

Semantic Web - OWL

CS 431 – April 2, 2008

Carl Lagoze – Cornell University

Acknowledgements for various slides and ideas

- Ian Horrocks (Manchester U.K.)
- Eric Miller (W3C)
- Dieter Fensel (Berlin)
- Volker Haarslev (Montreal)

RDF meta-model basic elements

- All defined in rdf namespace
 - <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- Types (or classes)
 - **rdf:Resource** – everything that can be identified (with a URI)
 - **rdf:Property** – specialization of a resource expressing a binary relation between two resources
 - **rdf:statement** – a triple with properties **rdf:subject**, **rdf:predicate**, **rdf:object**
- Properties
 - **rdf:type** - subject is an *instance* of that category or class defined by the value
 - **rdf:subject**, **rdf:predicate**, **rdf:object** – relate elements of statement tuple to a resource of type statement.

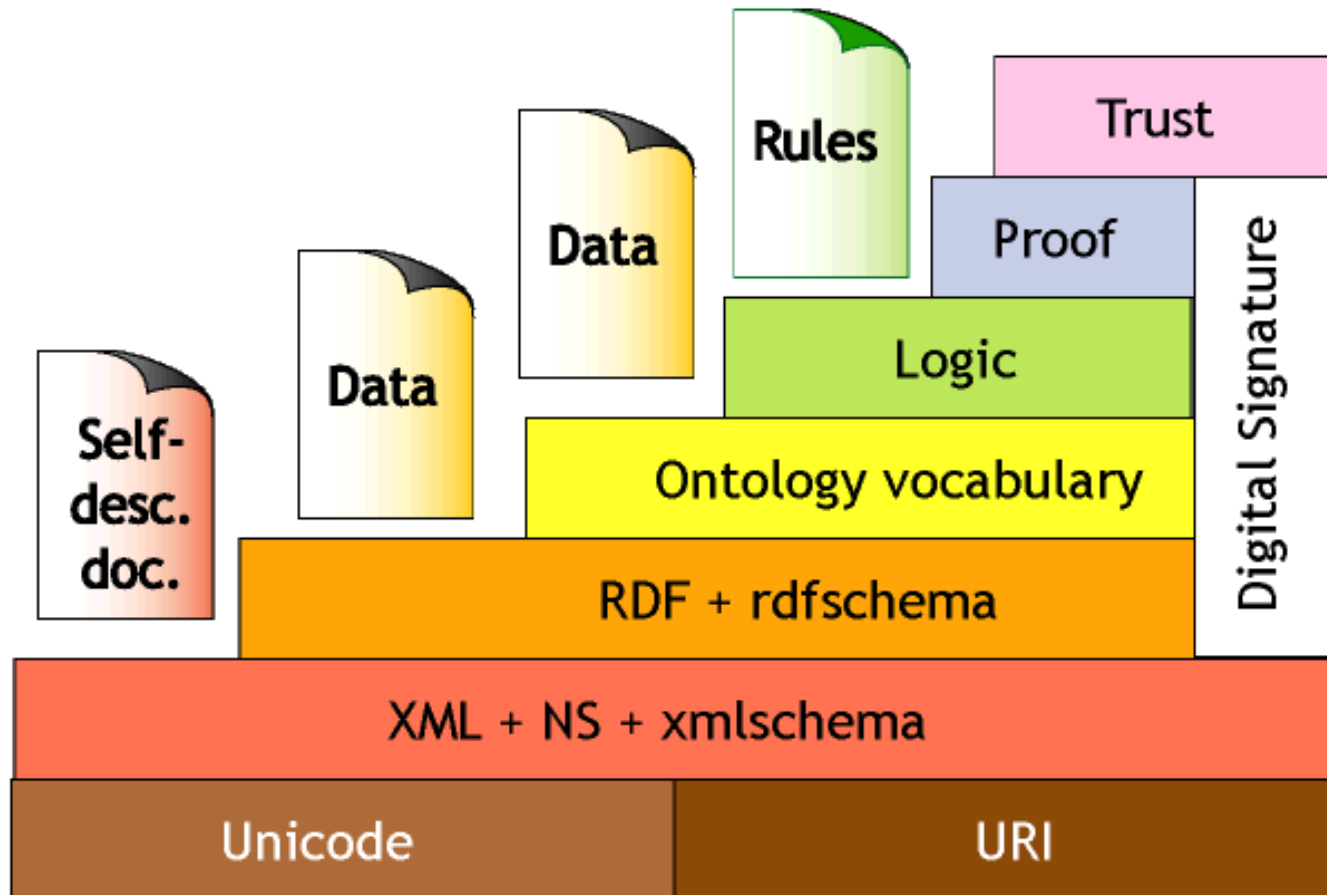
RDFs Namespace

- Class-related
 - rdfs:Class, rdfs:subClassOf
- Property-related
 - rdfs:subPropertyOf, rdfs:domain, rdfs:range

