

Semantic Web Technologies

Web Ontology Language OWL

Jos de Bruijn
<http://www.debruijn.net>

KRDB Research Group
Free University of Bolzano, Italy

31 October 2008

Outline

Limitations of RDFS

Web Ontology Language OWL

- Design of OWL

- OWL Layering

- OWL and Description Logics

- OWL Syntaxes

RDFS as an Ontology Language

- ▶ Classes
- ▶ Properties
- ▶ Class hierarchies
- ▶ Property hierarchies
- ▶ Domain and range restrictions

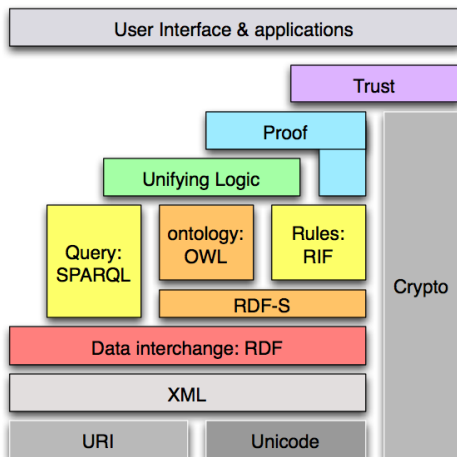
Expressive limitations of RDF(S)

- ▶ Only binary relations
- ▶ Characteristics of Properties (e.g. inverse, transitive, symmetric)
- ▶ Local range restrictions (e.g. for Class Person, the property `hasName` has range `xsd:string`)
- ▶ Complex concept descriptions (e.g. Person is defined by Man and Woman)
- ▶ Cardinality restrictions (e.g. a Person may have at most 1 name)
- ▶ Disjointness axioms (e.g. nobody can be both a Man and a Woman)

Layering issues

- ▶ Syntax
 - ▶ Only binary relations in RDF
 - ▶ Verbose Syntax
 - ▶ No limitations on graph in RDF
 - ▶ Every graph is valid
- ▶ Semantics
 - ▶ Malformed graphs
 - ▶ Use of vocabulary in language
 - ▶ e.g. $\langle \text{rdfs:Class}, \text{rdfs:subClassOf}, \text{ex:a} \rangle$
 - ▶ Meta-classes
 - ▶ e.g. $\langle \text{ex:a}, \text{rdf:type}, \text{ex:a} \rangle$

Where were we?



[following slides in part due to Frank van Harmelen

Stack of Languages

- ▶ XML
 - ▶ Surface syntax, no semantics
- ▶ XML Schema
 - ▶ Describes structure of XML documents
- ▶ RDF
 - ▶ Datamodel for “relations” between “things”
- ▶ RDF Schema
 - ▶ RDF Vocabulary Definition Language
- ▶ OWL
 - ▶ A more expressive Vocabulary Definition Language

RDF Schema recap

- ▶ RDFS provides
 - ▶ Classes
 - ▶ Class hierarchies
 - ▶ Properties
 - ▶ Property hierarchies
 - ▶ Domain and range restrictions
- ▶ RDFS does not provide
 - ▶ Property characteristics (inverse, transitive, ...)
 - ▶ Local range restrictions
 - ▶ Complex concept definitions
 - ▶ Cardinality restrictions
 - ▶ Disjointness axioms

Extending RDF Schema

- ▶ OWL extends RDF Schema to a full-fledged knowledge representation language for the Web
 - ▶ Logical expressions (and, or, not)
 - ▶ (in)equality
 - ▶ local properties
 - ▶ required/optional properties
 - ▶ required values
 - ▶ enumerated classes
 - ▶ symmetry, inverse

Design Goals for OWL

- ▶ **Shareable**
- ▶ **Changing** over time
- ▶ **Interoperability**
- ▶ **Inconsistency** detection
- ▶ Balancing **expressivity and complexity**
- ▶ **Ease of use**
- ▶ Compatible with **existing standards**
- ▶ **Internationalisation**

Requirements for OWL

- ▶ Ontologies are **object on the Web**
- ▶ with **their own meta-data**, versioning, etc...
- ▶ Ontologies are **extendable**
- ▶ They contain **classes, properties, data-types, range/domain, individuals**
- ▶ **Equality** (for classes, for individuals)
- ▶ **Classes as instances**
- ▶ **Cardinality** constraints
- ▶ **XML** syntax

