OWL Class Expression to SPARQL Rewriting

Lorenz Bühmann, Jens Lehmann (nicht sicher welche Reihenfolge)

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany,
{buehmann|lehmann}@informatik.uni-leipzig.de
http://aksw.org

Abstract. ... connection between OWL and SPARQL ...

1 Introduction

Motivation: OWL/DLs wurden über Jahrzehnte als Sprachen entwickelt um Konzepte einer Domäne abzubilden. Während OWL durchaus verbreitet ist, gibt es trotzdem viele Fälle in denen Wissensbasen direkt eher per SPARQL zugreifbar sind. Mit diesem Artikel soll ermöglicht werden OWL-Ausdrücke in SPARQL umzuschreiben, so dass dabei OWL und SPARQL Semantik berücksichtigt werden.

Verweis auf http://sparqr.dl-learner.org

Verweis auf ISWC Pattern Enrichment Paper

2 Syntax and Semantics of Description Logics

Aus PhD kopiert. Es sollte noch etwas gekürzt und geändert werden, aber auch nicht zu kurz, damit das Paper self-contained ist und die Reviewer/Leser auch verstehen was mit Interpretationen in Beschreibungslogiken gemeint ist.

In this section, we introduce description logics including their syntax and semantics.

Description logics is the name of a family of knowledge representation (KR) formalisms. They emerged from earlier KR formalisms like semantic networks and frames. Their origin lies in the work of Brachman on structured inheritance networks [**?**]. Since then, description logics have enjoyed increasing popularity. They can essentially be understood as fragments of first-order predicate logic. They have less expressive power, but usually decidable inference problems and a user-friendly variable free syntax.

Description logics represent knowledge in terms of *objects*, *concepts*, and *roles*. Concepts formally describe notions in an application domain, e.g. one could define the concept of being a father as "a man having a child" ($\texttt{Father} \equiv \texttt{Man} \sqcap \exists \texttt{hasChild}.\top$ in DL notation). Objects are members of concepts in the application domain and roles are binary relations between objects. Objects correspond to constants, concepts to unary predicates, and roles to binary predicates in first-order logic.

In description logic systems information is stored in a *knowledge base*. It is sometimes divided in two parts: *TBox* and *ABox*. The ABox contains *assertions* about objects. It relates objects to concepts and other objects via roles. The TBox describes the *terminology* by relating concepts and roles. For some expressive description logics this clear separation does not exist. Furthermore, the notion of an *RBox*, which contains knowledge about roles, is sometimes used in expressive description logics. We will usually consider those axioms as part of the TBox in this thesis.

As mentioned before, DLs are a family of KR formalisms. We use the terms *description language* and *description logic* synonymously for one particular element of this family. First, we introduce the $\mathcal{ALC}$ description logic as an example language. $\mathcal{ALC}$ is a proper fragment of OWL [**?**] and is generally considered to be a prototypical description logic for research investigations. $\mathcal{ALC}$ stands for *attributive language with complement*. It allows to construct complex concepts from simpler ones using various language constructs. The next definition shows how such concepts can be built.

Definition 1 (Syntax of $\mathcal{ALC}$ concepts). *Let N_R be a set of* role names *and N_C be a set of* concept names *($N_R \cap N_C = \emptyset$). The elements of N_C are also called* atomic concepts. *The set of $\mathcal{ALC}$ concepts is inductively defined as follows:*

1. *Each atomic concept is an $\mathcal{ALC}$ concept.*
2. *If C and D are $\mathcal{ALC}$ concepts and $r \in N_R$ a role, then the following are also $\mathcal{ALC}$ concepts:*
   - *$\top$ (top), $\bot$ (bottom)*
   - *$C \sqcup D$ (disjunction), $C \sqcap D$ (conjunction), $\neg C$ (negation)*
   - *$\forall r.C$ (value/universal restriction), $\exists r.C$ (existential restriction)*

Example 1 ($\mathcal{ALC}$ concepts). Some examples of complex concepts in $\mathcal{ALC}$ are:

- $\texttt{Man} \sqcap \exists \texttt{hasChild}.\top$
- $\texttt{Man} \sqcap \exists \texttt{hasChild}.(\texttt{Rich} \sqcup \texttt{Beautiful})$
- $\texttt{Man} \sqcap \exists \texttt{hasChild}.\neg\texttt{Adult}$
- $\texttt{Man} \sqcap \exists \texttt{hasChild}.\forall \texttt{hasFriend}.\texttt{ComputerScientist}$

Other description languages are usually named according to the expressive features they support. The choice of language is usually a tradeoff between expressivity and complexity of reasoning. The description logic navigator[1] provides detailed information about the complexity of a particular language. The following is a list of commonly used letters in the description logic naming scheme along with their meaning (note that if one feature can be expressed using other ones the letter is usually omitted in the language name).

$\boxed{\mathcal{S}}$ $\mathcal{ALC}$ + transitivity: For a transitive role r, we have that $r(a, b)$ and $r(b, c)$ implies $r(a, c)$.

