language [Fensel *et al.*, 2000], where the only primitive datatypes are integer and string, and new types are derived by adding minimum and maximum value constraints. Using the OIL typesystem we could, for example, define the type (min 21) and use it in the concept $\mathsf{Person} \sqcap \exists \mathit{age}.(\text{min } 21)$.

Named Individuals Allowing named individuals to occur in concepts provides additional expressive power that is useful in many applications; *nominals* (as such individuals can be called) are a prominent feature of hybrid logics [Blackburn and Seligman, 1998], and various extensions of modal and description logics with nominals have already been investigated (see, e.g., [Schaerf, 1994; De Giacomo, 1995; Areces *et al.*, 2000]). As we have seen, nominals occur naturally in ontologies as names for specific persons, companies, countries etcetera.

From a semantic point of view, it is important to distinguish between a nominal and an atomic concept/simple class, since the nominal stands for exactly one individual—in contrast to a concept, which is interpreted as some *set* of individuals. Modelling nominals as pairwise disjoint atomic concepts can lead to incorrect inferences, in particular with respect to implicit maximum cardinality constraints. For example, if Italy is modelled as an atomic concept, then it would not be possible to infer that persons who are citizens *only* of Italy cannot have dual-nationality (i.e., cannot be citizens of more than one country).

Finally, nominals can be viewed as a powerful generalisation of DL *Abox individuals* [Schaerf, 1994]: in an Abox we can assert that an individual is an instance of a concept or that a pair of individuals is an instance of a role (binary relation), but Abox individuals cannot be used *inside* concepts. For example, if Giuseppe and Italy are Abox individuals, we could assert that the pair (Giuseppe, Italy) is an instance of the *citizen-of* role, but we could not describe the concept Italian as a Person who is a *citizen-of* Italy. Using nominals, not only can we express this concept (i.e., $\mathsf{Person} \sqcap \exists \mathit{citizen\text{-}of}.\mathsf{Italy}$), but we can also capture Abox assertions with concept inclusion axioms of the form $\mathsf{Giuseppe} \sqsubseteq \mathsf{Italian}$ (Giuseppe is an Italian) and $\mathsf{Giuseppe} \sqsubseteq \exists \mathit{citizen\text{-}of}.\mathsf{Italy}$ (Giuseppe is a *citizen-of* Italy).

Construct Name	Syntax	Semantics
atomic concept $\mathbf{C}$	A	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
abstract role $\mathbf{R}_A$	R	$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
concrete role $\mathbf{R}_\mathbf{D}$	T	$T^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathbf{D}$
nominals $\mathbf{I}$	o	$o^\mathcal{I} \subseteq \Delta^\mathcal{I}$, $\sharp o^\mathcal{I} = 1$
datatypes $\mathbf{D}$	d	$d^\mathbf{D} \subseteq \Delta_\mathbf{D}$
conjunction	$C \sqcap D$	$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$
negation	$\neg C$	$(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$
	$\neg d$	$(\neg d)^\mathcal{I} = \Delta_\mathbf{D} \setminus d^\mathcal{I}$
exists restriction	$\exists R.C$	$(\exists R.C)^\mathcal{I} = \{x \mid \exists y. \langle x,y\rangle \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$
value restriction	$\forall R.C$	$(\forall R.C)^\mathcal{I} = \{x \mid \forall y. \langle x,y\rangle \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$
atleast restriction	$(\geqslant nS.C)$	$(\geqslant nS.C)^\mathcal{I} = \{x \mid \sharp(\{y. \langle x,y\rangle \in S^\mathcal{I}\} \cap C^\mathcal{I}) \geqslant n\}$
atmost restriction	$(\leqslant nS.C)$	$(\leqslant nS.C)^\mathcal{I} = \{x \mid \sharp(\{y. \langle x,y\rangle \in S^\mathcal{I}\} \cap C^\mathcal{I}) \leqslant n\}$
datatype exists	$\exists T.d$	$(\exists T.d)^\mathcal{I} = \{x \mid \exists y. \langle x,y\rangle \in T^\mathcal{I}$ and $y \in d^\mathbf{D}\}$
datatype value	$\forall T.d$	$(\forall T.d)^\mathcal{I} = \{x \mid \forall y. \langle x,y\rangle \in T^\mathcal{I}$ implies $y \in d^\mathbf{D}\}$