Objectives for OWL

Objectives:

- ▶ layered language
- ▶ complex datatypes
- ▶ digital signatures
- ▶ decidability (in part)
- ▶ local unique names (in part)

Disregarded:

- ▶ default values
- ▶ closed world option
- ▶ property chaining
- ▶ arithmetic
- ▶ string operations
- ▶ partial imports
- ▶ view definitions
- ▶ procedural attachments

Language layers of OWL

- ▶ OWL Lite
 - ▶ Classification hierarchy
 - ▶ Simple constraints
- ▶ OWL DL
 - ▶ Maximal expressiveness
 - ▶ While maintaining tractability
 - ▶ Standard formalisation in a DL
- ▶ OWL Full
 - ▶ Very high expressiveness
 - ▶ Losing tractability
 - ▶ All syntactic freedom of RDF (self-modifying)

Features of OWL language layers

▶ OWL Lite

- ▶ (sub)classes, individuals
- ▶ (sub)properties, domain, range
- ▶ conjunction
- ▶ (in)equality
- ▶ cardinality 0/1
- ▶ datatypes
- ▶ inverse, transitive, symmetric properties
- ▶ someValuesFrom
- ▶ allValuesFrom

▶ OWL DL

- ▶ Negation
 - ▶ Disjunction
 - ▶ Full cardinality
 - ▶ Enumerated types
 - ▶ hasValue
- ### ▶ OWL Full
- ▶ Meta-classes
 - ▶ Modify language

OWL Full

- ▶ **No restriction on use of vocabulary** (as long as legal RDF)
 - ▶ Classes as instances (and much more)
- ▶ **RDF style model theory**
 - ▶ Reasoning using FOL engine
 - ▶ Semantics should correspond to OWL DL for restricted KBs

OWL DL

- ▶ Use of vocabulary restricted
 - ▶ Can't be used to do “nasty things” (e.g. modify OWL)
 - ▶ No classes as instances
 - ▶ Defined by abstract syntax
- ▶ Standard DL-based model theory
 - ▶ Direct correspondence with a DL
 - ▶ Reasoning via DL engines

OWL Lite

- ▶ No explicit negation or union
- ▶ Restricted cardinality (0/1)
- ▶ No nominals (oneOf)
- ▶ DL-based semantics
 - ▶ Reasoning via DL engines (+datatypes)
- ▶ Semantically, only small restriction on OWL DL
 - ▶ No nominals
 - ▶ No arbitrary cardinality

OWL and Description Logics

- ▶ OWL Lite corresponds to the DL $\mathcal{SHIF}(\mathbf{D})$
 - ▶ Named classes (A)
 - ▶ Named properties (P)
 - ▶ Individuals ($C(o)$)
 - ▶ Property values ($P(o, a)$)
 - ▶ Intersection ($C \sqcap D$)
 - ▶ Union(!) ($C \sqcup D$)
 - ▶ Negation(!) ($\neg C$)
 - ▶ Existential value restrictions ($\exists P.C$)
 - ▶ Universal value restrictions ($\forall P.C$)
 - ▶ Unqualified (0/1) number restrictions ($\geq nP, \leq nP, = nP$),
 $0 \leq n \leq 1$
- ▶ OWL DL corresponds to the DL $\mathcal{SHOIN}(\mathbf{D})$
 - ▶ Arbitrary number restrictions ($\geq nP, \leq nP, = nP$), $0 \leq n$
 - ▶ Property value ($\exists P.\{o\}$)
 - ▶ Enumeration ($\{o_1, \dots, o_n\}$)

OWL constructs

OWL Construct	DL	Example
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	<i>Human</i> \sqcap <i>Male</i>
unionOf	$C_1 \sqcup \dots \sqcup C_n$	<i>Doctor</i> \sqcup <i>Lawyer</i>
complementOf	$\neg C$	\neg <i>Male</i>
oneOf	$\{o_1, \dots, o_n\}$	$\{john, mary\}$
allValuesFrom	$\forall P.C$	$\forall hasChild.Doctor$
someValuesFrom	$\exists P.C$	$\exists hasChild.Lawyer$
value	$\exists P.\{o\}$	$\exists citizenOf.USA$
minCardinality	$\geq nP.C$	$\geq 2hasChild.Lawyer$
maxCardinality	$\leq nP.C$	$\leq 1hasChild.Male$
cardinality	$= nP.C$	$= 1hasParent.Female$

+ XML Schema datatypes: int, string, real, etc...

