Ontology Languages

Πληροφοριακά Συστήματα Διαδικτύου

Ontology Languages

- "Ontology Languages for the Semantic Web" paper, IEEE Intelligent Systems, Jan/Feb 2002, pp 54-60, analyze the most representative ontology languages created for the Web, and compare them using a common framework.

  - **XOL** (XML-based Ontology Exchange Language)
  - **SHOE** (Simple HTML Ontology Extension)
  - **OML** (Ontology Markup Language)
  - **RDF(S)** (Resource Description Framework (Schema))
  - **OIL** (Ontology Interchange Language)
  - **DAML+OIL** (DARPA Agent Markup Language + OIL)
  - **OWL** (Ontology Web Language)

XML: fundament (w3c.org)

- It provides a standardized syntactical way to represent and exchange tree structures.
- XML schema allow to define a schema for XML documents and may already provide machine-understandable semantics of data:

```
<name>
        <first>John</first>
        <last>Smith</last>
</name>
```

- Strength of XML: **Its generality**

RDF Schema (w3c.org)

- The *Resource Description Framework*

  RDF=a standard datamodel for machine-processable semantics.

  - The basic construction in RDF is an object-attribute-value triple.
- *RDF Schema* defines a set of modeling primitives for structured vocabularies for machine-processable semantics of information.
  - Two crucial RDF Schema constructions are *subClassOf* and *subPropertyOf* allowing hierarchical structured vocabularies.

Requirements

Desirable features identified for a Web Ontology Language :

- **Compatible** with existing Web standards (XML, RDF, RDFS)
- **Easy to understand** and use
- **Formally specified** and of “adequate” expressive power
- Possible to provide **automated reasoning** support

Beyond RDF: OIL & DAML

- **OIL** (Ontology Inference Layer) extends RDF Schema to a full-fledged ontology representation language.
  - intuitive syntax plus high expressive power
  - well defined semantics
  - can use Description Logic systems to reason
  - http://www.ontoknowledge.org/oil/
- **DAML** (DARPA Agent Markup Language) / DAML-ONT = US sister of OIL
  - http://www.daml.org/

From OIL to DAML+OIL

- Efforts merged to produce DAML+OIL (development was overseen by joint EU/US committee)
- DAML+OIL **builds** on top of RDFS
  - RDFS based **syntax**
  - **Inherits** RDFS ontological primitives (subclass, range, domain)
  - Adds **much** richer set of primitives (transitivity, cardinality,...)
- DAML+OIL designed to describe the **structure** of a domain (**schema**)
  - **Object oriented**: classes (concepts) and properties (roles)
  - DAML+OIL ontology consists of set of **axioms** asserting characteristics of classes and properties
  - E.g., Person is **kind_of** Animal whose parents are Persons
- http://www.w3.org/Submission/2001/12/

