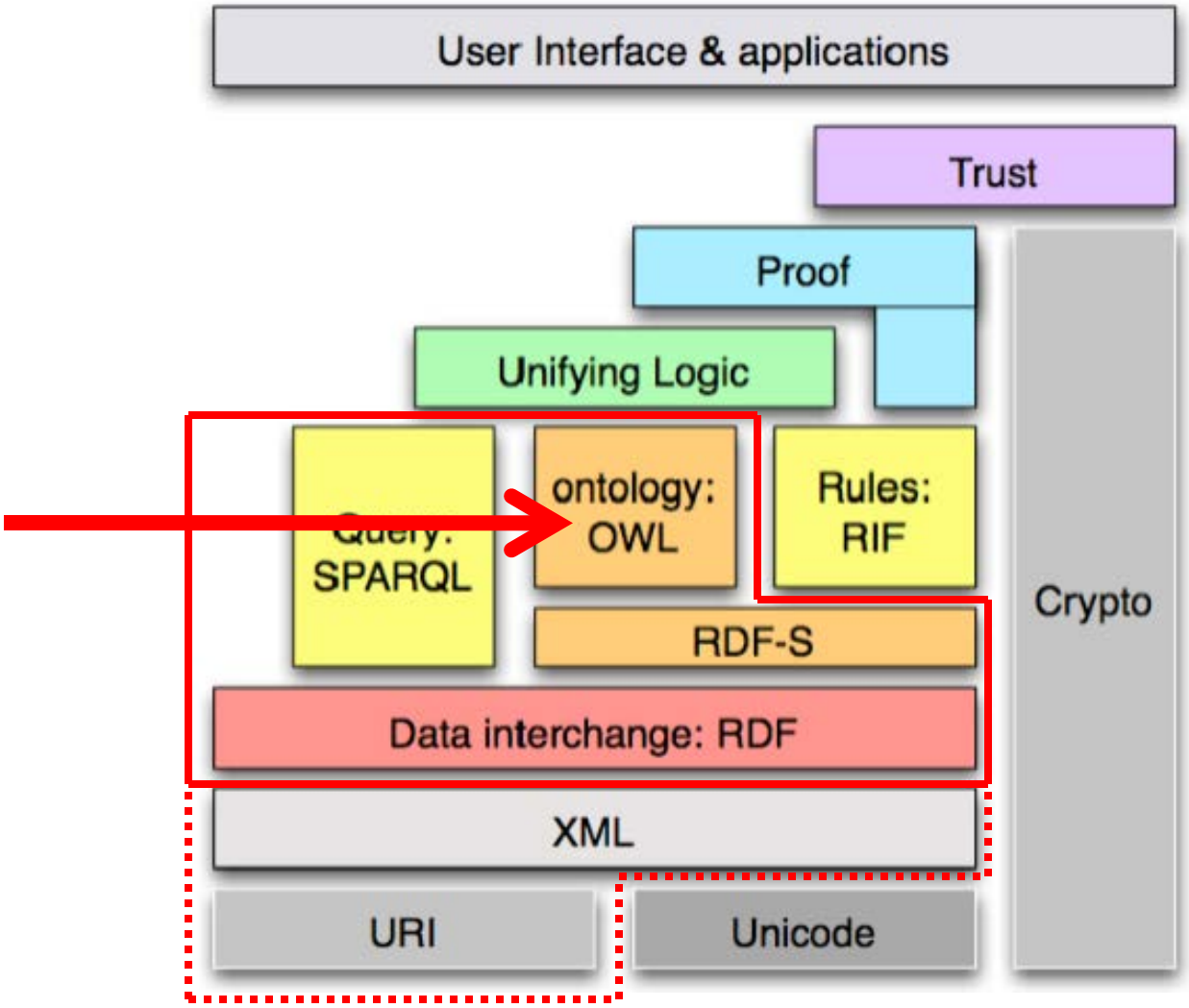
OWL

Some slides (derived) from Pascal Hirtzler/Sebastian Rudolph/Donald Kossmann

Today: OWL Syntax



Ontologies

- *„A specification of a conceptualization“*
- **Ontologies describe a domain:**
  - **Declarative and explicit**
  - **Give all concepts (classes)**
  - **Give all properties (attributes, relationships)**
  - **Constraints on properties**
  - **Give individuals**

