In diesem Kapitel betrachten wir, wie Wissensrepräsentation aus dem World Wide Web ein Semantic Web machen kann.

Das Semantic Web ist eine Vision von Tim Berners-Lee, dem Erfinder des WWW.

Die Kernidee ist die explizite Repräsentation von Wissen im WWW, so dass es von Such- und anderen Maschinen verwendet werden kann.


Unter http://www.semanticwebprimer.org sind Folien, Beispiele etc. online zu finden. Auf diesem Foliensatz basieren auch die nachfolgenden Abschnitte.

The semantic web layer cake

Universal Resource Identifiers
e.g. http://www.somedomain.org/thispage#thistag

XML

- eXtensible Markup Language
- Web standard (W3C) for data exchange:
  - Description of in- and output data of applications
  - Reduces degrees of freedom for industrial data description standards
- Complementary to HTML:
  - HTML describes presentation
  - XML describes content
- Database perspective: XML as data model for semi-structured data.
XML syntax (1) – XML element

XML element:
- Description of an object, which is embraced by matching tags like `<author>` and `</author>`.
- Content of an element: Text and/or other (sub)elements.
- Elements can be nested
- Elements can be empty: `<year>`/`</year>` (short: `<year/>`)

Start tag → `<author>`

End tag → `</author>`

subelements
- `<firstname`> Bernhard `</firstname`>
- `<lastname`> Ganter `</lastname`>
- `<email`> ganter@tu-dresden.de `</email`>

Element author

email address may be wrong!

XML syntax (2) – XML attribute

XML attribute:
- Name-string pair
- Associated with an element
- Alternative way for describing data

```
<author email="ganter@tu-dresden.de">
  <firstname`> Bernhard `</firstname`
  <lastname`> Ganter `</lastname`
</author`
```

Alternative description of the same (?) data:

```
<author firstname="Bernhard" lastname="Ganter" email="ganter@tu-dresden.de"/>
```

The Tree Model of XML Documents: An Example

```
<email>
  <head>
    <from name="Michael Maher" address="michaelmaher@cs.gu.edu.au"/>
    <to name="Grigoris Antoniou" address="grigoris@cs.unibremen.de"/>
  </head>
  <subject>Where is your draft?</subject>
</email>
```

The Tree Model of XML Docs

The tree representation of an XML document is an ordered labeled tree:
- There is exactly one root
- There are no cycles
- Each non-root node has exactly one parent
- Each node has a label.
- The order of elements is important
- ... but the order of attributes is not important
Path Expressions in XPath

XPath is core for XML query languages. Language for addressing parts of an XML document.
- It operates on the tree data model of XML.
- It has a non-XML syntax.

Examples
- Address all books with title “Artificial Intelligence”
  \[\text{/book[@title="Artificial Intelligence"]}\]
- Address the first author element node in the XML document
  \[\text{//author[1]}\]
- Address the last book element within the first author element node in the document
  \[\text{//author[1]/book[last()]}\]
- Address all book element nodes without a title attribute
  \[\text{//book[not @title]}\]

XML Schema

Complex language for data description:
- Many standardised base types, e.g. float, double, decimal, boolean in particular: string and integer
- Types and typed references
- Class hierarchy and inheritance
- Consistency constraints

Standard („W3C Recommendation“) as extension to XML

Namespaces

An XML document may use more than one DTD or schema. Prefixes are used to avoid name clashes.
Prefixes have URIs as values. They usually point to a description of the namespace syntax.

Example:

\[
<\text{vu:instructors}\ xmlns:vu="http://www.vu.com/empDTD"
  xmlns:gu="http://www.gu.au/empDTD"
  xmlns:unik="http://www.unik.de/empDTD">
  \text{<unik:dozent}\ xmlns:unik="http://www.unik.de/empDTD"
    \text{unik:title="Dr."
    unik:name="Andreas Hotho"
    unik:department="Computer Science"/>}
  \text{<gu:academicStaff}\ xmlns:gu="http://www.gu.au/empDTD"
    \text{gu:title="lecturer"
    gu:name="Mate Jones"
    gu:school="Information Technology"/>}
</vu:instructors>
\]

Style Sheets

Move data and metadata from one XML representation to another, e.g., when applications that use different DTDs or schemas need to communicate.