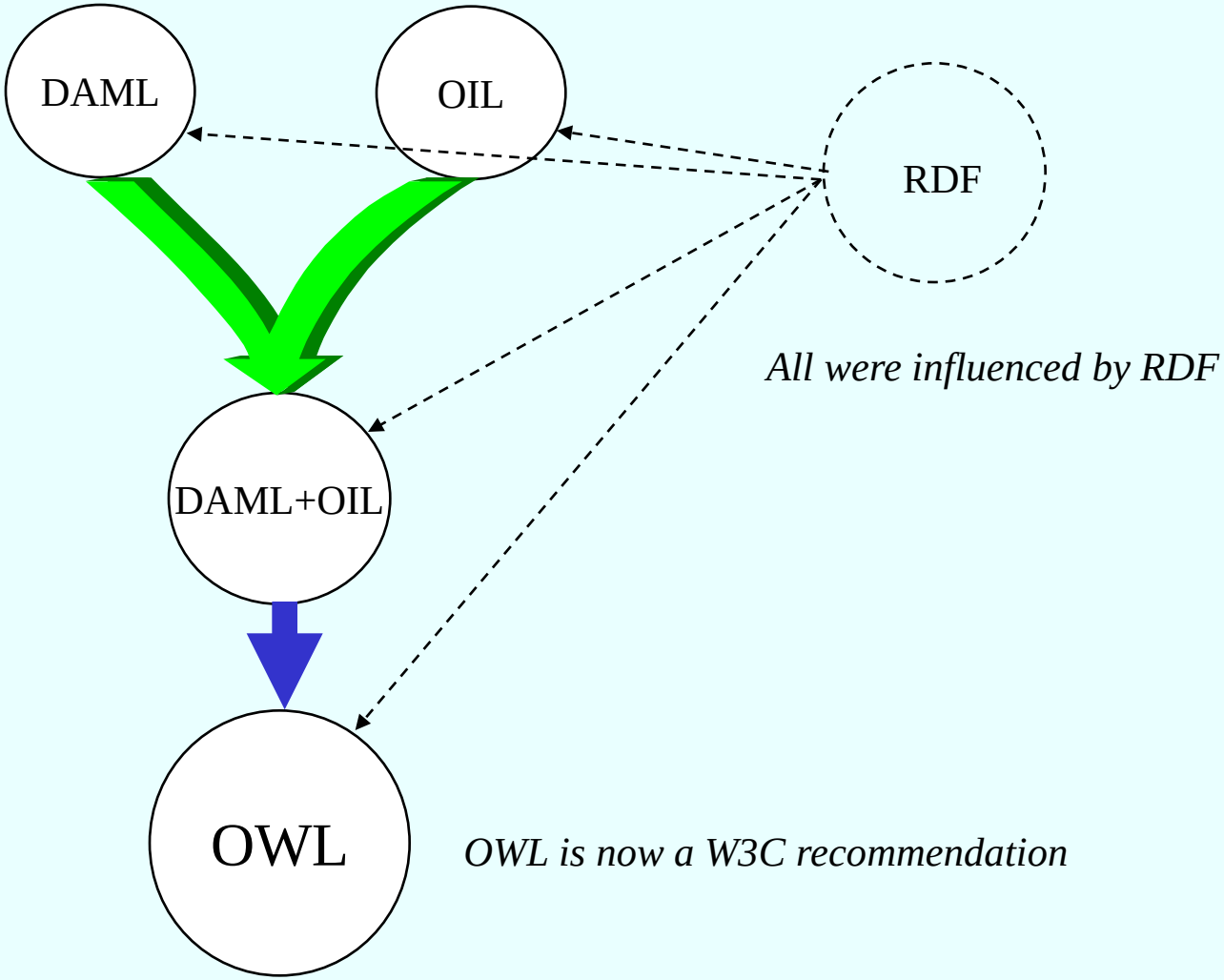
How DAML+OIL builds on RDFS

- Extends expressive power
  - Constraints (restrictions) on properties of classes (existential/universal/cardinality)
  - Boolean combinations of classes and restrictions
  - Equivalence, disjointness, coverings
  - Necessary and sufficient conditions
  - Constraints on properties
- Provides well defined semantics
  - Meaning of DAML+OIL statements is formally specified
  - Both model theoretic and axiomatic specifications provided
  - Allows for machine understanding and automated reasoning

OWL (Ontology Web Language)

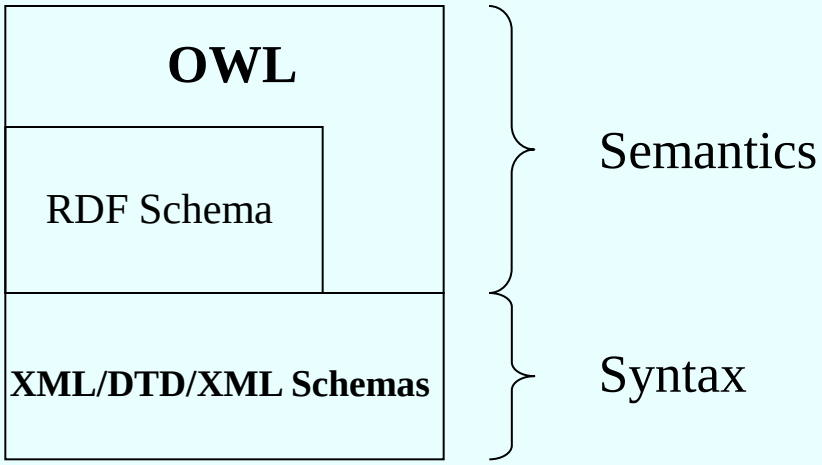
- OWL is a W3C Recommendation
- The **purpose** of OWL is identical to RDFS i.e. to provide an XML vocabulary to define classes, properties and their relationships.
  - RDFS enables you to express very rudimentary relationships and has limited inferencing capability.
  - OWL enables you to express much richer relationships, thus yielding a much enhanced inferencing capability.
- The **benefit** of OWL is that it facilitates a much greater degree of inferencing than you get with RDF Schemas.