Benefits of Ontologies

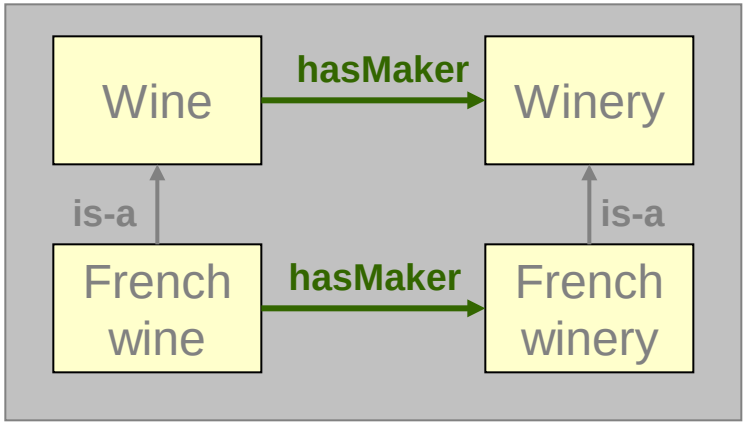
- Shared vocabulary (for humans and agents)
- Shared common understanding of the structure of information
- Reuse of domain knowledge
  - Avoid re-inventing the wheel
  - Establish standards

Benefits of Ontologies (ctd)

- **Assumptions become explicit**
  - **Explain assumptions**
  - **Change assumptions**
  - **Hypothetical reasoning (multiple scenarios)**
  - **Support for evolving systems where time and situations change necessitating re-evaluation of assumptions**
  - **Support for interoperation with other (legacy) systems**
- **Separation of types of knowledge**
  - **Declarative domain knowledge vs. procedural knowledge**
  - **Background knowledge (unchanging meta-data) from changing information**
  - **Authoritative vs. other…**

Example: subClasses

- **Subclass inherits all slots from superclass**
- **Subclass can add constraints to “narrow” the list of allowed values**
  - **Make the cardinality range smaller**
  - **Replace a class in the range with a subclass**



Example: Synonyms

- Synonym names for the same concept are not different classes
- Many systems allow listing synonyms as part of the class definition
- OWL allows defining necessary and sufficient conditional definitions thereby allowing synonym definitions to be “first class” terms

Use Case: Web Portals

- Describe terminology for a community
- Enable powerful search
- Enable uniformed publishing of data
- Examples:
  - OntoWeb: www.ontoweb.org
  - The Open Directory: www.dmoz.org

Use Case: Multimedia Data

- Specify meta-data for images, music, etc.
- Specify constraints on that data
  - E.g., Late Gregorian -> Britain, 1760-1811
- Makes search more powerful (inference)
  - Look for „Late Gregorian“ if interest in Britain

- Further Use Cases:
  - Ad-hoc networks: e.g., Jini, UPnP
  - Design documentation for engineering

Requirements for Ontologies

- Instantiation of classes by „Individuals“
- Hierarchies of concepts (taxonomies, „inheritance“):
- Classes, concepts
- Binary relationships among individuals: Properties, Roles
- Properties of relationships (e.g. range, transitive)
- Data types (e.g. numbers): concrete domains

RDF Schema as Ontology Language

- **Useful for simple Ontologies**
- **Pro: reasoning/entailment fairly efficient**
- **Con: not suitable for complex modeling**
- **Need for more languages: OWL**

Species of OWL

- **OWL Lite: simple constraints and classification hierarchy**
- **OWL DL: maximum expressiveness without losing computation completeness**
- **OWL Full: maximum expressiveness and full RDF syntactic freedom, but no computation guarantees (might specify undeciable rules)**
- **N.B. RDFS subset of OWL Full!**

OWL Documents

- **Are “normal“ RDF documents consisting of**
  - **Header**
  - **Actual ontology**

OWL Header (1)

Define all namespaces at root node

```
<rdf:RDF
xmlns ="http://www.semanticweb-
grundlagen.de/beispielontologie#"
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>
```

OWL Header (2)

General Information

```
<owl:Ontology rdf:about="">
<rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string“>
 SWRC Ontologie in der Version vom Dezember 2005
 </rdfs:comment>
 <owl:versionInfo>v0.5</owl:versionInfo>
 <owl:imports rdf:resource="http://www.semanticweb-
grundlagen.de/foo"/>
 <owl:priorVersion
rdf:resource="http://ontoware.org/projects/swrc"/>
 </owl:Ontology>
```

OWL Header - properties

- **Inherited from RDFS**
  - rdfs:comment
  - rdfs:label
  - rdfs:seeAlso
  - rdfs:isDefinedBy
- **Versioning**
  - owl:versionInfo
  - owl:priorVersion
  - owl:backwardCompatibleWith
  - owl:incompatibleWith
  - owl:DeprecatedClass
  - owl:DeprecatedProperty
- **Others**
  - owl:imports