The extensible stylesheet language XSL includes
- a transformation language (XSLT)
- a formatting language

XSLT specifies rules with which an input XML document is transformed to
- another XML document
- an HTML document
- plain text

The output document may use the same DTD or schema, or a completely different vocabulary. XSLT can be used independently of the formatting language.
The semantic web layer cake

RDF Recommendation consists of several parts
- RDF Primer (http://www.w3.org/TR/rdf-primer/)
- RDF Schema (http://www.w3.org/TR/rdf-schema/)
- ...

RDF - Resource Description Framework

- RDF provides metadata about web resources
- key component: Object -> Attribute -> Value triple
- Interconnected triples constitute a labelled graph
- RDF uses XML syntax

http://www.aifb.uni-karlsruhe.de/WBS/ysu

W3C

York

6086592

owner

tel

http://www.w3.org/RDF

.ServiceModel

[Image]

RDF Schema

- RDF defines vocabulary for RDF
- Vocabulary is organised as type hierarchy
  - Class, subClassOf
  - type
  - Property, subPropertyOf
  - domain, range

RDF Schema syntax in XML

```xml
<rdf:Description ID="Person">
  <rdf:type resource="http://www.w3.org/...#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/...#Resource"/>
</rdf:Description>

<rdf:Description ID="Teacher">
  <rdf:type resource="http://www.w3.org/...#Class"/>
  <rdfs:subClassOf rdf:resource="#Person"/>
</rdf:Description>

<rdf:Description ID="teaches">
  <rdf:type resource="http://www.w3.org/...#Property"/>
  <rdfs:domain rdf:resource="#Teacher"/>
  <rdfs:range rdf:resource="#Student"/>
</rdf:Description>

<rdf:Description ID="teaches well">
  <rdf:type resource="http://www.w3.org/...#Property"/>
  <rdfs:subPropertyOf rdf:resource="#teaches"/>
</rdf:Description>
```
RDFS - logical

Classes: unary predicates
subClassOf relation: implication

\[(\forall x) (\text{tutor}(x) \rightarrow \text{student}(x))\]

Properties: binary predicates
subPropertyOf relation: implication

\[(\forall x)(\forall y) (\text{supervises}(x,y) \rightarrow \text{responsibleFor}(x,y))\]

RDF statements are triples (Object, Property, Object)

- Objects can be
  - URIs
  - classes
  - properties
  - triples(!) -- (→ reification, i.e. second-order)

RDF Schema as ontology language

RDF(S) is useful for simple ontologies, but not for complex modelling
→ „Need for more expressivity!“

More expressive languages:
- OWL (based on description logics)
- F-Logic (based on logic programming)
- Hybrids and rules extensions for OWL

The Semantic Web layer cake

OWL – general info

- W3C Recommendation since 2004
- Semantic fragment of FOL (First-order predicate logic)
- Three variants: OWL Lite \( \subseteq \) OWL DL \( \subseteq \) OWL Full
- RDFS is fragment of OWL Full.
- No reification in OWL DL.
- OWL DL is decidable
- OWL DL = SHOIN(D)
Head of a document

Classes, roles and Individuals

Class relationships

- Boolean class constructors
- Role restrictions

Role properties

OWL documents

They consist of
- Head with general information
- Rest with the ontology

Head of an OWL document

Definition of namespaces in the root

```
<owl:Ontology rdf:about="">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SWRC Ontology December 2005</rdfs:comment>
  <owl:versionInfo>v0.5</owl:versionInfo>
  <owl:imports rdf:resource="http://www.semanticweb-grundlagen.de/foo"/>
</owl:Ontology>
```

General information

```
<owl:Ontology rdf:about="">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SWRC Ontology December 2005</rdfs:comment>
  <owl:versionInfo>v0.5</owl:versionInfo>
  <owl:imports rdf:resource="http://www.semanticweb-grundlagen.de/foo"/>
</owl:Ontology>
```
Classes, roles and individuals

Basic components of OWL ontologies:

Classes
- like resources in RDFS
- like classes in DL

Individuals
- like resources in RDFS
- like individuals in DL

Roles
- like attributes in RDFS
- like roles in DL

Definition

```
<owl:Class rdf:ID="Professor"/>
```

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

Individuals

Definition by class membership

```
<rdf:Description rdf:ID="RudiStuder">
  <rdf:type rdf:resource="#Professor"/>
</rdf:Description>
```