Sub-Class Inferencing

Explicit Model	Inferences
(:s rdf:type :o)	(:o rdf:type rdfs:Class)
(:s rdf:type :o) (:o rdfs:subClassOf :c)	(:s rdf:type :c)
(:s rdfs:subClassOf :o) (:o rdfs:subClassOf :c)	(:s rdfs:subClassOf :c)
(:s rdfs:subClassOf :o)	(:s rdf:type rdfs:Class) (:o rdf:type rdfs:Class)
(:s rdf:type rdfs:Class)	(:s rdfs:subClassOf rdf:Resource)

Components of the Semantic Web



Problems with RDF/RDFs

Non-standard, overly “liberal” semantics

- No distinction between class and instances
 - <Species, type, Class>
 - <Lion, type, Species>
 - <Leo, type, Lion>
- Properties themselves can have properties
 - <hasDaughter, subPropertyOf, hasChild>
 - <hasDaugnter, type, Property>
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
 - <type, range, Class>
 - <Property, type, Class>
 - <type, subPropertyOf, subclassOf>
- No known reasoners for these non-standard semantics

Problems with RDF/RDFs

Weaknesses in expressivity

- No localized domain and range constraints
 - Can't say the range of hasChild is person in context of persons and elephants in context of elephants
- No existence/cardinality constraints
 - Can't say that all instances of persons have a mother that is also a person
 - Can't say that persons have exactly two biological parents
- No transitive, inverse or symmetric properties
 - Can't say isPartOf is a transitive property
 - Can't say isPartOf is inverse of hasPart
 - Can't say touches is symmetric

So, we need a more expressive
and well-grounded ontology
language....

Web Ontology Language (OWL)

- W3C Web Ontology Working Group (WebOnt)
- Follow on to DAML, OIL efforts
- W3C Recommendation
- Vocabulary extension of RDF

Species of OWL

- *OWL Lite*
 - Good for classification hierarchies with simple constraints (e.g., thesauri)
 - Reasoning is computational simple and efficient
- *OWL DL*
 - Computationally complete and decidable (computation in finite time)
 - Correspondence to *description logics* (decidable fragment of first-order logic)
- *OWL Full*
 - Maximum expressiveness
 - No computational guarantees (probably never will be)
- Each language is extension of simpler predecessor

Relationship between OWL and RDF(s)

- OWL Full is extension of RDF
- OWL Lite and DL are extensions of a restricted view of RDF
- Every OWL document is an RDF document
- Every RDF document is an OWL Full document
- Only some RDF documents are OWL Lite or OWL DC
- Constraining an RDF document to be OWL Lite or DL
 - Every individual must have class membership (at least owl:thing)
 - URIs for classes, properties, and individuals must be mutually disjoint.

The “DL” in Owl DL

- Description Logics
- Goal: want to be able to reason (infer information) about a knowledge base
- Remember: a knowledge base consists of both meta (schema) information and instance (individual) information
- Remember: we want to do this based on an open world assumption
- OWL (Lite/DL) is then an RDF expression of DL

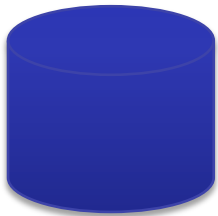
Closed World

John has a sister Betty

Betty has a sister Mary

Does John have a brother?

No!



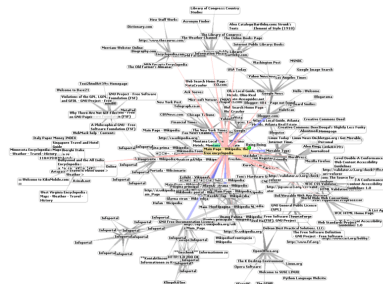
Open World

John has a sister Betty

Betty has a sister Mary

Does John have a brother?

