Chapter 9: Description Logics and OWL

In fact, in terms of syntax, OWL Just tends to be a bulky fowl, However, if it mates with Turtle This union turns out rather fertile; I deem the offspring of this love As graceful as a turtledove.

Introduction

- OWL is based on Description Logics with additional features
  - e.g., ontology versioning information and annotations.
- OWL supports modeling and reasoning with datatypes

- OWL DL compliant reasoning tool can be used to decide **SROIQ** knowledge base satisfiability as well as any other reasoning task which can be reduced to it.

Terms

OWL	DL	FOL
class name	concept name	unary predicate
class	concept	formula with one free variable
object property name	role name	binary predicate
object property	role	formula with two free variables
ontology	knowledge base	theory
axiom	axiom	sentence
vocabulary	vocabulary / signature	signature

Table: Synopsis of the corresponding terms used in the OWL vs. the DL vs first-order logic.

Translating DL KBs into OWL

- Translating **SROIQ** knowledge OWL 2 DL ontology
  - Satisfiability and entailment checks can be performed by OWL reasoning engines.

Technical Issue Considerations

- Both the used vocabulary as well as the constructors have to be URIs
  - The URIs for the used individual, concept, and role names can be chosen rather arbitrarily,
  - while the URIs for constructors etc. are prescribed and associated to specific namespaces usually associated to the prefixes owl:, rdfs:, rdf:, and xsd:.

Technical Issue Considerations ...

- The mainly used encoding of OWL is as an RDF document
    a. advantageous from a downward compatibility and tool interoperability point of view;

The translation of a SROIQ knowledge base KB contains three parts:

b. a preamble containing the definition of namespaces,
c. declarations of the used concept (resp. class) and role (resp. object property) names,
d. and finally a part containing the OWL counterparts of the axioms from KB

$$[\![\mathcal{KB}]\!] = \mathrm{Pre} + \mathrm{Dec}(\mathcal{KB}) + \sum_{\alpha \in \mathcal{KB}} [\![\alpha]\!]$$

where + denotes concatenation of strings.

$$\mathrm{Pre} = \begin{cases} \texttt{@prefix owl: <http://www.w3.org/2002/07/owl\#> .} \\ \texttt{@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema\#> .} \\ \texttt{@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#> .} \\ \texttt{@prefix xsd: <http://www.w3.org/2001/XMLSchema\#> .} \end{cases}$$

$$\mathrm{Dec}(\mathcal{KB}) = \sum_{A \in \mathsf{N}_C(\mathcal{KB})} A \; \texttt{rdf:type} \; \texttt{owl:Class} \; . \\ + \sum_{r \in \mathsf{N}_R(\mathcal{KB})} r \; \texttt{rdf:type} \; \texttt{owl:ObjectProperty} \; .$$

Fig: Declarations are expressed by according typing statements:

Example

RBox $\mathcal{R}$

$$\texttt{owns} \sqsubseteq \texttt{caresFor}$$

"If somebody owns something, they care for it."

TBox $\mathcal{T}$

$$\texttt{Healthy} \sqsubseteq \neg\texttt{Dead}$$

"Healthy beings are not dead."

$$\texttt{Cat} \sqsubseteq \texttt{Dead} \sqcup \texttt{Alive}$$

"Every cat is dead or alive."

$$\texttt{HappyCatOwner} \sqsubseteq \exists\texttt{owns.Cat} \sqcap \forall\texttt{caresFor.Healthy}$$

"A happy cat owner owns a cat and all beings he cares for are healthy."

ABox $\mathcal{A}$

$$\texttt{HappyCatOwner}\,(\texttt{schr\"odinger})$$

"Schrödinger is a happy cat owner."

RBox $\mathcal{R}$

$\texttt{owns} \sqsubseteq \texttt{caresFor}$

"If somebody owns someth

TBox $\mathcal{T}$

$\texttt{Healthy} \sqsubseteq \neg\texttt{Dead}$

"Healthy beings are not de

$\texttt{Cat} \sqsubseteq \texttt{Dead} \sqcup \texttt{Alive}$

"Every cat is dead or alive

$\texttt{HappyCatOwner} \sqsubseteq \exists\texttt{owns.Cat} \sqcap \forall\texttt{caresFor.Heal}$

"A happy cat owner owns
he cares for are healthy."

ABox $\mathcal{A}$

HappyCatOwner (schrödinger)

"Schrödinger is a happy ca

```
@prefix :     <http://www.example.org/#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .

:owns          rdf:type owl:ObjectProperty .
:caresFor      rdf:type owl:ObjectProperty .
:Cat           rdf:type owl:Class .
:Dead          rdf:type owl:Class .
:Alive         rdf:type owl:Class .
:Healthy       rdf:type owl:Class .
:HappyCatOwner rdf:type owl:Class .

:owns          rdfs:subPropertyOf :caresFor .

:Healthy       rdfs:subClassOf [ owl:complementOf :Dead ] .
:Cat           rdfs:subClassOf [ owl:unionOf (:Dead :Alive) ] .
:HappyCatOwner rdfs:subClassOf
       [ owl:intersectionOf
          ( [ rdf:type owl:Restriction ;
              owl:onProperty :owns ; owl:someValuesFrom :Cat ]
            [ rdf:type owl:Restriction ;
              owl:onProperty :caresFor ; owl:allValuesFrom :Healthy] )
       ] .

:schrödinger   rdf:type :HappyCatOwner .
```

Expressing OWL Axioms in SROIQ

- OWL specification features much more axiom types than the ones used above to translate SROIQ knowledge bases.

Axiom type	Turtle notation	DL paraphrase
Class Equivalence	$[C]_C$ `owl:equivalentClass` $[D]_C$.	$C \sqsubseteq D,\ D \sqsubseteq C$
Class Disjointness	$[C]_C$ `owl:disjointWith` $[D]_C$.	$C \sqcap D \sqsubseteq \bot$
Disjoint Classes	`[] rdf:type owl:AllDisjointClasses ;` `owl:members` ($[C_1]_C$ ... $[C_n]_C$) .	$C_i \sqcap C_j \sqsubseteq \bot$ for all $1 \leq i < j \leq n$
Disjoint Union	$[C]_C$ `owl:disjointUnionOf` ($[C_1]_C$ ... $[C_n]_C$) .	$\bigsqcup_{i<j} C_i \sqsubseteq C$ $C_i \sqcap C_j \sqsubseteq \bot$ for all $1 \leq i < j \leq n$
Property Equivalence	$[r]_R$ `owl:equivalentProperty` $[s]_R$.	$r \sqsubseteq s,\ s \sqsubseteq r$