Origins of OWL



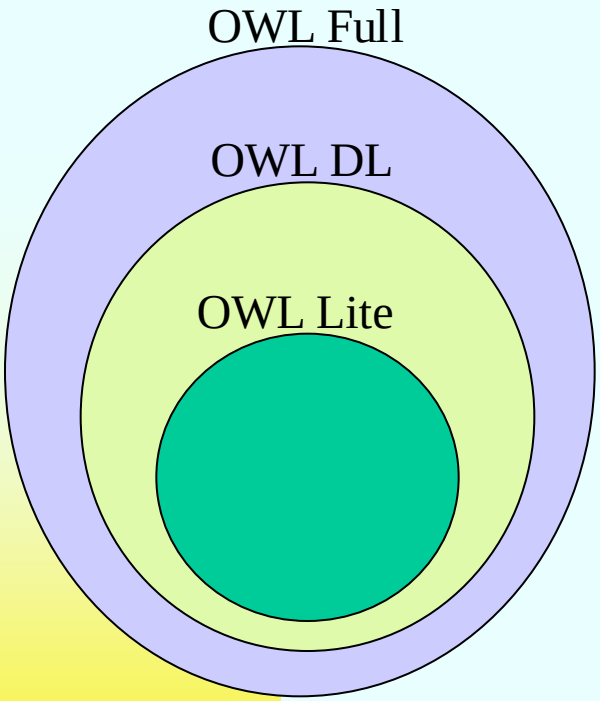
Origins of OWL

- OWL and RDF Schema enables *machine-processable semantics*



Versions of OWL

- Depending on the intended usage, OWL provides three increasingly expressive sublanguages



Full: Very expressive, no computation guarantees

DL (Description Logic): Maximum expressiveness, computationally complete.

Lite: Simple classification hierarchy with simple constraints.

Advantages/Disadvantages of versions

- Full:
  - The advantage of the Full version of OWL is that you get the full power of the OWL language.
  - The disadvantage is that it is very difficult to build a tool for this version. Also, the user of a Full-compliant tool may not get a quick and complete answer.
- DL/Lite:
  - The advantage of the DL or Lite version of OWL is that tools can be built more quickly and easily, and users can expect responses from such tools to come quicker and be more complete.
  - The disadvantage is that you don't have access to the full power of the language.

OWL DL

- OWL is based on Description Logic
- Description Logic is a fragment of first-order logic
- OWL inherits from Description Logic
  - The open-world assumption
  - The non-unique-name assumption

Open-world assumption

- We cannot conclude some statement x to be false simply because we cannot show x to be true i.e. we may not deduce falsity from the absence of truth
- It is the opposite of the **closed world assumption**, which holds that any statement that is not known to be true is false.

Open-world assumption: Example

- Statement: Marios is a citizen of Greece
- Question: Is George a citizen of Greece?
- "Closed world" (e.g. SQL) answer: **No**
- "Open world" answer: **unknown**
  ("I don't know if he is a citizen, but that's not enough reason to conclude that he isn't")

Unique-name assumption (UNA)

- When two individuals are known by different names, they are in fact different individuals
- This is an assumption that sometimes works (e.g. Product codes) and sometimes doesn't (e.g. Social environment)
- OWL does not make the unique-name assumption

Describing classes in OWL

OWL vs. RDFS

- Abstraction mechanism to group resources with similar characteristics
- OWL allows greater expressiveness, but
- OWL (DL/Lite) puts certain constraints on the use of RDF
  - i.e. a class may not act as an instance of another (meta)class (the same holds for properties)

RDFS

```
<rdfs:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
</rdfs:Class>
```

OWL

```
<owl:Class rdf:ID="River">
    <rdfs:subClassOf rdf:resource="#Stream"/>
</owl:Class>
```

Describing classes in OWL

Complex Classes

- Intersection of classes (owl:intersectionOf)
  - OR ($A \cup B$)
- Union of classes (owl:unionOf)
  - AND ($A \cap B$)
- Complement (owl:complementOf)
  - NOT
- Enumeration (owl:oneOf)
- Disjoint Classes (owl:disjointWith)

Describing classes in OWL

Property Restrictions

- Defining a Class by restricting its possible instances via their property values
- OWL distinguishes between the following two:
  - Value constraint
  - Cardinality constraint