[1] `http://www.cs.manchester.ac.uk/~ezolin/dl/`

$\boxed{\mathcal{H}}$ subroles: $r \sqsubseteq s$ says that r is a subrole of s, i.e. $r(a, b)$ implies $s(a, b)$.

$\boxed{\mathcal{I}}$ inverse roles: r^- denotes the inverse role of r, i.e. $r^-1(a, b)$ iff $r(b, a)$.

$\boxed{\mathcal{O}}$ nominals: Sets of objects can be used to construct concepts, e.g. $\{\texttt{MONICA}\}$ denotes the singleton set, which only contains `MONICA`. Nominals are useful in cases where the instances of a concept should be enumerated, e.g. the members of the European Union.

$\boxed{\mathcal{N}}$ number restrictions: Allows constructs of the form $\geq n\ r$ and $\leq n\ r$ to build concepts. This is useful if one wants to define a concept like "mother of at least three children" ($\texttt{Woman} \sqcap \geq 3\ \texttt{hasChild}$).

$\boxed{\mathcal{Q}}$ qualified number restrictions: Concept constructors of the form $\geq n\ r.C$ and $\leq n\ r.C$ can be used. If C is the top concept, this is equivalent to unqualified number restrictions. This is useful to define a concept like "mother of at least three male children" ($\texttt{Woman} \sqcap \geq 3\ \texttt{hasChild}.\texttt{Male}$).

$\boxed{\mathcal{F}}$ functional roles: Allows to express that a role r is functional, i.e. has at most one filler, which is equivalent to the axiom $\top \sqsubseteq \leq 1\ r$.

$\boxed{\mathcal{R}}$ complex role inclusions: Axioms of the form $r \circ s \sqsubseteq r$ (or $r \circ s \sqsubseteq s$) state that when $r(a, b)$ and $s(b, c)$ holds, then $r(a, c)$ (or $s(a, c)$) also holds. For instance, we could use the axiom $\texttt{locatedIn} \circ \texttt{partof} \sqsubseteq \texttt{locatedIn}$ to model the part of relationship for locations. Now, if we know that Leipzig is located in Saxony and Saxony is part of Germany, we can infer that Leipzig is located in Germany.

$\boxed{\text{D}}$ data types: Data types are used to incorporate different kinds of data, e.g. numbers or strings. This allows, for instance, to define the concept of an old person as a person of age 65 or higher.

While $\mathcal{ALC}$ is seen as a prototypical language and foundation for more expressive languages, there has also been a lot of research effort for simple languages with often tractable inference problems. Two of those languages, which are referred to within the thesis are $\mathcal{AL}$ and $\mathcal{EL}$:

$\mathcal{AL}$ is inductively defined as follows: $\top$, $\bot$, $\exists r.\top$, A, $\neg A$ with $A \in N_C$, $r \in N_R$ are $\mathcal{AL}$ concepts. If C and D are $\mathcal{AL}$ concepts, then $C \sqcap D$ is an $\mathcal{AL}$ concept. If C is an $\mathcal{AL}$ concept and r a role, then $\forall r.C$ is an $\mathcal{AL}$ concept.

$\mathcal{EL}$ is inductively defined as follows: $\top$, A with $A \in N_C$ are $\mathcal{EL}$ concepts. If C and D are $\mathcal{EL}$ concepts and $r \in N_R$, then $C \sqcap D$ and $\exists r.C$ are $\mathcal{EL}$ concepts.

The semantics of concepts is defined by means of interpretations. See the following definition and Table 1 listing common concept constructors.

Definition 2 (Interpretation). *An* interpretation $\mathcal{I}$ *consists of a non-empty* interpretation domain $\Delta^{\mathcal{I}}$ *and an* interpretation function $\cdot^{\mathcal{I}}$*, which assigns to each* $A \in N_C$ *a set* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ *and to each* $r \in N_R$ *a binary relation* $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Example 2 (Interpreting Concepts). Let the interpretation $\mathcal{I}$ be given by:

$$\Delta^{\mathcal{I}} = \{\texttt{MONICA}, \texttt{JESSICA}, \texttt{STEPHEN}\}$$
$$\texttt{Woman}^{\mathcal{I}} = \{\texttt{MONICA}, \texttt{JESSICA}\}$$
$$\texttt{hasChild}^{\mathcal{I}} = \{(\texttt{MONICA}, \texttt{STEPHEN}), (\texttt{STEPHEN}, \texttt{JESSICA})\}$$

construct	syntax	semantics
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
nominals	$\{o\}$	$\{o\}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}, \lvert\{o\}\rvert^{\mathcal{I}} = 1$
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
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	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
exists restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{a \mid \exists b.(a,b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$(\forall r.C)^{\mathcal{I}} = \{a \mid \forall b.(a,b) \in r^{\mathcal{I}} \text{ implies } b \in C^{\mathcal{I}}\}$
atleast restriction	$\geq n\ r.C$	$(\geq n\ r)^{\mathcal{I}} = \{a \mid \lvert\{b \mid (a,b) \in r^{\mathcal{I}}\}\rvert \geq n\}$
atmost restriction	$\leq n\ r.C$	$(\leq n\ r)^{\mathcal{I}} = \{a \mid \lvert\{b \mid (a,b) \in r^{\mathcal{I}}\}\rvert \leq n\}$

Table 1: Syntax and semantics for concepts in $\mathcal{SHOIN}$.