Figure 1: Syntax and semantics of $\mathcal{SHOQ}(\mathbf{D})$

$\mathcal{SHOQ}(\mathbf{D})$ Syntax and Semantics

Definition 1 Let $\mathbf{C}$, $\mathbf{R} = \mathbf{R}_A \uplus \mathbf{R}_\mathbf{D}$, and $\mathbf{I}$ be disjoint sets of *concept names*, abstract and concrete *role names*, 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 $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_\mathbf{D}$, or a transitivity axiom, which is of the form $\mathsf{Trans}(R)$ for $R \in \mathbf{R}_A$. A *role box* $\mathcal{R}$ is a finite set of role axioms.

A role R is called *simple* if, for $\underline{\circledast}$ the transitive reflexive closure of $\sqsubseteq$ on $\mathcal{R}$ and for each role S, $S \,\underline{\circledast}\, R$ implies $\mathsf{Trans}(S) \notin \mathcal{R}$.

The set of $\mathcal{SHOQ}(\mathbf{D})$-concepts is the smallest set such that each concept name $A \in \mathbf{C}$ is a concept, for each individual name $o \in \mathbf{I}$, o is a concept, and, for C and D concepts, R an abstract role, T a concrete role, S a simple role, and $d \in \mathbf{D}$ a concrete datatype, complex concepts can be built using the operators shown in Figure 1.

The semantics is given by means of an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consisting of a non-empty domain $\Delta^\mathcal{I}$, disjoint from the concrete domain $\Delta_\mathbf{D}$, and a mapping $\cdot^\mathcal{I}$, which maps atomic and complex concepts, roles, and nominals according to Figure 1 ($\sharp$ denotes set cardinality). An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *satisfies* a role inclusion axiom $R_1 \sqsubseteq R_2$ iff $R_1^\mathcal{I} \subseteq R_2^\mathcal{I}$, and it satisfies a transitivity axiom $\mathsf{Trans}(R)$ iff $R^\mathcal{I} = (R^\mathcal{I})^+$. An interpretation satisfies a role box $\mathcal{R}$ iff it satisfies each axiom in $\mathcal{R}$.

A $\mathcal{SHOQ}(\mathbf{D})$-concept C is *satisfiable* w.r.t. a role box $\mathcal{R}$ iff there is an interpretation $\mathcal{I}$ with $C^\mathcal{I} \neq \emptyset$ that satisfies $\mathcal{R}$. Such an interpretation is called a *model* of C w.r.t. $\mathcal{R}$. A concept C is *subsumed* by a concept D w.r.t. $\mathcal{R}$ iff $C^\mathcal{I} \sqsubseteq D^\mathcal{I}$ for each interpretation $\mathcal{I}$ satisfying $\mathcal{R}$. Two concepts are said to be equivalent (w.r.t. $\mathcal{R}$) iff they mutually subsume each other (w.r.t. $\mathcal{R}$).

Some remarks are in order: In the following, if $\mathcal{R}$ is clear from the context, we use $\mathsf{Trans}(R)$ instead of $\mathsf{Trans}(R) \in \mathcal{R}$.

Please note that the domain of each role is the abstract domain, and that we distinguish those roles whose range is also the abstract domain (*abstract roles*), and those whose range is the concrete domain (*concrete roles*). In the following, we use R for the former and T for the latter form of roles (possibly with index). We have chosen to disallow role inclusion axioms of the form $T \sqsubseteq R$ (or $R \sqsubseteq T$) for R an abstract and T a concrete role, since each model of such an axiom would necessarily interpret T (or R) as the empty relation.

Restricting number restrictions to simple roles is required to yield a decidable logic [Horrocks *et al.*, 1999].