- equivalent:
  ```
  <Professor rdf:ID="RudiStuder"/>
  ```

Abstract Roles

Abstract roles defined like classes

```
<owl:ObjectProperty rdf:ID="Affiliation"/>
```

Domain and Range of abstract Roles

```
<owl:ObjectProperty rdf:ID="Affiliation">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Organisation"/>
</owl:ObjectProperty>
```

Domain: \(\exists\text{Affiliation} . T \sqsubseteq \text{Person}\)
Range: \(T \sqsubseteq \forall\text{Affiliation.Organisation}\)
Concrete Roles

Concrete roles have datatypes in range
<owl:DatatypeProperty rdf:ID="firstname"/>

Domain and range of concrete roles
<owl:DatatypeProperty rdf:ID="firstname">
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

One can use many XML Schema Datatypes. The standard requires at least integer and string.

Individuals and Roles

<Person rdf:ID="RudiStuder">
<Affiliation rdf:resource="#AIFB"/>
<Affiliation rdf:resource="#ontoprise"/>
<firstname rdf:datatype="&xsd;string">Rudi</firstname>
</Person>

Roles are in general not functional.

OWL – contents

Head of a document
Classes, roles and Individuals
Class relationships
- Boolean class constructors
- Role restrictions
Role properties

Simple class relationships

<owl:Class rdf:ID="Professor">
<rdfs:subClassOf rdf:resource="#Faculty"/>
</owl:Class>

<owl:Class rdf:ID="Faculty">
<rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>

It can be inferred that Professor is a subclass of Person.
Simple class relationships

<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf rdf:resource="#Faculty"/>
</owl:Class>
<owl:Class rdf:ID="Book">
  <rdfs:subClassOf rdf:resource="#Publication"/>
</owl:Class>
<owl:Class rdf:ID="#Faculty">
  <owl:disjointWith rdf:resource="#Publication"/>
</owl:Class>

We infer that **Professor** and **Book** are also disjoint classes.

<owl:Class rdf:about="#Publication">
  <owl:equivalentClass rdf:resource="#Publikation"/>
</owl:Class>

We infer that **Book** is a subclass of **Publikation**.

Individuals and class relationships

<owl:Class rdf:ID="Book">
  <rdfs:subClassOf rdf:resource="#Publication"/>
</owl:Class>
<owl:Class rdf:about="#Publication">
  <owl:equivalentClass rdf:resource="#Publikation"/>
</owl:Class>

We infer that **SemanticWebGrundlagen** is a **Publication**.

Relationships between individuals

<Professor rdf:ID="RudiStuder"/>
.rdf:Description rdf:about="#RudiStuder">
  <owl:sameAs rdf:resource="#ProfessorStuder"/>
</rdf:Description>

We infer that **ProfessorStuder** is a **Professor**.

Inequality of individuals expressed using **owl:differentFrom**.

Author(SemanticWebGrundlagen,YorkSure)
Author(SemanticWebGrundlagen,PascalHitzler)
Relationships between individuals

<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <Person rdf:about="#RudiStuder"/>
    <Person rdf:about="#YorkSure"/>
    <Person rdf:about="#PascalHitzler"/>
  </owl:distinctMembers>
</owl:AllDifferent>

Shortcut for multiple usage of owl:differentFrom.

Closed classes (nominals)

<owl:Class rdf:about="#SecretaryOfStuder"
  rdf:parseType="Collection">
  <Person rdf:about="#GiselaSchillinger"/>
  <Person rdf:about="#AnneEberhardt"/>
</owl:oneOf>

There are exactly those two Individuals in the class SecretaryOfStuder.

SecretaryOfStuder ≡ \{GiselaSchillinger,AnneEberhardt\}

OWL - contents

Head of a document
Classes, roles and Individuals
Class relationships
  ■ Boolean class constructors
  ■ Role restrictions
Role properties

Boolean class constructors

Conjunction:
  owl:intersectionOf
Disjunction:
  owl:unionOf
Negation:
  owl:complementOf

Can be used to construct complex classes from class names.
**Conjunction**

\[
\text{SecretaryOfStuder} \equiv \text{Secretary} \cap \text{MemberAGStuder}
\]

We infer that all individuals in SecretaryOfStuder are also in Secretary.