We then have:

$$(\texttt{Woman} \sqcap \exists \texttt{hasChild}.\top)^{\mathcal{I}} = \{\texttt{MONICA}\}$$

In the most general case, *terminological axioms* are of the form $C \sqsubseteq D$ or $C \equiv D$, where C and D are (complex) concepts. The former axioms are called *inclusions* and the latter *equivalences*. An equivalence whose left hand side is an atomic concept is a *concept definition*. In some languages with low expressivity, like $\mathcal{AL}$, terminological axioms are restricted to definitions. We can define the semantics of terminological axioms in a straightforward way. An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and it satisfies the equivalence $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ satisfies a set of terminological axioms iff it satisfies all axioms in the set. An interpretation, which satisfies a (set of) terminological axiom(s) is called a *model* of this (set of) axiom(s). Two (sets of) axioms are *equivalent* if they have the same models. A finite set $\mathcal{T}$ of terminological axioms is called a *(general) TBox*. Let N_I be the set of object names (disjoint with N_R and N_C). An *assertion* has the form $C(a)$ (*concept assertion*), $r(a,b)$ (*role assertion*), where a, b are object names, C is a concept, and r is a role. An *ABox* $\mathcal{A}$ is a finite set of assertions.

Objects are also called individuals. To allow interpreting ABoxes we extend the definition of an interpretation. In addition to mapping concepts to subsets of our domain and roles to binary relations, an interpretation has to assign to each individual name $a \in N_I$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all $C(a) \in \mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all $r(a,b) \in \mathcal{A}$. An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (written $\mathcal{I} \models \mathcal{K}$) iff it is a model of $\mathcal{T}$ and $\mathcal{A}$.

Example 3 (Models of a Knowledge Base). Let the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be given by:

TBox $\mathcal{T}$:

$$\begin{aligned} \texttt{Man} &\equiv \neg\texttt{Woman} \sqcap \texttt{Person} \\ \texttt{Woman} &\sqsubseteq \texttt{Person} \\ \texttt{Mother} &\equiv \texttt{Woman} \sqcap \exists \texttt{hasChild}.\top \end{aligned}$$

ABox $\mathcal{A}$:

$$\begin{aligned} &\texttt{Man}(\texttt{STEPHEN}). \\ &\neg\texttt{Man}(\texttt{MONICA}). \\ &\texttt{Woman}(\texttt{JESSICA}). \\ &\texttt{hasChild}(\texttt{STEPHEN}, \texttt{JESSICA}). \end{aligned}$$

We will now look at some interpretations and determine whether or not they are a model of $\mathcal{K}$. For all interpretations, the domain $\{\texttt{MONICA}, \texttt{JESSICA}, \texttt{STEPHEN}\}$ is used and all object names are interpreted in the obvious way (STEPHEN is interpreted as STEPHEN etc.).

Let the interpretation $\mathcal{I}_1$ be given by:

$$\begin{aligned} \texttt{Man}^{\mathcal{I}_1} &= \{\texttt{JESSICA}, \texttt{STEPHEN}\} \\ \texttt{Woman}^{\mathcal{I}_1} &= \{\texttt{MONICA}, \texttt{JESSICA}\} \\ \texttt{Mother}^{\mathcal{I}_1} &= \emptyset \\ \texttt{Person}^{\mathcal{I}_1} &= \{\texttt{JESSICA}, \texttt{MONICA}, \texttt{STEPHEN}\} \\ \texttt{hasChild}^{\mathcal{I}_1} &= \{(\texttt{STEPHEN}, \texttt{JESSICA})\} \end{aligned}$$

Clearly this does not satisfy $\mathcal{T}$, because the definition $\texttt{Man} \equiv \neg\texttt{Woman} \sqcap \texttt{Person}$ is not satisfied. We have $\texttt{Man}^{\mathcal{I}_1} = \{\texttt{JESSICA}, \texttt{STEPHEN}\}$ and $(\neg\texttt{Woman} \sqcap \texttt{Person})^{\mathcal{I}_1} = \{\texttt{STEPHEN}\}$, which are not equal. However, $\mathcal{I}_1$ satisfies $\mathcal{A}$.