Next, negation of concepts and datatypes is relativised to both the abstract and the concrete domain.

As usual, subsumption and satisfiability can be reduced to each other, and $\mathcal{SHOQ}(\mathbf{D})$ has the expressive power to *internalise* general concept inclusion axioms [Horrocks *et al.*, 1999]. However, in the presence of nominals, we must also add $\exists O.o_1 \sqcap \ldots \sqcap \exists O.o_\ell$ 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.

Finally, we did not choose to make a *unique name assumption*, i.e., two nominals might refer to the same individual. However, the inference algorithm presented below can easily be adapted to the unique name case by a suitable initialisation of the inequality relation $\neq$.

3 A Tableau for $\mathcal{SHOQ}(\mathbf{D})$

For ease of presentation, we assume all concepts to be in *negation normal form* (NNF). Each concept can be transformed into an equivalent one in NNF by pushing negation inwards, making use of deMorgan's laws and the following equivalences:

$$\begin{array}{rclcrcl}
\neg\exists R.C & \equiv & \forall R.\neg C & & \neg\forall R.C & \equiv & \exists R.\neg C \\
\neg\exists T.d & \equiv & \forall T.\neg d & & \neg\forall T.d & \equiv & \exists T.\neg d \\
\neg(\leqslant nR.C) & \equiv & (\geqslant (n+1)R.C) \\
\neg(\geqslant (n+1)R.C) & \equiv & (\leqslant nR.C) \\
\neg(\geqslant 0R.C) & \equiv & C \sqcap \neg C
\end{array}$$

We use $\sim C$ to denote the NNF of $\neg C$. Moreover, for a concept D, we use $\mathsf{cl}(D)$ to denote the set of all subconcepts of D, the NNF of these subconcepts, and the (possibly negated) datatypes occurring in these (NNFs of) subconcepts.

Definition 2 If D is a $\mathcal{SHOQ}(\mathbf{D})$-concept in NNF, $\mathcal{R}$ a role box, and $\mathbf{R}_A^D$, $\mathbf{R}_\mathbf{D}^D$ are the sets of abstract and concrete roles occurring in D or $\mathcal{R}$, a tableau T for D w.r.t. $\mathcal{R}$ is defined to be a quadruple $(\mathbf{S}, \mathcal{L}, \mathcal{E}_A, \mathcal{E}_\mathbf{D})$ such that: $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{\mathsf{cl}(D)}$ maps each individual to a set of concepts which is a subset of $\mathsf{cl}(D)$, $\mathcal{E}_A : \mathbf{R}_A^D \to 2^{\mathbf{S}\times\mathbf{S}}$ maps each abstract role in $\mathbf{R}_A^D$ to a set of pairs of individuals, $\mathcal{E}_\mathbf{D} : \mathbf{R}_\mathbf{D}^D \to 2^{\mathbf{S}\times\Delta_\mathbf{D}}$ maps each concrete role in $\mathbf{R}_\mathbf{D}^D$ to a set of pairs of individuals and concrete values, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s, t \in \mathbf{S}$, $C, C_1, C_2 \in \mathsf{cl}(D)$, $R, S \in \mathbf{R}_A^D$, $T, T' \in \mathbf{R}_\mathbf{D}^D$, and

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

it holds that:

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

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

(P3) if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, 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 $R \sqsubseteq\!\!\!* \, S$, then $\langle s,t\rangle \in \mathcal{E}_A(S)$, if $\langle s,t\rangle \in \mathcal{E}_{\mathbf{D}}(T)$ and $T \sqsubseteq\!\!\!* \, T'$, then $\langle s,t\rangle \in \mathcal{E}_{\mathbf{D}}(T')$

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

(P6) if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ 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 $\langle s,t\rangle \in \mathcal{E}_A(R)$ for some $R \sqsubseteq\!\!\!* \, S$ with $\mathsf{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,

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

(P9) if $(\leqslant nS.C) \in \mathcal{L}(s)$, then $\sharp S^T(s,C) \leqslant n$, and