OWL axioms

OWL Axiom	DL	Example
SubClassOf	$C_1 \sqsubseteq C_2$	<i>Human</i> \sqsubseteq <i>Animal</i> \sqcap <i>Biped</i>
EquivalentClasses	$C_1 \equiv \dots \equiv C_n$	<i>Man</i> \equiv <i>Human</i> \sqcap <i>Male</i>
SubPropertyOf	$P_1 \sqsubseteq P_2$	<i>hasDaughter</i> \sqsubseteq <i>hasChild</i>
EquivalentProperties	$P_1 \equiv \dots \equiv P_n$	<i>cost</i> \equiv <i>price</i>
SameIndividual	$o_1 = \dots = o_n$	<i>President_Bush</i> = <i>G_W_Bush</i>
DisjointClasses	$C_i \sqsubseteq \neg C_j$	<i>Male</i> $\sqsubseteq \neg$ <i>Female</i>
DifferentIndividuals	$o_i \neq o_j$	<i>john</i> \neq <i>peter</i>
inverseOf	$P_1 \equiv P_2^-$	<i>hasChild</i> \equiv <i>hasParent</i> ⁻
Transitive	$P^+ \sqsubseteq P$	<i>ancestor</i> ⁺ \sqsubseteq <i>ancestor</i>
Symmetric	$P \equiv P^-$	<i>connectedTo</i> \equiv <i>connectedTo</i> ⁻

More on OWL species

- ▶ OWL Full is **not** a Description Logic
- ▶ OWL Lite has strong syntactic restrictions, but only limited semantics restrictions, compared with OWL DL
 - ▶ Negation can be encoded using disjointness
 - ▶ With negation and conjunction, you can encode disjunction

Class(C complete unionOf(B C))

is equivalent to:

DisjointClasses(notB B)

DisjointClasses(notC C)

Class(notBandnotC complete notB notC)

DisjointClasses(notBandnotC BorC)

Class(C complete notBandnotC)

More on layering

- ▶ For an OWL DL-restricted KB, OWL Full semantics is **not** equivalent to OWL DL semantics

John friend Susan .

OWL Full entails:

John rdf:type owl:Thing . Susan rdf:type owl:Thing . friend
rdf:type owl:ObjectProperty .

John rdf:type _:x . _:x owl:onProperty friend . _:x
owl:minCardinality "1"^^xsd:nonNegativeInteger .

Syntaxes of OWL

- ▶ RDF
 - ▶ Official exchange syntax
 - ▶ Hard for humans
 - ▶ RDF parsers are hard to write!
- ▶ UML
 - ▶ Large user base
- ▶ XML
 - ▶ Not the RDF syntax
 - ▶ Better for humans
 - ▶ More XML than RDF tools available
- ▶ Abstract syntax
 - ▶ Not defined for OWL Full
 - ▶ Human readable

OWL in RDF/XML

Example from [OwlGuide]:

```
<!ENTITY vin
"http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#" >
<!ENTITY food
"http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#" > ...
<rdf:RDF
xmlns:vin="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
... >
```

```
<owl:Class rdf:ID="Wine" > <rdfs:subClassOf
rdf:resource="&food;PotableLiquid" /> <rdfs:label
xml:lang="en" >wine</rdfs:label> <rdfs:label
xml:lang="fr" >vin</rdfs:label> ... </owl:Class>
```

```
<owl:Class rdf:ID="Pasta" > <rdfs:subClassOf
rdf:resource=" #EdibleThing" /> ... </owl:Class> </rdf:RDF>
```

OWL Abstract syntax

```
Class( professor partial ) Class( associateProfessor partial  
academicStaffMember)
```

```
DisjointClasses ( associateProfessor assistantProfessor )
```

```
DisjointClasses ( professor associateProfessor )
```

```
Class( faculty complete academicStaffMember)
```