Let the interpretation $\mathcal{I}_2$ be given by:

$$\begin{aligned} \texttt{Man}^{\mathcal{I}_2} &= \{\texttt{STEPHEN}\} \\ \texttt{Woman}^{\mathcal{I}_2} &= \{\texttt{JESSICA}, \texttt{MONICA}\} \\ \texttt{Mother}^{\mathcal{I}_2} &= \emptyset \\ \texttt{Person}^{\mathcal{I}_2} &= \{\texttt{JESSICA}, \texttt{MONICA}, \texttt{STEPHEN}\} \\ \texttt{hasChild}^{\mathcal{I}_2} &= \emptyset \end{aligned}$$

$\mathcal{I}_2$ satisfies $\mathcal{T}$, but not $\mathcal{A}$. We have $\texttt{hasChild}(\texttt{STEPHEN}, \texttt{JESSICA}) \in \mathcal{A}$, but $(\texttt{STEPHEN}^{\mathcal{I}_2}, \texttt{JESSICA}^{\mathcal{I}_2}) \notin \texttt{hasChild}^{\mathcal{I}_2}$.

Let the interpretation $\mathcal{I}_3$ be given by:

$$\begin{aligned} \texttt{Man}^{\mathcal{I}_3} &= \{\texttt{STEPHEN}\} \\ \texttt{Woman}^{\mathcal{I}_3} &= \{\texttt{JESSICA}, \texttt{MONICA}\} \\ \texttt{Mother}^{\mathcal{I}_3} &= \{\texttt{MONICA}\} \\ \texttt{Person}^{\mathcal{I}_3} &= \{\texttt{JESSICA}, \texttt{MONICA}, \texttt{STEPHEN}\} \\ \texttt{hasChild}^{\mathcal{I}_3} &= \{(\texttt{MONICA}, \texttt{STEPHEN}), (\texttt{STEPHEN}, \texttt{JESSICA})\} \end{aligned}$$

$\mathcal{I}_3$ is a model of $\mathcal{T}$ and $\mathcal{A}$, so it is a model of $\mathcal{K}$. One may argue that nothing in our knowledge base justifies the fact that we interpret `MONICA` as mother. However, in DLs we usually have the *open world assumption*. This means that the given knowledge is viewed as incomplete. There is nothing, which tells us that `MONICA` is not a mother. In databases one usually uses the *closed world assumption*, i.e. all facts, which are not explicitly stored, are assumed to be false.

As we have described, a knowledge base can be used to represent the information we have about an application domain. Besides this *explicit* knowledge, we can also deduce *implicit* knowledge from a knowledge base. It is the aim of *inference algorithms* to extract such implicit knowledge. There are some standard reasoning tasks in description logics, which we will briefly describe.

In *terminological reasoning* we reason about concepts. The standard problems are *consistency*, *satisfiability* and *subsumption*. Intuitively, consistency checks detect whether a knowledge base contains contradictions. Satisfiability determines whether a concept can be satisfied, i.e. it is free of contradictions. Subsumption of two concepts detects whether one of the concepts is more general than the other.

Definition 3 (Consistency). *A knowledge base $\mathcal{K}$ is* consistent *iff it has a model.*

Example 4 (Consistency). The knowledge base $\mathcal{K} = \{A_1 \equiv A_2 \sqcap \neg A_2, A_1(a)\}$ is not consistent, since A_1 is equivalent to $\bot$ and has an asserted instance a.

Definition 4 (Satisfiability). *Let C be a concept and $\mathcal{T}$ a TBox. C is* satisfiable *iff there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. C is* satisfiable with respect to $\mathcal{T}$ *iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.*

Example 5 (Satisfiability). `Man`$\sqcap$`Woman` is satisfiable. However, it is not satisfiable with respect to the TBox in Example 3.

Definition 5 (Subsumption, Equivalence). *Let C, D be concepts and $\mathcal{T}$ a TBox. C* is subsumed by *D, denoted by $C \sqsubseteq D$, iff for any model $\mathcal{I}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. C* is subsumed by *D* with respect to *$\mathcal{T}$, denoted by $C \sqsubseteq_{\mathcal{T}} D$, iff for any model $\mathcal{I}$ of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.*

C is equivalent to *D* (with respect to *$\mathcal{T}$*), *denoted by $C \equiv D$ ($C \equiv_{\mathcal{T}} D$), iff $C \sqsubseteq D$ ($C \sqsubseteq_{\mathcal{T}} D$) and $D \sqsubseteq C$ ($D \sqsubseteq_{\mathcal{T}} C$).*

C is strictly subsumed by *D* (with respect to *$\mathcal{T}$*), *denoted by $C \sqsubset D$ ($C \sqsubset_{\mathcal{T}} D$), iff $C \sqsubseteq D$ ($C \sqsubseteq_{\mathcal{T}} D$) and not $C \equiv D$ ($C \equiv_{\mathcal{T}} D$).*

Example 6 (Subsumption). `Mother` is not subsumed by `Woman`. However, `Mother` is subsumed by `Woman` with respect to the TBox in Example 3.

Subsumption allows to build a hierarchy of atomic concepts, commonly called the *subsumption hierarchy*. Analogously, for more expressive description logics *role hierarchies* can be inferred.