(P10) if $\{(\leqslant nS.C), (\geqslant nS.C)\} \cap \mathcal{L}(s) \neq \emptyset$ and $\langle s,t\rangle \in \mathcal{E}_A(S)$, then $\{C, \sim C\} \cap \mathcal{L}(t) \neq \emptyset$,

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

(P12) if $\forall T.d \in \mathcal{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}_{\mathbf{D}}(T)$, then $t \in d^{\mathbf{D}}$,

(P13) if $\exists T.d \in \mathcal{L}(s)$, then there is some $t \in \Delta_{\mathbf{D}}$ such that $\langle s,t\rangle \in \mathcal{E}_{\mathbf{D}}(T)$ and $t \in d^{\mathbf{D}}$.

Lemma 3 A $\mathcal{SHOQ}(\mathbf{D})$-concept D in NNF is satisfiable w.r.t. a role box $\mathcal{R}$ iff D has a tableau w.r.t. $\mathcal{R}$.

Proof: We concentrate on (P11) to (P13), which cover the the new logical features, i.e., nominals and datatypes; the remainder is similar to the proof found in [Horrocks *et al.*, 1999]. Roughly speaking, we construct a model $\mathcal{I}$ from a tableau by taking $\mathbf{S}$ as its interpretation domain and adding the missing role-successorships for transitive roles. Then, by induction on the structure of formulae, we prove that, if $C \in \mathcal{L}(s)$, then $s \in C^{\mathcal{I}}$. (P11) ensures that nominals are indeed interpreted as singletons, and (P12) and (P13) make sure that concrete datatypes are interpreted correctly.

For the converse, each model is by definition of the semantics a tableau. □

4 A tableau algorithm for $\mathcal{SHOQ}(\mathbf{D})$

From Lemma 3, an algorithm which constructs a tableau for a $\mathcal{SHOQ}(\mathbf{D})$-concept D can be used as a decision procedure for the satisfiability of D with respect to a role box $\mathcal{R}$. Such an algorithm will now be described in detail. Please note that, due to the absence of inverse roles, *subset blocking* is sufficient (see also [Baader and Sattler, 2000]) to ensure termination and correctness.

Definition 4 Let $\mathcal{R}$ be a role box, D a $\mathcal{SHOQ}(\mathbf{D})$-concept in NNF, $\mathbf{R}_A^D$ the set of abstract roles occurring in D or $\mathcal{R}$, and $\mathbf{I}^D$ the set of nominals occurring in D. A *completion forest* for D with respect to $\mathcal{R}$ is a set of trees F where each node x of the forest is labelled with a set

$$\mathcal{L}(x) \subseteq \mathsf{cl}(D) \cup \{\uparrow(R,o) \mid R \in \mathbf{R}_A^D \text{ and } o \in \mathbf{I}^D\},$$

and each edge $\langle x,y\rangle$ is labelled with a set of role names $\mathcal{L}(\langle x,y\rangle)$ containing roles occurring in $\mathsf{cl}(D)$ or $\mathcal{R}$. Additionally, we keep track of inequalities between nodes of the forest with a symmetric binary relation $\neq$ between the nodes of F. For each $o \in \mathbf{I}^D$ there is a *distinguished* node x_o in F such that $o \in \mathcal{L}(x)$. We use $\uparrow(R,o) \in \mathcal{L}(y)$ to represent an R labelled edge from y to x_o.

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 and $R' \in \mathcal{L}(\langle x,y\rangle)$, or $\uparrow(R',o) \in \mathcal{L}(x)$ and $y = x_o$. Ancestors and roots are defined as usual.

For a role S and a node x in F we define $S^{\mathsf{F}}(x,C)$ by

$$S^{\mathsf{F}}(x,C) := \{y \mid y \text{ is an } S\text{-successor of } x \text{ and } C \in \mathcal{L}(y)\}.$$

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 $\mathcal{L}(x) \subseteq \mathcal{L}(x')$. In this case we will say that x' blocks x. A node is *blocked* if is directly blocked or if its predecessor is blocked.