Classes, Roles and Indiviuals

The three building blocks of ontologies:

1. **Classes**
   Similar to classes in RDFS
2. **Individuals**
   Similar to objects in RDFS
3. **Roles**
   Similar to properties in RDFS

Classes

Definition

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

predefined

- owl:Thing
- owl:Nothing

Individuals

Definition by membership in a class

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

Same as

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

Abstract Roles

Defined the same way as classes

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

Domain and Range of abstract roles

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

Concrete Roles

Concrete roles have datatypes as range

```
<owl:DatatypeProperty rdf:ID="Vorname"/>
```

Domain and Range of concrete roles

```
<owl:DatatypeProperty   rdf:ID="Vorname">
   <rdfs:domain rdf:resource="#Person"    />
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Many XML Schema data types possible, `integer` and `string` mandatory

Individuals and Roles

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

Roles are typically not functional (see later!)

Simple relationships among classes

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

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

We can now infer that `Professor` is a subclass of `Person`

Simple relationships among classes

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

<owl:Class  rdf:ID=“Buch">
      <rdfs:subClassOf  rdf:resource="#Publication"/>
</owl:Class>

<owl:Class rdf:about="#Fakultaetsmitglied">
      <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

We can now infer that `Professor` and `Book` are also disjoint

Simple relationships among classes

```
<owl:Class  rdf:ID="Buch">
      <rdfs:subClassOf  rdf:resource="#Publikation"/>
</owl:Class>

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

We can now infer that `Buch` is a subclass of `Publication`

Individuals and Class Relationships

```
<Buch        rdf:ID="SemanticWebGrundlagen">
<Autor       rdf:resource="#PascalHitzler"/>
<Autor       rdf:resource="#MarkusKrötzsch"/>
<Autor       rdf:resource="#SebastianRudolph"/>
<Autor       rdf:resource="#YorkSure"/>
</Buch>

<owl:Class  rdf:about="#Buch">
<rdfs:subClassOf  rdf:resource="#Publikation"/>
</owl:Class>
```

We can now infer that `SemantikWebGrundlagen` **is of type** `Publication`

Relationships among individuals

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

We can now infer that `ProfessorStuder` **is a** `Professor`

Different of individuals via

`owl:differentFrom`

Relationships among individuals

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

- **Shorthand for multiple owl:differentFrom**

Enumerations

```
<owl:Class  rdf:about="#SekretaerinnenVonStuder">
<owl:oneOf  rdf:parseType="Collection">
      <Person rdf:about="#GiselaSchillinger"/>
      <Person rdf:about="#BeateKuehner"/>
</owl:oneOf>
</owl:Class>
```

- **There are only these two** `SekretaerinnenVonStuder`

Boolean combination of class expressions

- **logical And (conjunction):**
  `owl:intersectionOf`
- **logical Or (disjunction):**
  `owl:unionOf`
- **logical Not (negation):**
  `owl:complementOf`
- **Create complex classes from simple classes**

Conjunction

```
<owl:Class  rdf:about="#SekretaerinnenVonStuder">
<owl:equivalentClass>
<owl:Class>
<owl:intersectionOf     rdf:parseType="Collection">
<owl:Class  rdf:about="#Sekretaerinnen"/>
<owl:Class  rdf:about="#AngehoerigeAGStuder"/>
</owl:intersectionOf>
<owl:Class>
</owl:equivalentClass>
</owl:Class>
```

We can now infer that `SekretaerinnenVonStuder` are also `Sekretaerinnen`

Disjunction

```
<owl:Class  rdf:about="#Professor">
 <rdfs:subClassOf>
 <owl:Class>
  <owl:unionOf    rdf:parseType="Collection">
 <owl:Class       rdf:about="#aktivLehrend"/>
 <owl:Class       rdf:about="#imRuhestand"/>
 </owl:unionOf>
 </owl:Class>
 </rdfs:subClassOf>
</owl:Class>
```

Negation

```
<owl:Class  rdf:about="#Fakultaetsmitglied">
 <rdfs:subClassOf>
  <owl:Class>
  <owl:complementOf     rdf:resource="#Publikation"/>
</owl:Class>
</rdfs:subClassOf>
</owl:Class>
```

Semantically equivalent to

```
<owl:Class rdf:about="#Fakultaetsmitglied">
   <owl:disjointWith rdf:resource="#Publikation"/>
</owl:Class>
```