??



Description Logics

- Highly expressible fragment of FOL with:
 - **Decidability**: guaranteed that computation can be done in finite amount of time
 - **Completeness**: every question within the logical system can be answered, or there are no paradoxes
- Designed for logical representation of object-oriented formalisms
 - frames/classes/concepts
 - sets of objects
 - roles/properties
 - binary relations on objects
 - individuals
- Represented as a collection of statements, with unary and binary predicates that stand for concepts and roles, from which deductions can be made

Description Logics Primitives

- Atomic Concept
 - Human
- Atomic Role
 - likes
- Conjunction
 - human *intersection* male
- Disjunction
 - nice *union* rich
- Negation
 - *not* rich
- Existential Class Restriction
 - *exists* enrolledIn.CSclass
- Universal Class Restriction
 - *all.enrolledIn.CSclass*
- Cardinality Restriction
 - ≥ 2 has-wheels
- Inverse Roles
 - has-child, has-parent
- Transitive roles
 - has-child

Description Logic - Tboxes

- Terminological knowledge
- Concept Definitions
 - Father is conjunction of Man and has-child.Human
- Axioms
 - motorcycle *subset-of* vehicle
 - has-favorite.Brewery *subrelation-of* drinks.Beer

Description Logics: Aboxes

- **Assertional knowledge**
- Concept assertions
 - John is-a Man
- Role assertions
 - has-child(John, Bill)

Description Logics: Basic Inferencing

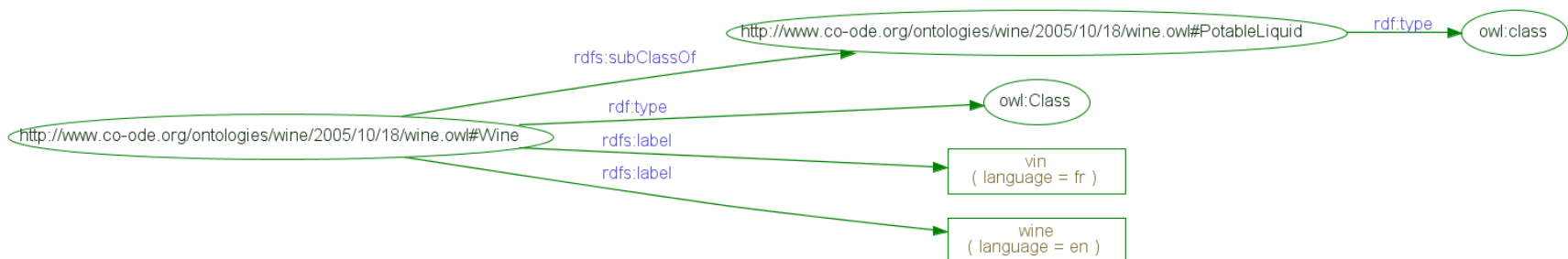
- **Subsumption**
 - Is C1 subclass-of C2
 - Compute taxonomy
- **Consistency**
 - Can C have any individuals
- Note what is **NOT** included
 - Does assertional knowledge follow constraints of terminological knowledge
 - Ontologies will provide an inference mechanisms, not a constraint mechanism (remember the open world)

Namespaces and OWL

```
<?xml version="1.0"?>  
<rdf:RDF xmlns="http://www.co-ode.org/ontologies/wine/2005/10/18/wine.owl#"   
  xml:base="http://www.co-ode.org/ontologies/wine/2005/10/18/wine.owl"   
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"   
  xmlns:dc="http://purl.org/dc/elements/1.1/"   
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"   
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"   
  xmlns:owl="http://www.w3.org/2002/07/owl#">
```