For a node x, $\mathcal{L}(x)$ is said to contain a *clash* if

1. for some concept name $A \in N_C$, $\{A, \neg A\} \subseteq \mathcal{L}(x)$,
2. for some role S, $(\leqslant nS.C) \in \mathcal{L}(x)$ and there are $n+1$ S-successors $y_0, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ for each $0 \leq i \leq n$ and $y_i \neq y_j$ for each $0 \leq i < j \leq n$,
3. $\mathcal{L}(x)$ contains (possibly negated) datatypes $d_1, \ldots, d_n$ such that $d_1^{\mathbf{D}} \cap \ldots \cap d_n^{\mathbf{D}}$ is empty, or if
4. for some $o \in \mathcal{L}(x)$, $x \neq x_o$.

If $o_1, \ldots, o_\ell$ are all individuals occurring in D, the algorithm initialises the completion forest F to contain $\ell+1$ root nodes $x_0, x_{o_1} \ldots, x_{o_\ell}$ with $\mathcal{L}(x_0) = \{D\}$ and $\mathcal{L}(x_{o_i}) = \{o_i\}$. The inequality relation $\neq$ is initialised with the empty relation. F is then expanded by repeatedly applying the rules from Figure 2, stopping if a clash occurs in one of its nodes.

The completion forest is *complete* when, for some node x, $\mathcal{L}(x)$ contains a clash, or when none of the 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 *satisfiable* w.r.t. $\mathcal{R}$", and "D is *unsatisfiable* w.r.t. $\mathcal{R}$" otherwise.

Lemma 5 When started with a $\mathcal{SHOQ}(\mathbf{D})$ concept D in NNF, the completion algorithm terminates.

Proof: Let $m = |\mathsf{cl}(D)|$, $k = |\mathbf{R}_A^D|$, n the maximal number in atleast number restrictions, and $\ell = |\mathbf{I}^D|$. Termination is a consequence of the following properties of the expansion rules: (1) Each rule but the $\leqslant$- 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 $\exists$- or the $\geqslant$-rule as successors of a node x for concepts of the form $\exists R.C$ and $(\geqslant nS.C)$ 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 $\leqslant$- or the $\mathbf{O}$-rule. If a successor y of x was generated for a concept $\exists S.C \in \mathcal{L}(x)$, and y is removed later, then there will always be some S-successor z of x such that $C \in \mathcal{L}(z)$, and hence the $\exists$-rule cannot be applied again to x and $\exists S.C$.

For the $\geqslant$-rule, if $y_1, \ldots, y_n$ were generated by an application of the $\geqslant$-rule for a concept $(\geqslant nS.C)$, then $y_i \neq y_j$ is added for each $i \neq j$. This implies that there will always