In *assertional reasoning* one reasons about objects. As one relevant task for learning in DLs, the *instance check problem* is to find out whether an object is an instance of a concept, i.e. belongs to it. A *retrieval* operation finds all instances of a given concept.

Definition 6 (Instance Check). *Let $\mathcal{A}$ be an ABox, $\mathcal{T}$ a TBox, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a knowledge base, C a concept, and $a \in N_I$ an object.* a is an instance of C with respect to $\mathcal{A}$, *denoted by $\mathcal{A} \models C(a)$, iff in any model $\mathcal{I}$ of $\mathcal{A}$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$.* a is an instance of C with respect to $\mathcal{K}$, *denoted by $\mathcal{K} \models C(a)$, iff in any model $\mathcal{I}$ of $\mathcal{K}$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$.*

To denote that a is not an instance of C with respect to $\mathcal{A}$ ($\mathcal{K}$) we write $\mathcal{A} \not\models C(a)$ ($\mathcal{K} \not\models C(a)$).

We use the same notation for sets S of assertions of the form $C(a)$, e.g. $\mathcal{K} \models S$ means that every element in S follows from $\mathcal{K}$.

Definition 7 (Retrieval). *Let $\mathcal{A}$ be an ABox, $\mathcal{T}$ a TBox, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a knowledge base, C a concept. The* retrieval $R_{\mathcal{A}}(C)$ of a concept C with respect to $\mathcal{A}$ *is the set of all instances of C: $R_{\mathcal{A}}(C) = \{a \mid a \in N_I \text{ and } \mathcal{A} \models C(a)\}$. Similarly the* retrieval $R_{\mathcal{A}}(C)$ of a concept C with respect to $\mathcal{K}$ *is $R_{\mathcal{K}}(C) = \{a \mid a \in N_I \text{ and } \mathcal{K} \models C(a)\}$.*

Example 7 (Instance Check, Retrieval). In Example 3 we have $R_{\mathcal{K}}(\texttt{Woman}) = \{\texttt{JESSICA}, \texttt{MONICA}\}$. `JESSICA` and `MONICA` are instances of `Woman`, because in any model $\mathcal{I}$ of $\mathcal{K}$ we have $\texttt{JESSICA}^{\mathcal{I}} \in \texttt{Woman}^{\mathcal{I}}$ and $\texttt{MONICA}^{\mathcal{I}} \in \texttt{Woman}^{\mathcal{I}}$.

We introduce some further notions, which are used in the thesis. A concept is in *negation normal form* iff negation only occurs in front of concept names. The *length* of a concept is defined in a straightforward way, namely as the sum of the numbers of concept names, role names, quantifier, and connective symbols occurring in the concept. In particular, for $\mathcal{ALC}$ we have:

Definition 8 (Length of an $\mathcal{ALC}$ Concept). *The* length $|C|$ of a concept C *is defined inductively (A stands for an atomic concept):*

$$\begin{aligned} |A| = |\top| = |\bot| &= 1 \\ |\neg D| &= |D| + 1 \\ |D \sqcap E| = |D \sqcup E| &= 1 + |D| + |E| \\ |\exists r.D| = |\forall r.D| &= 2 + |D| \end{aligned}$$

The *depth* of a concept is the maximal number of nested concept constructors. The *role depth* of a concept is the maximal number of nested roles. A *subconcept* of a concept C is a concept syntactically contained in C. For brevity we sometimes omit brackets. In this case, constructors involving quantifiers have higher priority, e.g. $\exists r.\top \sqcap A$ means $(\exists r.\top) \sqcap A$. In several proofs in the thesis we use a convenient abbreviated notation to denote $\forall r$ *chains* and $\exists r$ *chains*:

$$\forall r^n = \underbrace{\forall r. \ldots . \forall r}_{n-times} \qquad \exists r^n = \underbrace{\exists r. \ldots . \exists r}_{n-times}$$

For more detailed information about description logics, we refer the interested reader to [**?**,**?**,**?**] .

3 SPARQL Semantics

To show the connection to SPARQL, we first introduce basic notions. For a complete definition of the SPARQL syntax and semantics, however, we refer to [**?**], [**?**] and the official W3C recommendation[2]. We use the SPARQL algebra syntax and semantics in [**?**] in this section and repeat the essential notions below.

We use V to denote SPARQL variables. A mapping μ from V to a set U is a partial function $\mu : V \to U$. The domain of μ, denoted by $dom(\mu)$, is the subset of V where μ is defined. $var(P)$ is the set of variables contained in a basic graph pattern P. Given a triple pattern t and a mapping μ with $var(t) \subseteq dom(\mu)$, $\mu(t)$ is the triple pattern obtained by replacing the variables in t according to μ. This can be extended to a basic graph pattern P by defining $\mu(P) = \cup_{t \in P}\{\mu(t)\}$.