Describing classes in OWL

Restrictions on Property Classes

- **Properties:**
  - allValuesFrom: *rdfs:Class (lite/DL owl:Class)*
  - hasValue: *specific Individual*
  - someValuesFrom: *rdfs:Class (lite/DL owl:Class)*
  - cardinality: *xsd:nonNegativeInteger (in lite {0,1})*
  - minCardinality: *xsd:nonNegativeInteger (in lite {0,1})*
  - maxCardinality: *xsd:nonNegativeInteger (in lite {0,1})*

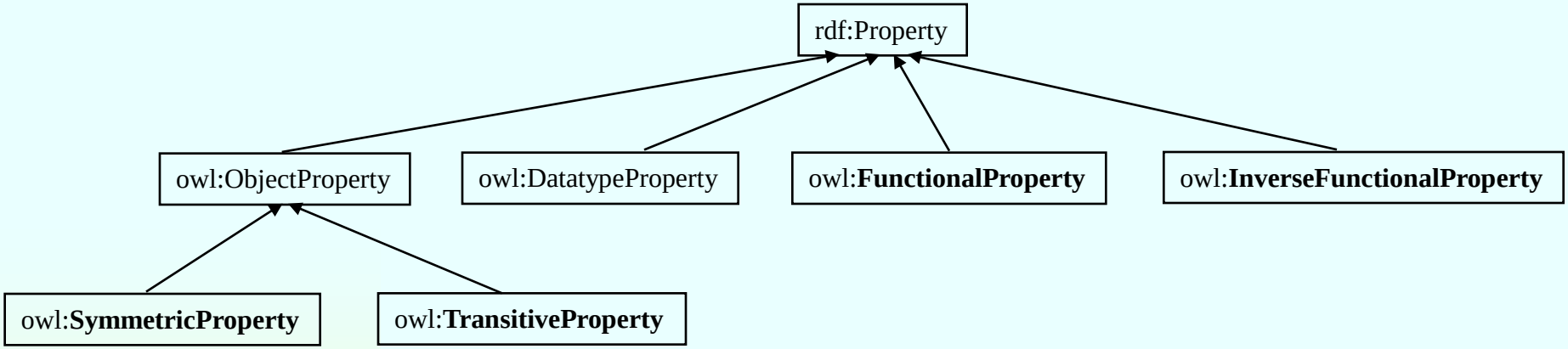
Describing properties in OWL

OWL vs. RDFS

- RDF Schema provides some of predefined properties:
  - `rdfs:range` used to indicate the range of values for a property.
  - `rdfs:domain` used to associate a property with a class.
  - `rdfs:subPropertyOf` used to specialize a property.
  - …
- OWL provides additional predefined properties:
  - `owl:cardinality` (indicate cardinality)
  - `owl:hasValue` (at least one of the specified property values)
  - …
- OWL provides additional property classes, which allow reasoning and inferencing:
  - `owl:FunctionalProperty`
  - `owl:TransitiveProperty`
  - …

Describing properties in OWL

OWL Property Classes



- An ObjectProperty relates one Resource to another Resource.
- An DatatypeProperty relates one Resource to a Literal an XML Schema data type.

Describing properties in OWL

- **owl:TransitiveProperty** (transitive property)
  - E.g. “has better grade than”, “is ancestor of”
- **owl:SymmetricProperty** (symmetry)
  - E.g. “has same grade as”, “is sibling of”
- **owl:FunctionalProperty** defines a property that has at most one value for each object
  - E.g. “age”, “height”, “directSupervisor”
- **owl:InverseFunctionalProperty** defines a property for which two different objects cannot have the same value

OWL tools

- Commercial Ontology Support Tools
  - SNOBASE
  - Cerebra
- Reasoners
  - FaCT
  - Racer
  - Cerebra
- Editors
  - OWL plug-in for Protégé
- APIs
  - Jena
  - Cerebra
- Parser/Validators

Reference: http://www.w3.org/2001/sw/WebOnt/impls

OWL 2.0

- OWL 2 extends OWL with a small but useful set of features that have been requested by users, which include extra syntactic sugar, additional property and qualified cardinality constructors, extended datatypes support, simple meta-modelling, and extended annotations.
- Apart from addressing problems with expressivity, the goal of OWL 2 was to provide a robust platform for future development.

References

- W3C Documents
  - Guide
    http://www.w3.org/TR/owl-guide/
  - Reference
    http://www.w3.org/TR/owl-ref/
  - Semantics and Abstract Syntax
    http://www.w3.org/TR/owl-semantics/
- OWL Tutorial
  - http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/