$\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$,
then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$,
then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\exists$-rule: if $\exists R.C \in \mathcal{L}(x)$, (or $\exists T.d \in \mathcal{L}(x)$) x is not blocked, and x has no R-successor y with $C \in \mathcal{L}(y)$
(resp. no T-successor y with $d \in \mathcal{L}(y)$),
then create a new node y with $\mathcal{L}(\langle x, y\rangle) = \{R\}$ and $\mathcal{L}(y) = \{C\}$ (resp. with $\mathcal{L}(\langle x, y\rangle) = \{T\}$ and $\mathcal{L}(y) = \{d\}$)
$\forall$-rule: if $\forall R.C \in \mathcal{L}(x)$ (or $\forall T.d \in \mathcal{L}(x)$), x is not blocked, and there is an R-successor y of x with $C \notin \mathcal{L}(y)$,
(resp. a T-successor y of x with $d \notin \mathcal{L}(y)$),
then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$ (resp. $\mathcal{L}(y) = \mathcal{L}(y) \cup \{d\}$)
$\forall_+$-rule: if $\forall S.C \in \mathcal{L}(x)$, x is not blocked, and there is some R with $\mathsf{Trans}(R)$ and $R \,\underline{\circledast}\, S$,
and an R-successor y of x with $\forall R.C \notin \mathcal{L}(y)$,
then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$
choose-rule: $\{(\geqslant nS.C), (\leqslant nS.C)\} \cap \mathcal{L}(x) \neq \emptyset$, x is not blocked, and y is an S-successor of x with $\{C, \sim C\} \cap \mathcal{L}(y) = \emptyset$,
then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \sim C\}$
$\geqslant$-rule: if $(\geqslant nS.C) \in \mathcal{L}(x)$, x is not blocked, and there are no n S-successors $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and
$y_i \neq y_j$ for $1 \leq i < j \leq n$,
then create n new nodes $y_1, \ldots, y_n$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}, \mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$\leqslant$-rule: if $(\leqslant nS.C) \in \mathcal{L}(x)$, x is not blocked, and x has $n+1$ S-successors $y_0, \ldots, y_n$ with $C \in \mathcal{L}(y_i)$ for
each $0 \leq i \leq n$, and there exist $i \neq j$ s. t. not $y_i \neq y_j$ and, if only one of y_i, y_j is distinguished, then it is y_i,
then 1. $\mathcal{L}(y_i) = \mathcal{L}(y_i) \cup \mathcal{L}(y_j)$ and add $y \neq y_i$ for each y with $y \neq y_j$, and
if both y_i, y_j are not distinguished, then 2. $\mathcal{L}(\langle x, y_i\rangle) = \mathcal{L}(\langle x, y_i\rangle) \cup \mathcal{L}(\langle x, y_j\rangle)$
if y_i is and y_j is not distinguished, then 2. $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\uparrow(S, o) \mid S \in \mathcal{L}(\langle x, y_j\rangle)\}$ for some $o \in \mathcal{L}(y_i)$
and 3. remove y_j and all edges leading to y_j from the completion forest
$\mathbf{O}$-rule: if $o \in \mathcal{L}(x)$, x is neither blocked nor distinguished, and not $x \neq x_o$
then, for z distinguished with $o \in \mathcal{L}(z)$, do 1. $\mathcal{L}(z) = \mathcal{L}(z) \cup \mathcal{L}(x)$, and
2. if x has a predecessor x', then $\mathcal{L}(x') = \mathcal{L}(x') \cup \{\uparrow(R, o) \mid R \in \mathcal{L}(\langle x', x\rangle)\}$,
3. add $y \neq z$ for each y with $y \neq x$, and remove x and all edges leading to x from the completion forest

Figure 2: The complete tableaux expansion rules for $\mathcal{SHOQ}(\mathbf{D})$

be n S-successors $y'_1, \ldots, y'_n$ of x since neither the $\leqslant$-rule nor the $\mathbf{O}$-rule ever merges two nodes y'_i, y'_j with $y'_i \neq y'_j$, and, whenever the $\leqslant$- or the $\mathbf{O}$-rule removes a successor of x, there will be some S-successor z of x that "inherits" all inequalities from y'_i.

Hence the out-degree of the forest is bounded by nm.

(3) Nodes are labelled with subsets of $\mathsf{cl}(D) \cup \{\uparrow(R, o) \mid R \in \mathbf{R}^D_A$ and $o \in \mathbf{I}^D\}$, so there are at most $2^{m+k\ell}$ different node labellings. Therefore, if a path p is of length at least $2^{m+k\ell}$, then, from the blocking condition in Definition 4, there are two nodes x, y on p such that x is directly blocked by y. Hence paths are of length at most $2^{m+k\ell}$. ☐

Lemma 6 If a $\mathcal{SHOQ}(\mathbf{D})$ concept D in NNF has a tableau w.r.t. $\mathcal{R}$, then the expansion rules can be applied to D and $\mathcal{R}$ such that they yield a complete, clash-free completion forest.