The evaluation of a basic graph pattern P, denoted by $[[P]]$, in $\mathcal{A}$ is then defined as the set of mappings $[[P]]^{\mathcal{A}} = \{\mu : V \to U \mid (dom(\mu) = var(P) \text{ and } \mu(P) \subseteq \mathcal{A}\}$. This can be extended to graph patterns and filters. Note that we will sometimes drop $\mathcal{A}$ if it is clear from the context. In the following, we will only introduce the notions required in our proof. For a conjunction of several triple patterns, their semantics are defined as $[[(P_1 \text{ AND } \ldots \text{ AND } P_n)]] = [[P_1]] \bowtie \cdots \bowtie [[P_n]]$. The join operator for two sets of mappings Ω_1 and Ω_2 is defined as $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ and } \mu_1, \mu_2 \text{ are compatible mappings }\}$. Two mappings μ_1 and μ_2 are *compatible* if for all $x \in dom(\mu_1) \cap dom(\mu_2)$, it is the case that $\mu_1(x) = \mu_2(x)$. The semantics of UNION patterns are defined as $[[P_1 \text{ UNION } \ldots \text{ UNION } P_n]] = [[P_1]] \cup \cdots \cup [[P_n]]$. For filters, we need to introduce the notion of *satisfaction*. Given a mapping μ and a built-in condition R, the satisfaction of R by μ is denoted by $\mu \models R$ in a three valued logic (`true`, `false`, `error`). In our case, built-in conditions of the form `?x = ?y` are relevant, which result in an error if $?x \notin dom(\mu)$ or $y \notin dom(\mu)$. They are `true` if $\mu(?x) = \mu(?y)$ and `false` otherwise. The built-in construct of the form `?v IN S` where S is a set of expressions, is satisfied if $\mu(?v) \in S$ and `false` otherwise. NOT EXISTS constructs in filters take a graph pattern P as input and evaluate it. They result in `false` if $\mu \in [[P]]$ and `true` otherwise. The semantics of a filter in $\mathcal{A}$ is defined as $[[P \text{ FILTER } R]]^{\mathcal{A}} = \{\mu \in [[P]]^{\mathcal{A}} \mid \mu \models R\}$.

A SELECT query restricts the evaluation of a basic graph pattern to a set of variables W. The evaluation of a SELECT query (W, P) over a dataset $\mathcal{A}$ is the set of mappings $[[(W, P)]]^{\mathcal{A}} = \{\mu_{|W} \mid \mu \in [[P]]^{\mathcal{A}}\}$. Since the interpretation of DL a concept results in a subset of the domain of the interpretation, we need to view such an evaluation as a set. Assuming a set Ω of mappings is given, this can be achieved by performing a selection over a single variable v and collecting the results in a set, i.e. $\Omega(v) = \{\mu(v) \mid \mu \in \Omega\}$.

[2] `http://www.w3.org/TR/sparql11-query/`

4 OWL Class Expression Rewriting Algorithm

We restrict the description of the pattern rewriting algorithm for brevity to SubClassOf axiom patterns, as it was the most frequently used schema axiom type we discovered during the evaluation, and secondly each TBox axiom can be rewritten as a set of SubClassOf axioms.

Let $C, D, C_1, \ldots, C_n$ be class expressions, A be an atomic class, r be an object property, $a_1, \ldots, a_n$ be individuals and $\Theta = \{\leq, \geq, =\}$. Let $\tau(C, v)$ be a mapping from a class expression C and a target variable v to a SPARQL graph pattern $\mathfrak{p}$ as described in Table 2. A given axiom pattern $C \sqsubseteq D$ can be converted into a SPARQL query pattern $\mathfrak{p}$ by applying $\mathfrak{p} := \tau(C \sqcap D)$. We assume that τ generates unused query variables in its recursion.

The purpose of the SPARQL query is to determine whether instance data is structured according to the axiom pattern. Naturally, the SPARQL queries are only an approximation and do not employ the OWL open world assumption or use reasoning unless provided by the SPARQL endpoint[3]. The goal is not to infer axioms, but rather to detect statistical evidence for axioms in the ABox.

Kann man das noch für weitere Konstrukte ausbauen um a) den Kernteil des Papers umfangreicher zu machen und es b) vom ISWC-Paper etwas abzugrenzen.

Es ist noch nicht ganz klar, ob wir nicht nur den Beweis, sondern ev. auch Teile des Algorithmus zum Umschreiben in diesen externen Artikel/Report auslagern sollten.

5 Formal Relationship between a Class Expression and its SPARQL Rewriting

In this section, we establish a formal connection between description logic concepts and the SPARQL rewriting algorithm presented in the paper. As a prerequiste, we assume that the reader is familiar with description logics and refer to [**?**] for details. Essentially, we show that the evaluation of the interpretation function in description logics corresponds to the evaluation of the SPARQL algebra expression of the generated query according to SPARQL semantics. Another way to view this result is that a retrieval for the description logic concept and an execution of the rewritten query return essentially the same results[4] when considering only facts as background knowledge (no inference), employing the unique names assumption and a closed world assumption.