OWL Class Definition

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.co-ode.org/ontologies/wine/2005/10/18/wine.owl#"
  xml:base="http://www.co-ode.org/ontologies/wine/2005/10/18/wine.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class rdf:ID="PotableLiquid"/>
  <owl:Class rdf:ID="Wine">
    <rdfs:subClassOf rdf:resource="#PotableLiquid"/>
    <rdfs:label xml:lang="en">wine</rdfs:label>
    <rdfs:label xml:lang="fr">vin</rdfs:label>
  </owl:Class>
</rdf:RDF>
```



Why owl:class vs. rdfs:class

- Rdfs:class is “class of all classes”
- In DL class can not be treated as individuals (undecidable)
- Thus owl:class, which is expressed as rdfs:subclass of rdfs:class
 - No problem for standard rdf processors since an owl:class “is a” rdfs:class
- Note: there are other times you want to treat class of individuals, but you give up decidability
 - Class drinkable liquids has instances wine, beer,
 - Class wine has instances merlot, chardonnay, zinfandel, ...

OWL class building operations

- **disjointWith**
 - No individuals who are vegetarians are carnivores
- **sameClassAs** (equivalence)
 - All individuals that are in class human are in class homo sapien
 - Relationship is reflexive, unlike subClass
- **Enumerations** (on instances)
 - The Ivy League only has individuals Cornell, Harvard, Yale,
- **Boolean set semantics** (on classes)
 - **Union** (logical disjunction)
 - Class *parent* is union of *mother*, *father*
 - **Intersection** (logical conjunction of class with properties)
 - Class *WhiteWine* is conjunction of things of class *wine* and have property *white*
 - **complimentOf** (logical negation)
 - Class *bad wine* is compliment of class *French wine*

OWL Properties

- Two types
 - ObjectProperty - relations between instances of classes
 - DatatypeProperty - relates an instance to an `rdfs:Literal` or XML Schema datatype
- (Both `rdfs:subClassOf` `rdf:Property`)

```
<owl:DatatypeProperty rdf:ID="name">  
  <rdfs:domain rdf:resource="Person" />  
  <rdfs:range rdf:resource=  
    "http://www.w3.org/2001/XMLSchema/string" />  
</owl:DatatypeProperty>  
  
<owl:ObjectProperty rdf:ID="activity">  
  <rdfs:domain rdf:resource="Person" />  
  <rdfs:range rdf:resource="ActivityArea" />  
</owl: ObjectProperty>
```

All properties must be one or the other.

OWL property building operations & restrictions

- Transitive Property
 - $P(x,y)$ and $P(y,z) \rightarrow P(x,z)$
- SymmetricProperty
 - $P(x,y)$ iff $P(y,x)$
- Functional Property
 - $P(x,y)$ and $P(x,z) \rightarrow y=z$
- inverseOf
 - $P_1(x,y)$ iff $P_2(y,x)$
- InverseFunctional Property
 - $P(y,x)$ and $P(z,x) \rightarrow y=z$
- Cardinality
 - Only 0 or 1 in lite and full

All this allows a whole bunch of new inferences

- Candy bar time....
- RDQLPlus - <http://rdqlplus.sourceforge.net/doc/ridiql.html>

Class/Property Example

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.co-ode.org/ontologies/wine/2005/10/18/wine.owl#"
  xml:base="http://www.co-ode.org/ontologies/wine/2005/10/18/wine.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:class rdf:ID="PotableLiquid"/>
  <owl:Class rdf:ID="Wine">
    <rdfs:subClassOf rdf:resource="#PotableLiquid"/>
    <rdfs:label xml:lang="en">wine</rdfs:label>
    <rdfs:label xml:lang="fr">vin</rdfs:label>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="color">
    <rdfs:domain rdf:resource="#Wine"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema/string"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="Appellation"/>
  <owl:ObjectProperty rdf:ID="hasAppellation">
    <rdfs:domain rdf:resource="#Wine"/>
    <rdfs:range rdf:resource="#Appellation"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="producesWine">
    <rdfs:range rdf:resource="#Wine"/>
    <rdfs:domain rdf:resource="#Appellation"/>
    <owl:inverseOf rdf:resource="#hasAppellation"/>
  </owl:ObjectProperty>
</rdf:RDF>
```

OWL DataTypes

- Full use of XML schema data type definitions
- Examples
 - Define a type age that must be a non-negative integer
 - Define a type clothing size that is an enumeration
“small” “medium” “large”

OWL Instance Creation

- Create individual objects filling in slot/attribute/property definitions

```
<Person ref:ID="William Arms">  
  <rdfs:label>Bill</rdfs:label>  
  <age><xsd:integer rdf:value="57"/></age>  
  <shoesize><xsd:decimal rdf:value="10.5"/></shoesize>  
</Person>
```

OWL Lite Summary

RDF Schema Features:

- [*Class \(Thing, Nothing\)*](#)
- [*rdfs:subClassOf*](#)
- [*rdf:Property*](#)
- [*rdfs:subPropertyOf*](#)
- [*rdfs:domain*](#)
- [*rdfs:range*](#)
- [*Individual*](#)