Proof: Again, we concentrate on the new features nominals and datatypes and refer the reader to [Horrocks *et al.*, 1999] for the remainder. Given a tableau T for D w.r.t. $\mathcal{R}$, we can apply the non-deterministic rules, i.e., the $\sqcup$-, *choose*, and $\leqslant$-rule, in such a way that we obtain a complete and clash-free tableau: inductively with the generation of new nodes, we define a mapping π from nodes of the completion forest to individuals in the tableau and concrete values in such a way that $\mathcal{L}(x) \subseteq \mathcal{L}(\pi(x))$ for $\pi(x) \in \mathbf{S}$ and, for each pair of nodes x, y and each (abstract or concrete) role R, if y is an R-successor of x, then $\langle \pi(x), \pi(y)\rangle \in \mathcal{E}_A(R)$ or $\langle \pi(x), \pi(y)\rangle \in \mathcal{E}_{\mathbf{D}}(R)$. Please note that the latter also holds in the case that y is not a successor of x but a distinguished node (i.e., $\uparrow(R, o) \in \mathcal{L}(x)$ and $y = x_o$), and in the case that y is a concrete value (i.e., $\pi(y) \notin \mathbf{S}$). Due to (P12) and (P13), we do not encounter a clash of the form (3), and (P11) makes sure that the $\mathbf{O}$-rule can be applied correctly. ☐

Lemma 7 If the expansion rules can be applied to a $\mathcal{SHOQ}(\mathbf{D})$ 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. $\mathcal{R}$.

Proof: From a complete and clash-free completion forest F, we can obtain a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E}_A, \mathcal{E}_{\mathbf{D}})$ by *unravelling* as usual. That is, each element of the tableau is a *path* in the completion forest that starts at one of the root nodes and that, instead of going to a blocked node, goes to the node that is blocking this node (we disregard nodes that have datatypes in their labels). $\mathcal{E}$-successorship for abstract roles is defined according to the labels of edges (i.e., if $R' \in \mathcal{L}(\langle x_n, x_{n+1}\rangle)$ in F with $R' \,\underline{\circledast}\, R$, then $\langle x_0 \ldots x_n, x_0 \ldots x_n x_{n+1}\rangle \in \mathcal{E}_A(R)$ in T) and following labels $\uparrow(R, o)$ (i.e., if $\uparrow(R, o) \in \mathcal{L}(x_n)$ in F, then $\langle x_0 \ldots x_n, x_o\rangle \in \mathcal{E}_A(R)$). $\mathcal{E}$-successorship for concrete roles is defined following the edges to those (disregarded) nodes with datatypes in their labels. Clash-freeness makes sure that this is possible.

To satisfy (P8) also in cases where two R-successors y_1, y_2 of a node x with $(\geqslant nR.C)$ are blocked by the same node z, we must distinguish between individuals that, instead of going to y_i, go to z. This can be easily done as in [Horrocks *et* *al.*, 1999], annotating points in the path accordingly. Finally, we set $\mathcal{L}'(x_0 \dots x_n) = \mathcal{L}(x_n)$.

It remains to prove that T satisfies each (Pi). (P1) to (P10) are similar to those in [Horrocks *et al.*, 1999]. (P11) is due to completeness (otherwise, the **O**-rule was applicable), which implies that nominals can be found only in the labels of distinguished nodes (note that the definition of blocking is such that a distinguished node can never block another one). (P12) and (P13) are due to the fact that F has no clash of form (3), and that the $\exists$- and $\forall$-rule are not applicable. $\square$

As an immediate consequence of Lemmas 3, 5, 6, and 7, the completion algorithm always terminates, and answers with "D is satisfiable w.r.t. $\mathcal{R}$" iff D is satisfiable w.r.t. $\mathcal{R}$. Next, subsumption can be reduced to (un)satisfiability. Finally, as we mentioned in Section 2, $\mathcal{SHOQ}(\mathbf{D})$ can internalise general concept inclusion axioms, and we can thus decide these inference problems also w.r.t. terminologies.

Theorem 8 The completion algorithm presented in Definition 4 is a decision procedure for satisfiability and subsumption of $\mathcal{SHOQ}(\mathbf{D})$ concepts w.r.t. terminologies.