In the following, we assume that an ABox $\mathcal{A}$ is given, which contains statements of the form $A(a)$, i.e. class assertions to named classes, and $r(a, b)$, i.e. role assertions. In RDF, those facts correspond to triples `a rdf:type C` and `a r b` respectively. We denote the objects, classes and roles in the ABox with N_O, N_C

[3] See `http://www.w3.org/TR/sparql11-entailment/`.

[4] In DL, a set of resources is returned, whereas in SPARQL it is a set of mappings from a variable ?var to a resource, i.e. a table with a single column. By equal, we mean that the returned resources are the same.

Class Expression C_i	Graph Pattern $\mathfrak{p} = \tau(C_i, \texttt{?var})$
A	{?var a A.}
$\neg C$	{?var ?p ?o . FILTER NOT EXISTS {$\tau(C,$?var)}}
$\{a_1, \ldots, a_n\}$	{?var ?p ?o . FILTER (?var IN ($a_1, \ldots, a_n$))}
$C_1 \sqcap \ldots \sqcap C_n$	{$\tau(C_1,$?var) $\cup \ldots \cup$ $\tau(C_n,$?var)}
$C_1 \sqcup \ldots \sqcup C_n$	{$\tau(C_1,$?var)} UNION ... UNION {$\tau(C_n,$?var)}
$\exists\ r.C$	{?var r ?s.} $\cup$ $\tau(C,$?s)
$\exists\ r.\{a\}$	{?var r a.}
$\forall\ r.C$	{ ?var r ?s0. { SELECT ?var (count(?s1) AS ?cnt1) WHERE { ?var r ?s1 . $\tau(C, ?s1)$ } GROUP BY ?var } { SELECT ?var (count(?s2) AS ?cnt2) WHERE { ?var r ?s2 } GROUP BY ?var } FILTER (?cnt1 = ?cnt2) }
$\Theta n\ r.C$	{ ?var r ?s0. { SELECT ?var WHERE { ?var r ?s . $\tau(C, ?s)$ } GROUP BY ?var HAVING (count(?s) Θ n) } }

Table 2: Conversion of class expressions into a SPARQL graph pattern.

and N_R, respectively. Using this, we can define an interpretation $\mathcal{I}$ as:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= N_O \\
a^{\mathcal{I}} &= a \\
A^{\mathcal{I}} &= \{a \mid A(a) \in \mathcal{A}\} \quad \text{for all } A \in N_C \\
r^{\mathcal{I}} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\} \quad \text{for all } r \in N_R
\end{aligned}$$

This interpretation is called the *canonical interpretation* of $\mathcal{A}$ [**?**]. In the following, let τ be the function defined in Table 2. For simplicity, we assume that τ returns a SPARQL query algebra expression (in contrast to a query string). We can then show that executing the query converted from a concept C using τ over $\mathcal{A}$ is the same as the canoncial interpretation of C.

Proposition 1. *Let $\mathcal{A}$ be an ABox, C a DL concept and $\mathcal{I}$ the canoncial interpretation of $\mathcal{A}$. Then $C^{\mathcal{I}} = [[\tau(C, ?var)]]^{\mathcal{A}}(?var)$.*

Proof. We prove by induction over the structure of class expressions. In each case, we show that the DL and SPARQL semantics are equal with respect to $\cdot^{\mathcal{I}}$ and τ. For the cases $\forall r.C$ and $\Theta n\ r.C$, the conversion between both semantics is only sketched (due to length).
Induction Base: $A^{\mathcal{I}}$ contains the explicit instances of A by the definition of canonical interpretation and $[[\tau((?var, rdf : type, A))]](?var)$ contains the subjects of triples asserting instances to A. Hence, $A^{\mathcal{I}} = [[\tau(A, ?var)]](?var)$.
Induction Step:

$C = C_1 \sqcap \cdots \sqcap C_n$: We have $C^{\mathcal{I}} = C_1^{\mathcal{I}} \cap \cdots \cap C_n^{\mathcal{I}}$, i.e. $C^{\mathcal{I}}$ is the intersection of the interpretation of all C_i. By SPARQL semantics, we also have $[[\tau(C, ?var)]] = [[C_1]] \bowtie \cdots \bowtie [[C_n]]$. For all i with $1 \leq i \leq n$, $C_i^{\mathcal{I}} = [[\tau(C_i, ?var)]](?var)$ follows by induction. Since all $\tau(C_i, ?var)$ just contain a single common variable, the compatibility check in the join operation ensures that ?var is mapped to the same entity for the evaluation of each subpattern $[[\tau(C_i, ?var)]]$. Thus, the result only contains elements which are in each set $C_i^{\mathcal{I}}$ and, therefore, $(C_1 \sqcap \cdots \sqcap C_n)^{\mathcal{I}} = [[\tau(C_1 \sqcap \cdots \sqcap C_n, ?var)]](?var)$.

