Semantic Web Modeling Languages
Lecture III: More OWL

Markus Krötzsch & Sebastian Rudolph
ESSLLI 2009 Bordeaux

• Advanced Features of OWL
  – more class constructors
  – extended property modeling
  – handling of data values
More Complex Classes: Qualified At-Least Restriction

- \[
  \text{[ rdf:type owl:Restriction; \}
  \text{owl:minQualifiedCardinality } "n"\text{^^xsd:nonNegativeInteger; \}
  \text{owl:onProperty prop; owl:onClass class ]}
\]

- \[ I_C(...)=\{x| \# \{ (x,y)\in I_R(prop) | y\in I_C(class) \} > n \}\]

- Example:
  \[
  \text{[ rdf:type owl:Restriction; \}
  \text{owl:minQualifiedCardinality } "2"\text{^^xsd:nonNegativeInteger; \}
  \text{owl:onClass ex:Male; \}
  \text{owl:onProperty ex:parentOf ]}
\]
• in analogy to at-least restrictions:
  – at-most:
    \texttt{owl:maxQualifiedCardinality}
  – exact cardinality:
    \texttt{owl:QualifiedCardinality}
More Complex Classes: Enumeration of Individuals

- \([\text{owl:oneOf} (\text{induril}, \ldots, \text{indurin})]\)
- \(I_c([\text{owl:oneOf} (\text{induril}, \ldots, \text{indurin})]) = \{I_I(\text{induril}), \ldots, I_I(\text{indurin})\}\)

- Example:
  \([\text{owl:oneOf} (\text{ex:georgec}, \text{ex:arnolds})]\)
More Complex Classes: Self Restriction

• [ rdf:type owl:Restriction ;
  owl:onProperty prop ;
  owl:hasSelf "true"^^xsd:boolean ]

• $I_C(...) = \{ x | \langle x, x \rangle \in I_R(prop) \}$

• Example: [ rdf:type owl:Restriction ;
  owl:onProperty ex:hasKilled ;
  owl:hasSelf "true"^^xsd:boolean ]

$IC \downarrow$

IR
Inverse Properties

- $[\text{owl:inverseOf prop}]
- \text{I}_R([\text{owl:inverseOf prop}]) = \{\langle y, x \rangle \mid \langle x, y \rangle \in \text{I}_R(prop)\}
- \text{Example: } [\text{owl:inverseOf ex:childOf}]

\[
\text{I}_R \downarrow
\]

\[
\text{I}_R
\]
Property Chain Axioms

- \( \text{prop \ owl:propertyChainAxiom} (\text{prop}_1, \ldots, \text{prop}_n) \).
- True in \( I \), if \( I_R(\text{prop}_1) \circ \ldots \circ I_R(\text{prop}_n) \subseteq I_R(\text{prop}) \).
- Example:

\[
\text{ex:siblingOf \ owl:propertyChainAxiom} (\text{ex:childOf}, \text{ex:parentOf}).
\]
Decidability problems

• role chain axioms can easily lead to undecidability
• in order to retain decidability, two global constraints are imposed on OWL DL ontologies:
  – the set of property chain axioms and subproperty statements must be regular
  – properties used in cardinality and self restrictions must be simple properties
in the following, we abbreviate

\[ R \text{ owl:propertyChainAxiom} (S_1 \ldots S_n) \]

by \( S_1 \circ \ldots \circ S_n \sqsubseteq R \)

\[ S \text{ owlrdfs:subPropertyOf} R \]

by \( S \sqsubseteq R \)

- regularity restriction:
  - there must be a strict linear order \( \prec \) on the properties
  - every property chain or subproperty axiom has to have one of the following forms where \( S_i \prec R \) for all \( i = 1, 2, \ldots, n \):

\[
\begin{align*}
R \circ R & \sqsubseteq R & [\text{owl:inverseOf } R] & \sqsubseteq R & S_1 \circ S_2 \circ \ldots \circ S_n & \sqsubseteq R \\
R \circ S_1 \circ S_2 \circ \ldots \circ S_n & \sqsubseteq R & S_1 \circ S_2 \circ \ldots \circ S_n \circ R & \sqsubseteq R
\end{align*}
\]

