Semantic Technologies

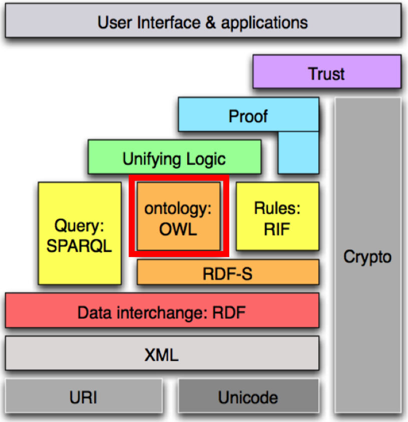
Part 14: OWL – Syntax and Intuitive Semantics

Werner Nutt

Acknowledgment

These slides are based on the Latex version of slides
by Markus Krötzsch of TU Dresden

OWL


User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
Unicode

Agenda

Ontology in Philosophy

- Notion exists only in sigular (no “ontologies”)
- Denotes the “study of being”
- Can be found in philosophical writings of Aristotle (Socrates), Thomas Aquinas, Descartes, Kant, Hegel, Wittgenstein, Heidegger, Quine, . . .
- Term first mentioned in 17th century

Ontology in Computer Science

Gruber (1993):

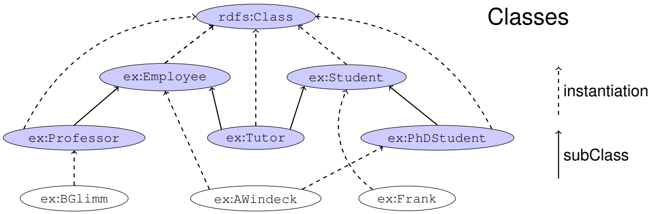
“An Ontology is a

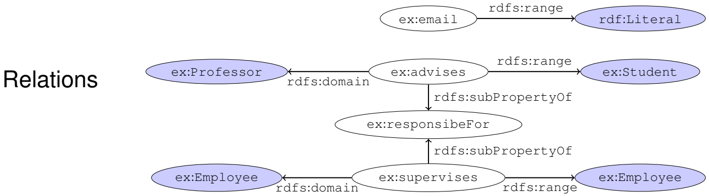
formal specification	$\Rightarrow$ interpretable by machines
of a **shared**	$\Rightarrow$ based on consensus
conceptualization	$\Rightarrow$ describes relevant notions
of a **domain** of interest”	$\Rightarrow$ referring to a “topic”

Ontologies in Practice: Some Requirements

- instantiation of classes by individuals
- conceptual hierarchies (taxonomies, “inheritance”): classes, concepts
- binary relations between individuals: properties, roles
- characteristics of relations (z.B. range, transitive)
- datatypes (e.g. numbers): concrete domains
- logical operators
- clear semantics

RDFS – Simple Ontologies





RDF Schema as Ontology Language?

- Appropriate for simple ontologies
- Advantage: automated inferencing relatively efficient
- But: not appropriate for more complex modeling
- Resort to more expressive languages, like
  - OWL
  - RIF ...

Agenda

OWL – General Remarks

- W3C Recommendation since 2004
- semantic fragment of FOL
- three variants:
  - OWL Lite
  - OWL DL
  - OWL Full
- no reification in OWL DL
  $\leadsto$ RDFS is fragment of OWL Full
- OWL DL is decidable
  corresponds to description logic SHOIN(D)
- W3C documents contain details that cannot all be covered here

OWL 1 Variants

- OWL Full
  - contains OWL DL and OWL Lite
  - contains all of RDFS (as the only OWL variant)
  - undecidable inferences
  - limited support by tools
- OWL DL
  - contains OWL Lite and is sublanguage of OWL Full
  - widely supported by tools
  - worst-case complexity: NExpTime (= non-deterministic exponential time)
- OWL Lite
  - sublanguage of OWL DL and OWL Full
  - low expressivity
  - worst-case complexity: ExpTime (= exponential time)

OWL Documents

- . . . are RDF documents
  (at least in the standard syntax; there are others)
- . . . consist of
  - head with general information
  - rest with actual ontology

Head of an OWL Document

Definition of name spaces in the root

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

Head of an OWL Document

General information

```
<owl:Ontology rdf:about="">
  <rdfs:comment
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
   SWRC ontology, version of June 2007
  </rdfs:comment>
  <owl:versionInfo>v0.7.1</owl:versionInfo>
  <owl:imports rdf:resource="http://www.example.org/foo" />
  <owl:priorVersion
    rdf:resource="http://ontoware.org/projects/swrc" />
</owl:Ontology>
```

Head of an OWL Document

taken from RDFS
`rdfs:comment`
`rdfs:label`
`rdfs:seeAlso`
`rdfs:isDefinedBy`

in addition
`owl:imports`

for versioning
`owl:versionInfo`
`owl:priorVersion`
`owl:backwardCompatibleWith`
`owl:incompatibleWith`
`owl:DeprecatedClass`
`owl:DeprecatedProperty`