Restrictions on Roles

Define complex classes by restricting roles

```
<owl:Class  rdf:ID="Pruefung">
<rdfs:subClassOf>
      <owl:Restriction>
      <owl:onProperty   rdf:resource="#hatPruefer"/>
      <owl:allValuesFrom rdf:resource="#Professor"/>
      </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

i.e. all examiners need to be Professor

Restrictions on Roles

```
<owl:Class  rdf:about="#Pruefung">
<rdfs:subClassOf>
      <owl:Restriction>
      <owl:onProperty   rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom rdf:resource="#Person"/>
      </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

i.e. there needs to be at *least one* examiner

Restrictions on Roles: Cardinalities

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

There can be *at most* 2 examiners

Restrictions on Roles: Cardinalities

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

There need to be *at least* three topics in an exam

Restrictions on Roles: Cardinalities

```
<owl:Class  rdf:about="#Pruefung">
<rdfs:subClassOf>
      <owl:Restriction>
      <owl:onProperty   rdf:resource="#hatThema"/>
      <owl:cardinality
            rdf:datatype="&xsd;nonNegativeInteger">
            3
      </owl:minCardinality>
      </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

There need to be *exactly* three topics in an exam

Restrictions on Roles: Instance References

```
<owl:Class  rdf:ID="PruefungBeiStuder">
<rdfs:equivalentClass>
      <owl:Restriction>
      <owl:onProperty   rdf:resource="#hatPruefer"/>
      <owl:hasValue rdf:resource="#RudiStuder"/>
      </owl:Restriction>
</rdfs:equivalentClass>
</owl:Class>
```

owl:hasValue always refers to concrete instances

Restrictions on Roles: Instance References

```
<owl:Class  rdf:ID="PruefungBeiStuder">
<owl:equivalentClass>
<owl:Restriction>
      <owl:onProperty   rdf:resource="#hatPruefer"/>
      <owl:someValuesFrom>
      <owl:oneOf  rdf:parseType="Collection">
      <owl:Thing  rdf:about="#RudiStuder"/>
      </owl:oneOf>
      </owl:someValuesFrom
</owl:Restriction>
</owl:equivalentClass>
</owl:Class>
```

Alternative solution

Relationships among Roles

```
<owl:ObjectProperty rdf:ID="hatPruefer">
      <rdfs:subPropertyOf
        rdf:resource="#hatAnwesenden"/>
</owl:ObjectProperty>
```

Similarly: `owl:equivalentProperty`

Roles can also express the *inverse* of other roles:

```
<owl:ObjectProperty rdf:ID="hatPruefer">
      <owl:inverseOf rdf:resource="#prueferVon"/>
</owl:ObjectProperty>
```

Role Properties

- **Domain**
- **Range**
- **Transitivity, i.e.**
  r(a,b) and r(b,c) implies r(a,c)
- **Symmetry, i.e.**
  r(a,b) implies r(b,a)

- **Functionality**
  r(a,b) and r(a,c) implies b=c

- **Inverted Functionality**
  r(a,b) und r(c,b) implies a=c

Domain and Range

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

equivalent to

```
<owl:Class  rdf:about="\&owl;Thing">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty   rdf:resource="#Zugehoerigkeit"/>
<owl:allValuesFrom
      rdf:resource="#Organisation"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Caveats with Domain and Range

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


<Zahl rdf:ID="Fuenf">
<Zugehoerigkeit rdf:resource="#Primzahlen"/>
</Zahl>
```

`Primzahlen` is now an `Organisation`!

Example on Role Properties