- Example 1: \( R \circ S \sqsubseteq R \quad S \circ S \sqsubseteq S \quad R \circ S \circ R \sqsubseteq T \)
  - regular with order \( S \prec R \prec T \)

- Example 2: \( R \circ T \circ S \sqsubseteq T \)
  - not regular because form not admissible

- Example 3: \( R \circ S \sqsubseteq S \quad S \circ R \sqsubseteq R \)
  - not regular because no adequate order exists
• combining property chain axioms and cardinality or self restrictions may lead to undecidability
• restriction: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)
• technically:
  – for any property chain axiom \( S_1 \circ S_2 \circ ... \circ S_n \subseteq R \) with \( n>1 \), \( R \) is non-simple
  – for any subproperty axiom \( S \subseteq R \) with \( S \) non-simple, \( R \) is non-simple
  – all other properties are simple
• Example:
\[
Q \circ P \subseteq R \quad R \circ P \subseteq R \quad R \subseteq S \quad P \subseteq R \quad Q \subseteq S
\]
non-simple: \( R, S \)  simple: \( P, Q \)
• OWL also allows for specifying that properties are:
  – disjoint from another
  – functional
  – inverse functional
  – transitive
  – symmetric
  – asymmetric
  – reflexive
  – irreflexive

syntactic sugar w.r.t.
already introduced
modeling features
Datatypes in OWL

• like in RDF, properties can also be used to associate individuals with data values:

  ex:john ex:hasAge "42"^^xsd:integer .

• those *datatype properties* must not be used as individual-interrelating *object properties* at the same time

• datatypes supported by OWL:

  owl:real, owl:rational, xsd:decimal, xsd:integer,
  xsd:nonNegativeInteger, xsd:nonPositiveInteger,
  xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int,
  xsd:short, xsd:byte, xsd:unsignedLong, xsd:unsignedInt,
  xsd:unsignedShort, xsd:unsignedByte, xsd:double, xsd:float,
  xsd:string, xsd:normalizedString, xsd:token, xsd:language,
  xsd:Name, xsd:NCName, xsd:NMTOKEN, xsd:boolean,
  xsd:hexBinary, xsd:base64Binary, xsd:anyURI, xsd:dateTime,
  xsd:dateTimeStamp, rdf:XMLLiteral
Datatype Ranges

- Property ranges for datatype properties:
  Datatypes (e.g. from XML Schema)

- Example:

```ontologies
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
...
ex:hasAge rdfs:range xsd:integer .
```

- Interpretation of datatypes defined in XML Schema (OWL adds some clarifications, e.g. “Do floating point and integer numbers overlap?”)

- Attention: datatypes still have to be explicitly specified in RDF and OWL! Given the above axiom, we find:

```ontologies
ex:jean ex:hasAge "17"^^xsd:integer . ← Correct
ex:paul ex:hasAge "23"^^xsd:decimal . ← Correct
ex:claire ex:hasAge "42" . ← Inconsistent!
```
Defining New Datatypes

- XML Schema has ways of restricting datatypes → **datatype facets**

- Example:

```owl
ex:personAge owl:equivalentClass
  [ rdf:type rdfs:Datatype;
    owl:onDatatype xsd:integer;
    owl:withRestrictions (  
      [ xsd:minInclusive "0"^^xsd:integer ]
      [ xsd:maxInclusive "150"^^xsd:integer ]
    )
  ] .
```

- Possible facets depend on datatype, some facets restricted in OWL → see specs for details
Simple Data Integration in OWL

- Practical problem: given ontologies from different sources, which URIs refer to the same individuals?