Agenda

Classes, Roles and Individuals

Three building blocks of ontology axioms

- classes
  - comparable with classes in RDFS
- individuals
  - comparable with “proper” instances in RDFS
- roles
  - comparable with properties in RDFS

Classes

Definition

- `<owl:Class rdf:about ="Professor"/>`
- equivalent to

```
<rdf:Description rdf:about="Professor">
  <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```

Pre-defined

- `owl:Thing`
- `owl:Nothing`

Individuals

Definition via class membership

```
<rdf:Description rdf:about="francescoRicci">
  <rdf:type rdf:resource="Professor"/>
</rdf:Description>
```

equivalent:

```
<Professor rdf:about="francescoRicci"/>
```

Abstract Roles (= Object Properties)

Abstract roles are defined in a way similar to classes

```
<owl:ObjectProperty rdf:about="hasAffiliation" />
```

Abstract roles connect individuals

Domain and range of abstract roles

```
<owl:ObjectProperty rdf:about="hasAffiliation">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="Organization" />
</owl:ObjectProperty>
```

Concrete Roles (= Datatype Properties)

Concrete roles have datatypes as range

```
<owl:DatatypeProperty rdf:about="firstName" />
```

Concrete roles connect individuals with data values

Domain and range of concrete roles

```
<owl:DatatypeProperty rdf:about="firstName">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

Many XML datatypes can be used

Individuals and Roles

```
<Person rdf:about="francescoRicci">
   <hasAffiliation rdf:resource="unibz" />
   <hasAffiliation rdf:resource="facultyCS" />
   <firstName rdf:datatype="&xsd;string">
     Francesco
   </firstName>
</Person>
```

In general roles are not functional, that is, one individual can be connected to more than one individual (or value)

Agenda

Simple Class Relationships: Subclasses

```
<owl:Class rdf:about="Professor">
 <rdfs:subClassOf rdf:resource="FacultyMember" />
</owl:Class>
<owl:Class rdf:about="FacultyMember">
 <rdfs:subClassOf rdf:resource="Person" />
</owl:Class>
```

It logically follows that `Professor` is a subclass of `Person`

Simple Class Relationships: Disjointness

```
<owl:Class rdf:about="Professor">
  <rdfs:subClassOf rdf:resource="FacultyMember" />
</owl:Class>
<owl:Class rdf:about="Book">
  <rdfs:subClassOf rdf:resource="Publication" />
</owl:Class>
<owl:Class rdf:about="FacultyMember">
  <owl:disjointWith rdf:resource="Publication" />
</owl:Class>
```

It logically follows that `Professor` and `Book` are also disjoint classes

Simple Class Relationships: Class Equivalence

```
<owl:Class rdf:about="Man">
  <rdfs:subClassOf rdf:resource="Person" />
</owl:Class>
<owl:Class rdf:about="Person">
  <owl:equivalentClass rdf:resource="Human" />
</owl:Class>
```

It logically follows that `Man` is a subclass of `Human`

Individuals and Class Relationships

```
<Book rdf:about="http://semantic-web-book.org/uri">
   <author rdf:resource="pascalHitzler" />
   <author rdf:resource="markusKroetzsch" />
   <author rdf:resource="sebastianRudolph" />
</Book>
<owl:Class rdf:about="Book">
   <rdfs:subClassOf rdf:resource="Publication" />
</owl:Class>
```

It logically follows that

`Foundations of Semantic Web Technologies`
is a `Publication`.

Relationships between Individuals (sameAs)

```
<Professor rdf:about="francescoRicci" />
  <rdf:Description rdf:about="francescoRicci">
  <owl:sameAs rdf:resource="professorRicci" />
</rdf:Description>
```

It logically follows that `professorRicci` is a `Professor`

Distinctness of individuals is expressed via `owl:differentFrom`.

Relationships between Individuals

```
<owl:AllDifferent>
<owl:distinctMembers rdf:parseType="Collection">
   <Person rdf:about="francescoRicci" />
   <Person rdf:about="diegoCalvanese" />
   <Person rdf:about="wernerNutt" />
</owl:distinctMembers>
</owl:AllDifferent>
```

This is an abbreviated notation instead of using several `owl:differentFrom`

Usage of `owl:AllDifferent` and `owl:distinctMembers` exclusively for this purpose

Closed Classes

```
<owl:Class rdf:about="TechniciansOfCS">
  <owl:oneOf rdf:parseType="Collection">
    <Person rdf:about="amantiaPano" />
    <Person rdf:about="konradHofer" />
  </owl:oneOf>
</owl:Class>
```

tells that there are only *exactly these two* `TechniciansOfCS`

Agenda

Logical Class Constructors

- logical `and` (conjunction):
  `owl:intersectionOf`
- logical `or` (disjunction):
  `owl:unionOf`
- logical `not` (negation):
  `owl:complementOf`