5 Conclusion

As we have seen, ontologies are set to play a key rôle in the Semantic Web, where they will provide a source of shared and precisely defined terms for use in descriptions of web resources. Moreover, such descriptions should be amenable to *automated reasoning* if they are to be used effectively by automated processes.

We have presented the DL $\mathcal{SHOQ}(\mathbf{D})$, along with a sound and complete decision procedure for concept satisfiability/subsumption. With its support for both nominals and concrete datatypes, $\mathcal{SHOQ}(\mathbf{D})$ is well suited to the provision of reasoning support for ontology languages in general, and web based ontology languages in particular. In addition, the $\mathcal{SHOQ}(\mathbf{D})$ decision procedure is similar to the $\mathcal{SHIQ}$ decision procedure implemented in the highly successful FaCT system, and should be amenable to a similar range of performance enhancing optimisations.

The only feature of languages such as OIL and DAML (and $\mathcal{SHIQ}$) that is missing in $\mathcal{SHOQ}(\mathbf{D})$ is inverse roles. Its exclusion was motivated by the very high complexity of reasoning that results from the unconstrained interaction of inverse roles with nominals and datatypes. Future work will include a detailed study of this interaction with a view to providing (restricted) support for inverse roles without triggering the explosion in complexity. An implementation (based on the FaCT system) is also planned, and will be used to test empirical performance.

References

[Areces *et al.*, 2000] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 2000. To appear.

[Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, pages 452–457, 1991.

[Baader and Sattler, 2000] F. Baader and U. Sattler. Tableau algorithms for description logics. In *Proc. TABLEAUX 2000*, vol. 1847 of *LNAI*, pages 1–18, 2000.

[Berners-Lee, 1999] T. Berners-Lee. *Weaving the Web*. Orion Business Books, 1999.

[Biron and Malhorta, 2000] Xml schema part 2: Datatypes. W3C Candidate Recommendation, Oct 2000. `http://www.w3.org/TR/xmlschema-2/`.

[Blackburn and Seligman, 1998] P. Blackburn and J. Seligman. What are hybrid languages? In *Advances in Modal Logic*, vol. 1, pages 41–62. CSLI Publications, 1998.

[Corcho and Pérez, 2000] O. Corcho and A. Gómez Pérez. Evaluating knowledge representation and reasoning capabilities of ontology specification languages. In *Proc. of ECAI-00 Workshop on Applications of Ontologies and Problem-Solving Methods*, 2000.

[De Giacomo, 1995] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Università degli Studi di Roma "La Sapienza", 1995.

[Decker *et al.*, 2000] S. Decker *et al.* The semantic web — on the respective roles of XML and RDF. *IEEE Internet Computing*, 2000.

[Fensel *et al.*, 2000] D. Fensel *et al.* OIL in a nutshell. In *Proc. of EKAW-2000*, LNAI, 2000.

[Hendler and McGuinness, 2001] J. Hendler and D. L. McGuinness. The DARPA agent markup language. *IEEE Intelligent Systems*, 2001.

[Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, vol. 1705 of *LNAI*, 1999.

[Horrocks, 2000] I. Horrocks. Benchmark analysis with FaCT. In *Proc. TABLEAUX 2000*, vol. 1847 of *LNAI*, 2000.

[Lutz, 2000] C. Lutz. Nexptime-complete description logics with concrete domains. In *Proceedings of the ESSLLI-2000 Student Session*, 2000.

[McGuinness, 1998] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS-98*. IOS-press, 1998.

[Schaerf, 1994] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[Streett, 1982] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Computation*, 54:121–141, 1982.

[Tobies, 2000] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *JAIR*, 12:199–217, 2000.

[Uschold and Grüninger, 1996] M. Uschold and M. Grüninger. Ontologies: Principles, methods and applications. *Knowledge Eng. Review*, 11(2), 1996.

[van Heijst *et al.*, 1997] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in KBS development. *Int. J. of Human-Computer Studies*, 46(2/3), 1997.