- Typical approaches in OWL:
  - Explicitly specify equality with owl:sameAs
  - Use inverse functional properties ("same values → same individual")

- Problems:
  - owl:sameAs requires explicit mappings (rare on the Web)
  - OWL DL disallows inverse functional datatype properties (complicated interplay with datatype definitions!)
  - Only one property used globally for identification, no property combinations (Example: "All ESSLLI participants with the same name and birthday are the same.")
OWL 2 Keys

OWL 2 provides a way to model
“All ESSLLI students with same name and birthday are the same.“

→ Keys

```
ex:ESSLLIStudent owl:hasKey (ex:name, ex:birthday) .
```

**Restriction:** Keys apply only to named individuals – objects of the interpretation domain to which a URI refers.

More explicitly:
If there are two URIs $u$ and $v$, and there is some name $n$ and birthday $b$ such that

```
u rdf:type ex:ESSLLIStudent; ex:name n ; ex:birthday b .
v rdf:type ex:ESSLLIStudent; ex:name n ; ex:birthday b .
```

then we conclude: $u$ owl:sameAs $v$ .
**Quo Vadis, OWL Lite?**

*OWL Lite as failure:*
- Defined as fragment of OWL 1 DL, intended to be simpler
- However: almost as complex as OWL DL (ExpTime)
- Complex syntax hides real expressive power
- Current usage in ontologies coincidental rather than intentionally

Original goal: simpler implementation and usage
→ approach in OWL 2: three simpler language profiles:
- **OWL 2 QL**
- **OWL 2 EL**
- **OWL 2 RL**
Design principle for profiles:
Identify maximal OWL 2 sublanguages that are still implementable in PTime.

Main source of intractability: **non-determinism** (requires guessing/backtracking)
- `owl:unionOf`, `or` `owl:complementOf` + `owl:intersectionOf`
- Max. cardinality restrictions
- Combining existentials (`owl:someValuesFrom`) and universals
  (`owl:allValuesFrom`) in superclasses
- Non-unary finite class expressions (`owl:oneOf`) or datatype expressions

→ features that are not allowed in any OWL 2 profile

Many further features can lead to non-determinism – care needed!
OWL 2 EL

**OWL profile based on description logic EL++**

- Intuition: focus on terminological expressivity used for light-weight ontologies
- **Allow** `owl:someValuesFrom` (existential) but not `owl:allValuesFrom` (universal)
- Property domains, class/property hierarchies, class intersections, disjoint classes/properties, property chains, `owl:hasSelf`, `owl:hasValue`, and keys fully supported
- No inverse or symmetric properties
- `rdfs:range` allowed but with some restrictions
- **No** `owl:unionOf` or `owl:complementOf`
- Various restrictions on available datatypes
OWL 2 EL: Features

- Standard reasoning in OWL 2 EL: PTime-complete
- Used by practically relevant ontologies: Prime example is SNOMED CT (clinical terms ontology with classes and properties in the order of $10^5$)
- Fast implementations available: full classification of SNOMED-CT in $<10$ min; real-time responsivity when preprocessed (modules)
**OWL 2 QL**

**OWL profile that can be used to query data-rich applications:**

- **Intuition:** use OWL concepts as light-weight queries, allow query answering using rewriting in SQL on top of relational DBs

- **Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`:**
  - subclasses can only be class names or `owl:someValuesFrom` (existential) with unrestricted (`owl:Thing`) filler
  - superclasses can be class names, `owl:someValuesFrom` or `owl:intersectionOf` with superclass filler (recursive), or `owl:complementOf` with subclass filler

- **Property hierarchies, disjointness, inverses, (a)symmetry supported, restrictions on range and domain**

- **Disjoint or equivalence of classes only for subclass-type expressions**

- **No** `owl:unionOf, owl:allValuesFrom, owl:hasSelf, owl:hasKey, owl:hasValue, owl:oneOf, owl:sameAs, owl:propertyChainAxiom, owl:TransitiveProperty`, **cardinalities, functional properties**