. . . used to construct complex classes from simple classesp

Conjunction

```
<owl:Class rdf:about="TechniciansOfCS">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="Technicians" />
    <owl:Class rdf:about="StaffOfCS" />
  </owl:intersectionOf>
</owl:Class>
```

it logically follows that all `TechniciansOfCS` are also `Technicians`

Disjunction

```
<owl:Class rdf:about="Professor">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="ActivelyTeaching" />
        <owl:Class rdf:about="Retired" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

Negation

```
<owl:Class rdf:about="FacultyMember">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:complementOf rdf:resource="Publication" />
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

Semantically equivalent:

```
<owl:Class rdf:about="FacultyMember">
  <owl:disjointWith rdf:resource="Publication" />
</owl:Class>
```

Role Restrictions (allValuesFrom)

Used to define complex classes via roles

```
<owl:Class rdf:about="Exam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasExaminer" />
      <owl:allValuesFrom rdf:resource="Professor" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

I.e., *all* examiners of an exam have to be professors

Role Restrictions (someValuesFrom)

```
<owl:Class rdf:about="Exam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasExaminer" />
      <owl:someValuesFrom rdf:resource="Person" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

I.e., every exam must have *at least one* examiner

Role Restrictions (Cardinalities)

```
<owl:Class rdf:about="Exam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasExaminer"/>
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">
        2
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

I.e., an exam may have *at most two* examiners

Role Restrictions (Cardinalities)

```
<owl:Class rdf:about="Exam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasTopic"/>
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">3
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

I.e., an exam must cover *at least three* topics

Role Restrictions (Cardinalities)

```
<owl:Class rdf:about="Exam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasTopic"/>
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">3
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

An exam must cover *exactly three* topics

Role Restrictions (hasValue)

```
<owl:Class rdf:about="ExamRicci">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasExaminer" />
      <owl:hasValue rdf:resource="francescoRicci" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

`owl:hasValue` always refers to one singular individual

The above is equivalent to the example on the next slide

Role Restrictions (hasValue)

```
<owl:Class rdf:about="ExamRicci">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasExaminer" />
      <owl:someValuesFrom>
        <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about="francescoRicci" />
        </owl:oneOf>
      </owl:someValuesFrom>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Agenda

Role Relationships

```
<owl:ObjectProperty rdf:about="hasExaminer">
  <rdfs:subPropertyOf rdf:resource="hasParticipant" />
</owl:ObjectProperty>
```

Likewise: `owl:equivalentProperty`

Roles can be inverses of each other:

```
<owl:ObjectProperty rdf:about="hasExaminer">
  <owl:inverseOf rdf:resource="examinerOf"/>
</owl:ObjectProperty>
```

Role Characteristics

- domain
- range
- transitivity, i.e.
  `r(a, b)` and `r(b, c)` imply `r(a, c)`
- symmetry, i.e.
  `r(a, b)` implies `r(b, a)`
- functionality
  `r(a, b)` and `r(a, c)` imply `b = c`
- inverse functionality
  `r(a, b)` and `r(c, b)` imply `a = c`

Domain and Range

```
<owl:ObjectProperty rdf:about="isMemberOf">
  <rdfs:range rdf:resource="Organization" />
</owl:ObjectProperty>
```

equivalent to:

```
<owl:Class rdf:about="&owl;Thing">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="isMemberOf" />
      <owl:allValuesFrom rdf:resource="Organization" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Domain and Range: Caution!

```
<owl:ObjectProperty rdf:about="isMemberOf">
  <rdfs:range rdf:resource="Organization" />
</owl:ObjectProperty>
<number rdf:about="five">
  <isMemberOf rdf:resource="PrimeNumbers" />
</number>
```

It follows that `PrimeNumbers` is an Organization!

Role Characteristics

```
<owl:ObjectProperty rdf:about="hasColleague">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hasProjectLeader">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="isProjectLeaderFor">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
</owl:ObjectProperty>
<Person rdf:about="francescoRicci">
  <hasColleague rdf:resource="diegoCalvanese" />
  <hasColleague rdf:resource="wernerNutt" />
  <isProjectLeaderFor rdf:resource="bzTraffic" />
</Person>
<Project rdf:about="optique">
  <hasProjectLeader rdf:resource="diegoCalvanese" />
  <hasProjectLeader rdf:resource="calvaneseDiego" />
</Project>
```

Consequences from the Example

- `diegoCalvanese hasColleague francescoRicci`
- `diegoCalvanese hasColleague wernerNutt`
- `diegoCalvanese owl:sameAs calvaneseDiego`