**Disjunction**

\[
\text{Professor} \subseteq \text{activeTeacher} \cup \text{retired}
\]

**Negation**

\[
\text{Faculty} \cap \text{Publication} \equiv \bot
\]

**OWL – contents**

- Head of a document
- Classes, roles and Individuals
- Class relationships
  - Complex class definitions
- Boolean class constructors
- Role restrictions
- Role properties
Using roles for defining complex classes

\[
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}}
\]

Role restrictions (allValuesFrom)

\[
\begin{align*}
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}}
\end{align*}
\]

I.e. all examiners of an exam must be professors.

Role restrictions (someValuesFrom)

\[
\begin{align*}
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Person}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Person}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Person}}
\end{align*}
\]

I.e. each exam must have at least one examiner.

Number restrictions (cardinalities)

\[
\begin{align*}
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}}
\end{align*}
\]

An exam must have at most two examiners.

\[
\begin{align*}
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}} & \\
\text{Exam} \sqsubseteq \text{\texttt{hasExaminer.Professor}}
\end{align*}
\]

An exam must cover at least three topics.
Number restrictions (cardinalities)

An exam must cover exactly three topics.

Role restrictions (hasValue)

owl:hasValue always points to an individual. This is equivalent to the example on the next slide.
Role relationships

hasExaminer ⊆ hasParticipant

Similar: owl:equivalentProperty

Roles can be inverse to each other:

hasExaminer ⇔ examinerOf

Domain

Range

Transitivity i.e. (a,b) and r(b,c) implies r(a,c)

Symmetry i.e. r(a,b) implies r(b,a)

Functionality i.e. r(a,b) and r(a,c) implies b = c

Inverse functionality

Domain and Range

Is equivalent to the following:

Domain and Range

<owl:ObjectProperty rdf:ID="\&owl;Thing">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Affiliation"/>
      <owl:allValuesFrom rdf:resource="#Organisation"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

It follows that PrimeNumber is an Organisation!
Role properties

<owl:ObjectProperty rdf:ID="hasColleague">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProjectLeader">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isProjectLeaderFor">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
</owl:ObjectProperty>

<Person rdf:ID="YorkSure">
  <hasColleague rdf:resource="#PascalHitzler"/>
  <hasColleague rdf:resource="#AnupriyaAnkolekar"/>
  <isProjectLeaderFor rdf:resource="#SEKT"/>
</Person>

<Projekt rdf:ID="SmartWeb">
  <hasProjectLeader rdf:resource="#PascalHitzler"/>
  <hasProjectLeader rdf:resource="#HitzlerPascal"/>
</Projekt>

Logical consequences from example

AnupriyaAnkolekar hasColleague YorkSure
AnupriyaAnkolekar hasColleague PascalHitzler
PascalHitzler owl:sameAs HitzlerPascal

The Semantic Web layer cake

OWL variants

OWL Full
- Contains OWL DL and OWL Lite
- Contains all of RDFS
- Undecidable
- Limited support by existing software

OWL DL (= SHOIN(D))
- Contains OWL Lite and is contained in OWL Full
- decidable
- Tools available
- Complexity NExpTime (worst-case)

OWL Lite (= SHIF(D))
- Is contained in OWL DL and OWL Full
- decidable
- Less expressive
- Complexity ExpTime (worst-case)
OWL Full

Unlimited usage of all OWL and RDFS constructs (must be valid RDFS).

Difficult is e.g. the non-existent type separation (Classes, Roles, Individuals), hence:

- owl:Thing is the same as rdfs:resource
- owl:Class is the same as rdfs:Class
- owl:DatatypeProperty subclass of owl:ObjectProperty
- owl:ObjectProperty is the same as rdf:Property

Why types are not seperated in OWL Full

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

One often does not really need inferencing over such information.

OWL DL

- is SHOIN(D).
- Allowed are only certain RDFS constructs (like those in the examples. Not allowed: rdfs:Class, rdfs:Property
- Type separation. Classes and Roles must be declared explicitly.
- Concrete Roles must not be transitive, symmetric, inverse or inverse functional.
- Number restrictions must not be used with transitive roles, their subroles, or their inverses.