```
<owl:ObjectProperty rdf:ID="hatKollegen">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hatProjektleiter">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="istProjektleiterFuer">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
</owl:ObjectProperty>
<Person rdf:ID="YorkSure">
  <hatKollegen rdf:resource="#PascalHitzler"/>
  <hatKollegen rdf:resource="#AnupriyaAnkolekar"/>
  <istProjektleiterFuer rdf:resource="#SEKT"/>
</Person>
<Projekt rdf:ID="SmartWeb">
  <hatProjektleiter rdf:resource="#PascalHitzler"/>
  <hatProjektleiter rdf:resource="#HitzlerPascal"/>
</Projekt>
```

Derivations from the example

- `AnupriyaAnkolekar hatKollegen YorkSure`
- `AnupriyaAnkolekar hatKollegen PascalHitzler`
- `PascalHitzler owl:sameAs HitzlerPascal`

Species of OWL (revisited)

- OWL Full
  - Contains OWL DL und OWL Lite
  - Contains RDFS (not true for the others)
  - Undecidable
  - Very limited tool support
- OWL DL (Description Logic)
  - Contains OWL Lite, part of OWL Full.
  - Decidable.
  - Almost complete tool support
  - Complexity NExpTime (worst-case)
- OWL Lite
  - Part of OWL DL and OWL Full
  - Decidable.
  - Limited expressiveness
  - Complexity ExpTime (worst-case)

OWL Lite Features (1)

- **RDF Schema Features**
  - **Class, rdfs:subClassOf , Individual**
  - **rdf:Property, rdfs:subPropertyOf**
  - **rdfs:domain , rdfs:range**
  - **data types**
- **Equality and Inequality and Intersection**
  - **sameClassAs , samePropertyAs , sameIndividualAs**
  - **differentIndividualFrom, setOfMutuallyDifferentIndividuals**
  - **intersectionOf**

OWL Lite Features (2)

- **Restricted Cardinality**
  - **minCardinality, maxCardinality (restricted to 0 or 1)**
  - **cardinality (restricted to 0 or 1)**
- **Property Characteristics**
  - **inverseOf , TransitiveProperty , SymmetricProperty**
  - **FunctionalProperty (unique) , InverseFunctionalProperty (injective)**
  - **allValuesFrom, someValuesFrom (universal and existential local range restrictions)**
- **Header Information and Others**
  - **imports , Dublin Core Metadata , versionInfo, annotations**

OWL (non Lite) Features

- Class Axioms
  - oneOf (enumerated classes)
  - disjointWith (two classes are disjoint)
- Boolean Combinations of Class Expressions
  - unionOf, intersectionOf, complementOf
- Arbitrary Cardinality (any values allowed)
  - minCardinality,maxCardinality,cardinality
- Filler Information
  - hasValue (property value): Dutch have “Netherlands” as value of their “nation” Property

Essential OWL Features

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	`owl:Thing/owl:Nothing`	(axiomatise)	$\top/\bot$
Class intersection	`owl:intersectionOf`	$\wedge$	$\sqcap$
Class union	`owl:unionOf`	$\vee$	$\sqcup$
Class complement	`owl:complementOf`	$\neg$	$\neg$
Enumerated class	`owl:oneOf`	(ax. with $\approx$)	{a}
Property restrictions	`owl:onProperty`		
Existential	`owl:someValueFrom`	$\exists y \ldots$	$\exists R.C$
Universal	`owl:allValuesFrom`	$\forall y \ldots$	$\forall R.C$
Min. cardinality	`owl:minQualifiedCardinality` `owl:onClass`	$\exists y1 \ldots yn. \ldots$	$\geq n\ S.C$
Max. cardinality	`owl:maxQualifiedCardinality` `owl:onClass`	$\forall y1 \ldots yn+1. \ldots \rightarrow \ldots$	$\leq n\ S.C$
Local reflexivity	`owl:hasSelf`	$R(x,x)$	$\exists R.Self$

Essential OWL Features

Feature	Related OWL vocabulary	DL
Property chain	`owl:propertyChainAxiom`	$\circ$
Inverse	`owl:inverseOf`	R^-
Key	`owl:hasKey`	
Property disjointness	`owl:propertyDisjointWith`	Dis(R,S)
Property characteristics	**rdf:type**	
Symmetric	`owl:SymmetricProperty`	Sym(R)
Asymmetric	`owl:AsymmetricProperty`	Asy(R)
Reflexive	`owl:ReflexiveProperty`	Ref(R)
Irreflexive	`owl:IrreflexiveProperty`	Irr(R)
Transitivity	`owl:TransitiveProperty`	Tra(R)

Subclass	`rdfs:subClassOf`	$\forall x.C(x) \rightarrow D(x)$	$C \sqsubseteq D$
Subproperty	`rdfs:subPropertyOf`	$\forall x,y.R(x,y) \rightarrow S(x,y)$	$R \sqsubseteq S$

OWL Tools

- **Editors**
  - **Protegé, http://protege.stanford.edu**
  - **SWOOP, http://www.mindswap.org/2004/SWOOP/**
  - **OWL Tools, http://owltools.ontoware.org/**

- **Reasoners**
  - **Pellet, http://www.mindswap.org/2003/pellet/index.shtml**
  - **KAON2, http://kaon2.semanticweb.org**
  - **FACT++, http://owl.man.ac.uk/factplusplus/**
  - **Racer, http://www.racer-systems.com/**
  - **Cerebra, http://www.cerebra.com/index.html**