$C = C_1 \sqcup \cdots \sqcup C_n$: We have $C^{\mathcal{I}} = C_1^{\mathcal{I}} \cup \cdots \cup C_n^{\mathcal{I}}$, i.e. $C^{\mathcal{I}}$ is the union of the interpretation of all C_i. For the SPARQL rewrite, we have $[[\tau(C, ?var)]] = [[\tau(C_1 \text{ UNION } \ldots \text{ UNION } C_n, ?var)]] = [[\tau(C_1, ?var)]] \cup \cdots \cup [[\tau(C_n, ?var)]]$, which is trivially equal to $C^{\mathcal{I}}$ by induction.

$C = \neg C'$: We have $C^{\mathcal{I}} = \Delta \setminus C'^{\mathcal{I}}$ by DL semantics. For SPARQL, we have $[[\tau(C, ?var)]] = \{\mu \in [[(?var, ?p, ?o)]] \mid \mu \notin [[\tau(C', ?var)]]\}$ by filter semantics. In this formula, $[[(?var, ?p, ?o)]]$ contains mappings from `?var` to all elements of the domain Δ, whereas $[[\tau(C', ?var)]](?var) = C'^{\mathcal{I}}$ by induction. Thus, $[[\tau(C, ?var)]] = \Delta \setminus C'^{\mathcal{I}} = C^{\mathcal{I}}$.

$C = \exists r.C'$: $C^{\mathcal{I}} = \{a \mid \exists b\, \langle a, b\rangle \in r^{\mathcal{I}} \wedge b \in C'^{\mathcal{I}}\}$ according to DL semantics. This can also be written as a join of a binary relation r and a unary relation C on the second argument of r (and a projection to the first argument of r), i.e. $C^{\mathcal{I}} = \pi_1(r^{\mathcal{I}} \bowtie_{2=1} C'^{\mathcal{I}})$. For SPARQL, we can evaluate $[[\tau(C, ?var)]] = [[(?var, r, ?s)]] \bowtie [[\tau(C', ?s)]]$, i.e. we have the same structure. $[[(?var, r, ?s)]]$ corresponds to $r^{\mathcal{I}}$ and $[[\tau(C', ?s)]]$ corresponds to $C'^{\mathcal{I}}$ by induction. Thus, we get $[[\tau(C, ?var)]] = [[\tau(C, ?var)]](?var)$. We do not treat $\exists r.\{a\}$ in the definiton of τ separately as it only provides a shortcut to a longer equivalent query.

$C = \{a_1, \ldots, a_n\}$: By DL semantics, we have $C^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$. For SPARQL, we have $[[\tau(C, ?var)]] = \{\mu \in [[(?var, ?p, ?o)]] \mid \mu(?var) \in C^{\mathcal{I}}\}$ by filter semantics. Similar to negation, $[[(?var, ?p, ?o)]]$ contains mappings from `?var` to all elements of the domain Δ (and is only used as a dummy since filters cannot

stand alone), whereas $\mu(?var) \in C^{\mathcal{I}}$ restricts those exactly to the elements of $C^{\mathcal{I}}$.

$C = \forall r.C'$ (sketch): By DL semantics, we have $(\forall r.C)^{\mathcal{I}} = \{a \mid \forall b : \langle a, b\rangle \in r^{\mathcal{I}}$ implies $b \in C'^{\mathcal{I}}\}$, i.e. for all fillers of r, we need to check whether they are in C'. In $\tau(C, ?var)$, we achieve this by first counting the number of all fillers of r in a sub-select (`?cnt1`) and then counting the number of all fillers of r belonging to C' (`?cnt2`). This is done for each subject of r (`GROUP BY ?var`). All subjects, for which both counts are equal, are part of $[[\tau(C, ?var)]]$, i.e. the SPARQL query corresponds exactly to DL semantics.

$C = \Theta n\ r.C$ (sketch): By DL semantics, we have $(\Theta n\ r)^{\mathcal{I}} = \{a \mid |(\{b \mid (a, b) \in r^{\mathcal{I}}\}| \ \Theta\ n\}$. Again, the SPARQL query is corresponding exactly to DL semantics: In the SPARL query, we count the fillers of r (`count(?s)`) grouped by subject (`GROUP BY ?var`). The HAVING expression is then used to verify whether the cardinality restriction is satisfied.

6 Related Work

7 Conclusions and Future Work

References

1. M. Arenas, C. Gutierrez, and J. Pérez. On the semantics of sparql. In *Semantic Web Information Management*, pages 281–307. Springer, 2010.
2. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
3. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2007.
4. R. J. Brachman. A structural paradigm for representing knowledge. Technical Report BBN Report 3605, Bolt, Beraneck and Newman, Inc., Cambridge, MA, 1978.
5. P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. CRC Press/Chapman & Hall, 2009.
6. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 57–67. AAAI Press, 2006.
7. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
8. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):16, 2009.