OWL Lite

- is SHIF(D).
- All restrictions for OWL DL apply.
- Not allowed: oneOf, unionOf, complementOf, hasValue, disjointWith
- Number restrictions only allowed with 0 and 1.
- Some restrictions on the occurrence of anonymous (complex) classes apply, e.g. they must not occur in the subject of rdfs:subClassOf.
OWL Tools

Editors
- Protegé, http://protege.stanford.edu

Inference engines
- KAON2, http://kaon2.semanticweb.org
- FACT++, http://owl.man.ac.uk/factplusplus/

Appendix: OWL language components

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

Relations between individuals
owl:sameAs
owl:differentFrom
owl:AllDifferent
(together with
owl:distinctMembers)

Required datatypes
xsd:strong
xsd:integer

Class constructors and relationships
owl:Class
owl:Thing
owl:Nothing
rdfs:subClassOf
owl:disjointWith
owl:equivalentClass
owl:intersectionOf
owl:unionOf
owl:complementOf

Appendix: OWL language components

Role restrictions
owl:allValuesFrom
owl:someValuesFrom
owl:hasValue
owl:cardinality
owl:minCardinality
owl:maxCardinality
owl:oneOf

Role constructors, relations and properties
owl:ObjectProperty
owl:DatatypeProperty
rdfs:subPropertyOf
owl:equivalentProperty
owl:InverseOf
rdfs:domain
rdfs:range
rdf:resource="&owl;TransitiveProperty"
rdf:resource="&owl;SymmetricProperty"
rdf:resource="&owl;FunctionalProperty"
rdf:resource="&owl;InverseFunctionalProperty"
OWL as DL: Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\cap$ Male, Doctor $\cap$ Lawyer</td>
<td>$C_3(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>$\neg$Male</td>
<td>$C_2(x) \lor \ldots \lor C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$C$</td>
<td>$\forall P.C$</td>
<td>$\forall x. P(x, y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \cup \ldots \cup {x_n}$</td>
<td>$\forall P.C$</td>
<td>$\forall y. P(x, y) \land C(y)$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists P.C$</td>
<td>$\exists \forall y. P(x, y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists P.C$</td>
<td>$\exists \exists y. P(x, y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1$hasChild</td>
<td>$\exists \leq y. P(x, y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2$hasChild</td>
<td>$\exists \geq y. P(x, y)$</td>
</tr>
</tbody>
</table>

XMLS datatypes as well as classes in $\forall P.C$ and $\exists P.C$

- E.g., $\exists$hasAge.$\_$nonNegativeInteger
- Arbitrarily complex nesting of constructors
  - E.g., Person $\cap \forall$hasChild.Doctor $\cup \exists$hasChild.Doctor

RDFS Syntax

E.g., Person $\cap \forall$hasChild.Doctor $\cup \exists$hasChild.Doctor:

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:Class rdf:about="#Doctor"/>
    </owl:Restriction>
    <owl:unionOf rdf:parseType="collection">
      <owl:Class rdf:about="#hasChild"/>
      <owl:hasClass rdf:resource="#Doctor"/>
    </owl:unionOf>
  </owl:intersectionOf>
</owl:Class>
```

OWL as DL: Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\cap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\cap$ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \perp C_2$</td>
<td>Male $\sqsubseteq$ $\neg$Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>${x_1} \equiv {x_2}$</td>
<td>{President.Bush} $\equiv$ {G.W.Bush}</td>
</tr>
<tr>
<td>differentFrom</td>
<td>${x_1} \sqsubseteq {x_2}$</td>
<td>{John} $\sqsubseteq$ {Peter}</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \rightarrow P_2$</td>
<td>hasChild $= \neg$ hasParent</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P \sqsubseteq P_1$</td>
<td>ancestor $\sqsubseteq$ ancestor</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>$\top \sqsubseteq P_1$</td>
<td>$\top \sqsubseteq$ hasMother</td>
</tr>
<tr>
<td>InverseFunctionalProperty</td>
<td>$\top \sqsubseteq P_1$</td>
<td>$\top \sqsubseteq$ hasSSN</td>
</tr>
</tbody>
</table>

Axioms (mostly) reducible to inclusion ($\sqsubseteq$)

- $C \sqsubseteq D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$

Obvious FOL equivalences

- E.g., $C \equiv D$ iff $\forall x. (C(x) \leftrightarrow D(x))$
- $C \subseteq D$ iff $\forall x. (C(x) \Rightarrow D(x))$