(In)Equality:

- [*equivalentClass*](#)
- [*equivalentProperty*](#)
- [*sameAs*](#)
- [*differentFrom*](#)
- [*AllDifferent*](#)
- [*distinctMembers*](#)

Property Characteristics:

- [*ObjectProperty*](#)
- [*DatatypeProperty*](#)
- [*inverseOf*](#)
- [*TransitiveProperty*](#)
- [*SymmetricProperty*](#)
- [*FunctionalProperty*](#)
- [*InverseFunctionalProperty*](#)

Property Restrictions:

- [*Restriction*](#)
- [*onProperty*](#)
- [*allValuesFrom*](#)
- [*someValuesFrom*](#)

Restricted Cardinality:

- [*minCardinality*](#) (only 0 or 1)
- [*maxCardinality*](#) (only 0 or 1)
- [*cardinality*](#) (only 0 or 1)

Header Information:

- [*Ontology*](#)
- [*imports*](#)

Class Intersection:

- [*intersectionOf*](#)

Versioning:

- [*versionInfo*](#)
- [*priorVersion*](#)
- [*backwardCompatibleWith*](#)
- [*incompatibleWith*](#)
- [*DeprecatedClass*](#)
- [*DeprecatedProperty*](#)

Annotation Properties:

- [*rdfs:label*](#)
- [*rdfs:comment*](#)
- [*rdfs:seeAlso*](#)
- [*rdfs:isDefinedBy*](#)
- [*AnnotationProperty*](#)
- [*OntologyProperty*](#)

Datatypes

- [*xsd datatypes*](#)

OWL DL and Full Summary

Class Axioms:

- [oneOf](#)
[dataRange](#)
- [disjointWith](#)
- [equivalentClass](#)
(applied to class expressions)
- [rdfs:subClassOf](#)
(applied to class expressions)

Boolean Combinations of Class

Expressions:

- [unionOf](#)
- [complementOf](#)
- [intersectionOf](#)

Arbitrary Cardinality:

- [minCardinality](#)
- [maxCardinality](#)
- [cardinality](#)

Filler Information:

- [hasValue](#)

OWL DL vs. OWL-Full

- Same vocabulary
- OWL DL restrictions
 - Type separation
 - Class can not also be an individual or property
 - Property can not also be an individual or class
 - Separation of ObjectProperties and DatatypeProperties

Language Comparison

	DTD	XSD	RDF(S)	OWL
Bounded lists (“X is known to have exactly 5 children”)				X
Cardinality constraints (Kleene operators)	X	X		X
Class expressions (unionOf, complementOf)				X
Data types		X		X
Enumerations	X	X		X
Equivalence (properties, classes, instances)				X
Formal semantics (model-theoretic & axiomatic)				X
Inheritance			X	X
Inference (transitivity, inverse)				X
Qualified constraints (“all children are of type person”)				X
Reification			X	X

Storing and querying RDF-based models

- Persistent storage implementations
 - Jena 2 - <http://www.hpl.hp.com/semweb/jena2.htm>
 - Relational databases (mysql , postgres, oracle)
 - Kowari – <http://www.kowari.org>
 - Mapped files
 - Sesame - <http://www.openrdf.org/>
 - Relational databases (mysql, postgres, oracle)
- Query languages
 - RDQL (Kowari, Jena)
 - SPARQL
 - W3C working draft
 - <http://www.w3.org/TR/rdf-sparql-query/>

RDQL-by-example

- RDF source
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-db-3.rdf>
- Queries
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q1>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q2>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q3>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q4>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q5>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q6>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q7>
 - <http://www.cs.cornell.edu/courses/cs431/2006sp/examples/RDQL/vc-q8>

Protégé and RACER – tools for building, manipulating and reasoning over ontologies

- Protégé - <http://protege.stanford.edu/>
 - Use the 3.x version
 - Multiple plug-ins are available
- Protégé OWL plug-in
 - <http://protege.stanford.edu/plugins/owl/>
- Other semantic web related plug-ins
 - <http://protege.cim3.net/cgi-bin/wiki.pl?ProtegePluginsLibraryByTopic#nid349>
- Racer
 - Description Logic based reasoning engine
 - Server-based
 - Integrates with Protégé-OWL