- **Some restrictions on available datatypes**
OWL 2 QL: Features

- Standard reasoning in OWL 2 QL: PTime, for some cases even LogSpace (<PTime)
- Convenient light-weight interface to legacy data
- Fast implementations on top of legacy database systems (relational or RDF): highly scalable to very large datasets
OWL profile that resembles an OWL-based rule language:

- Intuition: subclass axioms in OWL RL can be understood as rule-like implications with head (superclass) and body (subclass)
- Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`:
  - Subclasses can only be class names, `owl:oneOf`, `owl:hasValue`, `owl:intersectionOf`, `owl:unionOf`, `owl:someValuesFrom` if applied only to subclass-type expressions
  - Superclasses can be class names, `owl:allValuesFrom` or `owl:hasValue`; also max. cardinalities of 0 or 1 are allowed, all with superclass-type filler expressions only
- Property domains and ranges only for subclass-type expressions; property hierarchies, disjointness, inverses, (a)symmetry, transitivity, chains, (inverse)functionality, irreflexivity fully supported
- Disjoint classes and classes in keys need subclass-type expressions, equivalence only for expressions that are sub- and superclass-type, no restrictions on `owl:sameAs`
- Some restrictions on available datatypes
OWL 2 RL: Features

- Standard reasoning in OWL 2 RL: PTime-complete

- Rule-based reading simplifies modeling and implementation:
  even naïve implementations can be useful

- Fast and scalable implementations exist

Also: possibly useful for combining OWL with rules (see Lecture 5)
Do We Really Need So Many OWLs?

Three new OWL profiles with somewhat complex descriptions ... why not just one?

- The union of any two of the profiles is no longer light-weight! QL+RL, QL+EL, RL+EL all ExpTime-hard
- Restricting to fewer profiles = giving up potentially useful feature combinations
- Rationale: profiles are “maximal” (well, not quite) well-behaved fragments of OWL 2
  → Pick suitable feature set for applications
- In particular, nobody is forced to implement all of a profile
OWL in Practice: Tools

- Editors ([http://semanticweb.org/wiki/Editors](http://semanticweb.org/wiki/Editors))
  - Most common editor: Protégé 4
  - Other tools: TopBraid Composer ($), NeOn toolkit
  - Special purpose apps, esp. for light-weight ontologies (e.g. FOAF editors)
- Reasoners ([http://semanticweb.org/wiki/Reasoners](http://semanticweb.org/wiki/Reasoners))
  - OWL DL: Pellet, HermiT, FaCT++, RacerPro ($)
  - OWL EL: CEL, SHER, snorocket ($), ELLY (extension of IRIS)
  - OWL RL: OWLIM, Jena, Oracle Prime (part of O 11g) ($),
  - OWL QL: Owlgres, QuOnto, Quill
- Many tools use the **OWL API** library (Java)
- Note: many other Semantic Web tools are found online
There is more to do than editing and inferencing:

- **Explanation**: reasoning task of providing axiom sets to explain a conclusion (important for editing and debugging)
- **Conjunctive querying**: check entailment of complex query patterns (cf. Lecture 5)
- **Modularisation**: extract sub-ontologies that suffice for (dis)proving a certain conclusion
- **Repair**: determine ways to repair inconsistencies (related to explanation)
- **Least Common Subsumer**: assuming that class unions are not available, find the smallest class expression that subsumes two given classes
- **Abduction**: given an observed conclusion, derive possible input facts that would lead to this conclusion