Agenda

OWL 1 Variants

- OWL Full
  - contains OWL DL and OWL Lite
  - contains all of RDFS (as the only OWL variant)
  - undecidable inferences
  - limited support by tools
- OWL DL
  - contains OWL Lite and is sublanguage of OWL Full
  - widely supported by tools
  - worst-case complexity: NExpTime (= non-deterministic exponential time)
- OWL Lite
  - sublanguage of OWL DL and OWL Full
  - low expressivity
  - worst-case complexity: ExpTime (= exponential time)

OWL Full

- Unrestricted use of all OWL and RDFS language elements (has to be valid RDFS)
- Difficult, e.g.: non-existent type separation (classes, roles, individuals), thus:
    - `owl:Thing` becomes the same as `rdfs:resource`
    - `owl:Class` becomes the same as `rdfs:Class`
    - `owl:DatatypeProperty` becomes a subclass of `owl:ObjectProperty`
    - `owl:ObjectProperty` becomes the same as `rdf:Property`

Example for Confusion of Types in OWL Full

```
<owl:Class rdf:about="Book">
  <germanName rdf:datatype="&xsd;string">Buch</germanName>
  <frenchName rdf:datatype="&xsd;string">livre</frenchName>
</owl:Class>
```

Inferences about such constructs are rarely needed in practice

OWL DL

- Only usage of RDFS language elements that are explicitly allowed (like those in our examples)
  not allowed: `rdfs:Class`, `rdfs:Property`
- Type separation: classes and roles have to be explicitly declared
- Concrete roles must not be specified as transitive, symmetric, inverse or inverse functional
- Number restrictions must not be used with transitive roles, their subroles, or inverses thereof

OWL Lite

- All restrictions of OWL DL
- Moreover:
  - not allowed: `oneOf`, `unionOf`, `complementOf`, `hasValue`, `disjointWith`
  - number restrictions only allowed with 0 and 1
  - some constraints referring to anonymous (complex) classes, e.g., only in the subject of `rdfs:subClassOf`

Agenda

Terminological Queries to OWL Ontologies

- Class equivalence
- Subclass relationships
- Disjointness of classes
- Global consistency (aka satisfiability)
- Class consistency: a class is *inconsistent* if it is equivalent to `owl:Nothing` – this hints at a modeling error:

```
<owl:Class rdf:about="Book">
  <owl:subClassOf rdf:resource="Publication"/>
  <owl:disjointWith rdf:resource="Publication"/>
</owl:Class>
```

Assertional Queries to OWL Ontologies

- Instance checking: does a given individual belong to a given class?
- Search for all individuals that are members of a given class
- Are two given individuals linked by a role?
- Search for all individual pairs that are linked by a certain role
- . . . caution: only “provable” answers will be given!

OWL 1 Language Elements

Head

- `rdfs:comment`
- `rdfs:label`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
- `owl:versionInfo`
- `owl:priorVersion`
- `owl:backwardCompatibleWith`
- `owl:incompatibleWith`
- `owl:DeprecatedClass`
- `owl:DeprecatedProperty`
- `owl:imports`

Relationships between individuals

- `owl:sameAs`
- `owl:differentFrom`
- `owl:AllDifferent`
- `owl:distinctMembers`

Pre-defined datatypes (OWL 1)

- `xsd:string`
- `xsd:integer`

OWL Language Elements

Class constructors and relationships

- `owl:Class`
- `owl:Thing`
- `owl:Nothing`
- `rdfs:subClassOf`
- `owl:disjointWith`
- `owl:equivalentClass`
- `owl:intersectionOf`
- `owl:unionOf`
- `owl:complementOf`

Role restrictions

- `owl:allValuesFrom`
- `owl:someValuesFrom`
- `owl:hasValue`
- `owl:cardinality`
- `owl:minCardinality`
- `owl:maxCardinality`
- `owl:oneOf`

OWL Language Elements

Role constructors, relationships and characteristics

- `owl:ObjectProperty`
- `owl:DatatypeProperty`
- `rdfs:subPropertyOf`
- `owl:equivalentProperty`
- `owl:inverseOf`
- `rdfs:domain`
- `rdfs:range`
- `owl:TransitiveProperty`
- `owl:SymmetricProperty`
- `owl:FunctionalProperty`
- `owl:InverseFunctionalProperty`

Further Literature

- `http://www.w3.org/2004/OWL/`
  central W3C web page for OWL
- `http://www.w3.org/TR/owl-features/`
  overview over OWL
- `http://www.w3.org/TR/owl-ref/`
  comprehensive description of the OWL language components
- `http://www.w3.org/TR/owl-guide/`
  introduction into OWL knowledge modeling
- `http://www.w3.org/TR/owl-semantics/`
  describes the semantics of OWL and the abstract syntax for OWL DL
  ($\leadsto$ later lecture)