OWL Abstract syntax

In DL syntax:

associateProfessor \sqsubseteq academicStaffMember

associateProfessor $\sqsubseteq \neg$ assistantProfessor

professor $\sqsubseteq \neg$ associateProfessor

faculty \equiv academicStaffMember

More examples

```
DatatypeProperty( age range( xsd:nonNegativeInteger ) )  
ObjectProperty( lecturesIn )
```

```
ObjectProperty( isTaughtBy domain( course ) range( academicStaffMember ) )  
SubPropertyOf( isTaughtBy involves )
```

```
ObjectProperty( teaches inverseOf( isTaughtBy )  
domain( academicStaffMember ) range( course ) )
```

```
EquivalentProperties ( lecturesIn teaches )
```

```
ObjectProperty( hasSameGradeAs Transitive Symmetric domain( student )  
range( student ) )
```

More examples

In DL syntax:

$\top \sqsubseteq \forall \text{age.xsd} : \text{nonNegativeInteger}$

$\top \sqsubseteq \forall \text{isTaughtBy}^- . \text{course}$

$\top \sqsubseteq \forall \text{isTaughtBy} . \text{academicStaffMember}$

$\text{isTaughtBy} \sqsubseteq \text{involves}$

$\text{teaches} \equiv \text{isTaughtBy}^-$

$\top \sqsubseteq \forall \text{teaches}^- . \text{academicStaffMember}$

$\top \sqsubseteq \forall \text{teaches} . \text{course}$

$\text{lecturesIn} \equiv \text{teaches}$

$\text{hasSameGradeAs}^+ \sqsubseteq \text{hasSameGradeAs}$

$\text{hasSameGradeAs} \equiv \text{hasSameGradeAs}^-$

$\top \sqsubseteq \forall \text{hasSameGradeAs}^- . \text{student}$

$\top \sqsubseteq \forall \text{hasSameGradeAs} . \text{student}$

More examples

Individual (949318 type(lecturer))

Individual (949352 type(academicStaffMember) value(age
"39"^^xsd:integer))

ObjectProperty(isTaughtBy Functional)

Individual (CIT1111 type(course) value(isTaughtBy 949352)
value(isTaughtBy 949318))

DifferentIndividuals (949318 949352) DifferentIndividuals (949352
949111 949318)

More examples

In DL syntax:

949318 : *lecturer*

949352 : *academicStaffMember*

$\langle 949352, "39" \text{^^} \&xsd; integer \rangle$: *age*

$\top \sqsubseteq \leq 1$ *isTaughtBy*

CIT1111 : *course*

$\langle CIT1111, 949352 \rangle$: *isTaughtBy*

$\langle CIT1111, 949318 \rangle$: *isTaughtBy*

949318 \neq 949352

949352 \neq 949111

949111 \neq 949318

949352 \neq 949318

More examples

```
Class( firstYearCourse partial restriction (isTaughtBy allValuesFrom  
( Professor )))
```

```
Class(mathCourse partial restriction (isTaughtBy hasValue (949352)))
```

```
Class(academicStaffMember partial restriction (teaches someValuesFrom  
(undergraduateCourse)))
```

```
Class(course partial restriction (isTaughtBy minCardinality(1)))
```

```
Class(department partial restriction (hasMember minCardinality(10))  
restriction (hasMember maxCardinality(30)))
```

More examples

In DL syntax:

firstYearCourse $\sqsubseteq \forall isTaughtBy. Professor$

mathCourse $\sqsubseteq \exists isTaughtBy. \{949352\}$

academicStaffMember $\sqsubseteq \exists teaches. undergraduateCourse$

course $\sqsubseteq_{\geq 1} isTaughtBy$

department $\sqsubseteq_{\geq 10} hasMember \sqcap \leq 30 hasMember$

More examples

```
Class(course partial complementOf(staffMember))
```

```
Class(peopleAtUni complete unionOf(staffMember student))
```

```
Class(facultyInCS complete intersectionOf ( faculty  
restriction (belongsTo hasValue (CSDepartment))))
```

```
Class(adminStaff complete intersectionOf ( staffMember  
complementOf(unionOf(faculty techSupportStaff))))
```

More examples

In DL syntax:

$course \sqsubseteq \neg staffMember$

$peopleAtUni \equiv staffMember \sqcup student$

$facultyInCS \equiv faculty \sqcap \exists belongsTo. \{CSDepartment\}$

$adminStaff \equiv staffMember \sqcap \neg (faculty \sqcup techSupportStaff)$

Summary

Limitations of RDFS

Web Ontology Language OWL

- Design of OWL

- OWL Layering

- OWL and Description Logics

- OWL Syntaxes