→ All implemented, tasks on top common in standard tools today
## Overview: Essential OWL Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Related OWL vocabulary</th>
<th>FOL</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>top/bottom class</td>
<td>owl:Thing/owl:Nothing</td>
<td>(axiomatise)</td>
<td>⊤/⊥</td>
</tr>
<tr>
<td>Class intersection</td>
<td>owl:intersectionOf</td>
<td>∧</td>
<td>⊓</td>
</tr>
<tr>
<td>Class union</td>
<td>owl:unionOf</td>
<td>∨</td>
<td>⊔</td>
</tr>
<tr>
<td>Class complement</td>
<td>owl:complementOf</td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>Enumerated class</td>
<td>owl:oneOf</td>
<td>(ax. with ≈)</td>
<td>{a}</td>
</tr>
<tr>
<td><strong>Property restrictions</strong></td>
<td><strong>owl:onProperty</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Existential</td>
<td>owl:someValueFrom</td>
<td>∃y ...</td>
<td>∃R.C</td>
</tr>
<tr>
<td>Universal</td>
<td>owl:allValuesFrom</td>
<td>∀y ...</td>
<td>∀R.C</td>
</tr>
<tr>
<td>Min. cardinality</td>
<td>owl:minQualifiedCardinality; owl:onClass</td>
<td>∃y1...yn....</td>
<td>≥n S.C</td>
</tr>
<tr>
<td>Max. cardinality</td>
<td>owl:maxQualifiedCardinality; owl:onClass</td>
<td>∀y1...yn+1...→...</td>
<td>≤n S.C</td>
</tr>
<tr>
<td>Local reflexivity</td>
<td>owl:hasSelf</td>
<td>R(x,x)</td>
<td>∃R.Self</td>
</tr>
</tbody>
</table>
### Overview: Essential OWL Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Related OWL vocabulary</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property chain</td>
<td>owl:propertyChainAxiom</td>
<td>◦</td>
</tr>
<tr>
<td>Inverse</td>
<td>owl:inverseOf</td>
<td>$R^\sim$</td>
</tr>
<tr>
<td>Key</td>
<td>owl:hasKey</td>
<td>rule, see Lecture 5</td>
</tr>
<tr>
<td>Property disjointness</td>
<td>owl:propertyDisjointWith</td>
<td>Dis(R,S)</td>
</tr>
<tr>
<td>Property characteristics</td>
<td>rdf:hasType</td>
<td></td>
</tr>
<tr>
<td>Symmetric</td>
<td>owl:SymmetricProperty</td>
<td>Sym(R)</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>owl:AsymmetricProperty</td>
<td>Asy(R)</td>
</tr>
<tr>
<td>Reflexive</td>
<td>owl:ReflexiveProperty</td>
<td>Ref(R)</td>
</tr>
<tr>
<td>Irreflexive</td>
<td>owl:IrreflexiveProperty</td>
<td>Irr(R)</td>
</tr>
<tr>
<td>Transitivity</td>
<td>owl:TransitiveProperty</td>
<td>Tra(R)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subclass</th>
<th>rdfs:subClassOf</th>
<th>$\forall x. C(x) \rightarrow D(x)$</th>
<th>$C \sqsubseteq D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subproperty</td>
<td>rdfs:subPropertyOf</td>
<td>$\forall x,y. R(x,y) \rightarrow S(x,y)$</td>
<td>$R \sqsubseteq S$</td>
</tr>
</tbody>
</table>
Summary and Outlook

- OWL: expressive ontology language with practical impact
- Structurally representable in RDF (e.g. using Turtle syntax)
- Reasoning typical based on extensional (“direct”) semantics:
  - closely related to description logics and first-order logic (with equality)
  - different from RDF semantics, but compatible for many purposes
- Various flavours for different applications:
  - OWL Full provides RDF-based semantics (undecidable)
  - OWL DL decidable but complex (N2ExpTime)
  - OWL profiles for light-weight reasoning (in PTime)

Version 2 of the Web Ontology Language almost complete:

Official specification expected by Oct 2009
Further Reading

- P. Hitzler, S. Rudolph, M. Krötzsch: **Foundations of Semantic Web Technologies.** CRC Press, 2009. (Chapter 4 and 5 closely related to this lecture)


Selected research articles:


- F. Baader, S. Brandt, C. Lutz: **Pushing the EL envelope.** In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005), 2005. (paper introducing description logic EL++ underlying OWL EL)


- H. J. ter Horst: **Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary.** J. of Web Semantics 3(2–3):79–115, 2005. (rule-based implementation of parts of OWL Full, considerations that influenced the design of OWL RL)