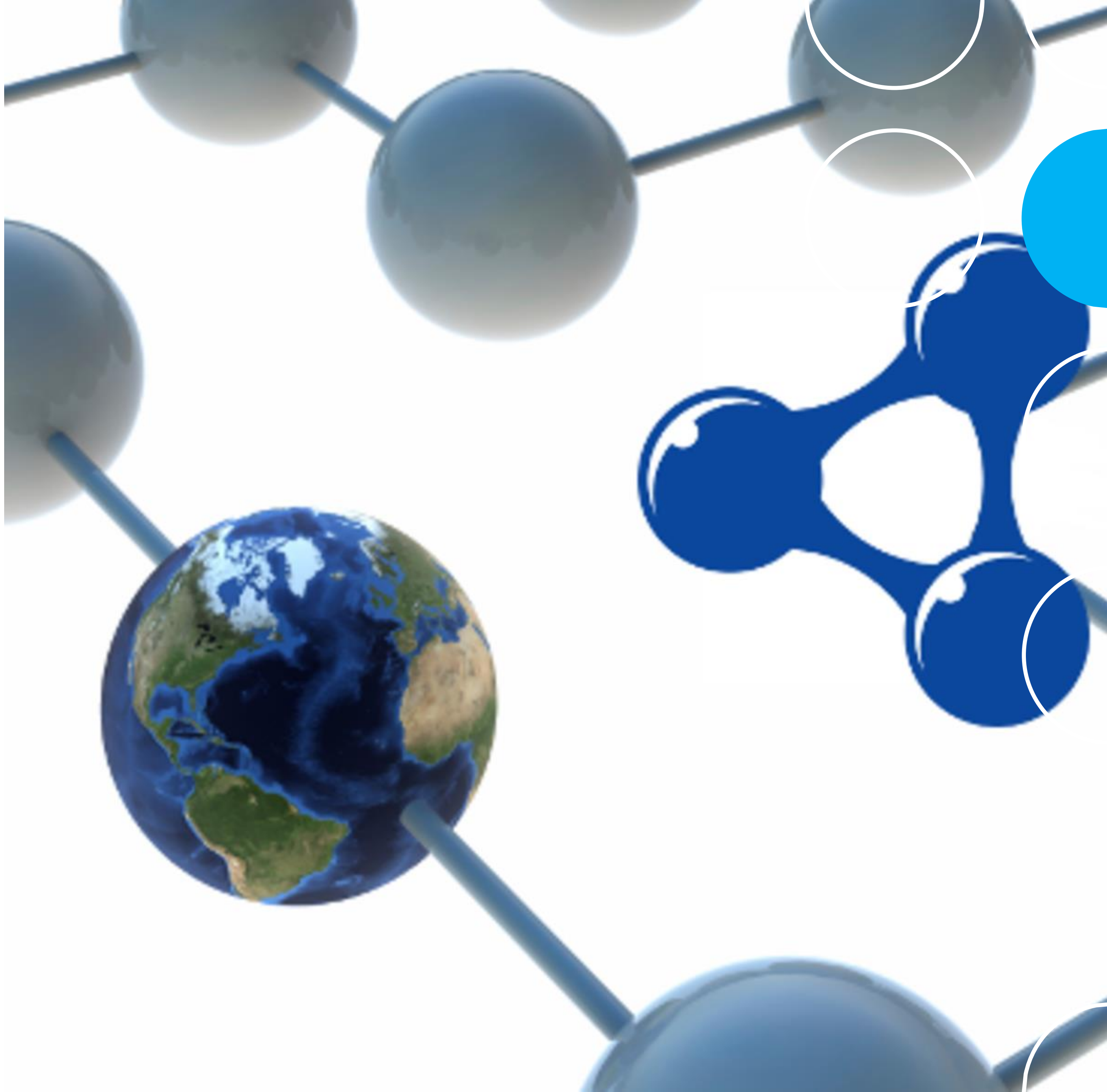
Semantic Web and Linked Data

Liliana Ferreira

2022/23



Class 8: Learning Objectives

- Introduction to of ontologies;
- Types of ontologies;

- The OWL ontology language
  - Logic and Inference;
  - The Protégé open-source tool.

Remember...

- Next class:
  - Practical Work Check Point II
  - Short presentation to the class including context and motivation, identified requirements, existing solutions/approaches for the same problem, knowledge sources to be used and architecture of the solution (byLaws).

Introduction

RDF(S) too weak to describe resources in sufficient detail

- No localized range and domain constraints.
  - E.g., cannot state that the range of hasChild is person when applied to persons and elephant when applied to elephants
- No combination of classes with union, intersection, and complement.
- No existence/cardinality constraints.
  - E.g., cannot state that all persons have a mother that is also a person, or that persons have exactly 2 parents
- No transitive, inverse or symmetrical properties.
  - E.g., cannot state that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical

Introduction

- Difficult to provide reasoning support for predicates beyond the RDF and RDFS namespace as there is not formal foundation on which you can build those.

Remember...

- Why do we need reasoning?
  - Ensure that a knowledge base (KB) is:
    - Meaningful - all named classes can have instances
    - Correct - captured intuitions of domain experts
    - Minimally redundant - no unintended synonyms
- Answer queries over ontology classes and instances, e.g.:
  - Find more general and specific classes
  - Retrieve individuals or tuples matching a given query

Ontologies

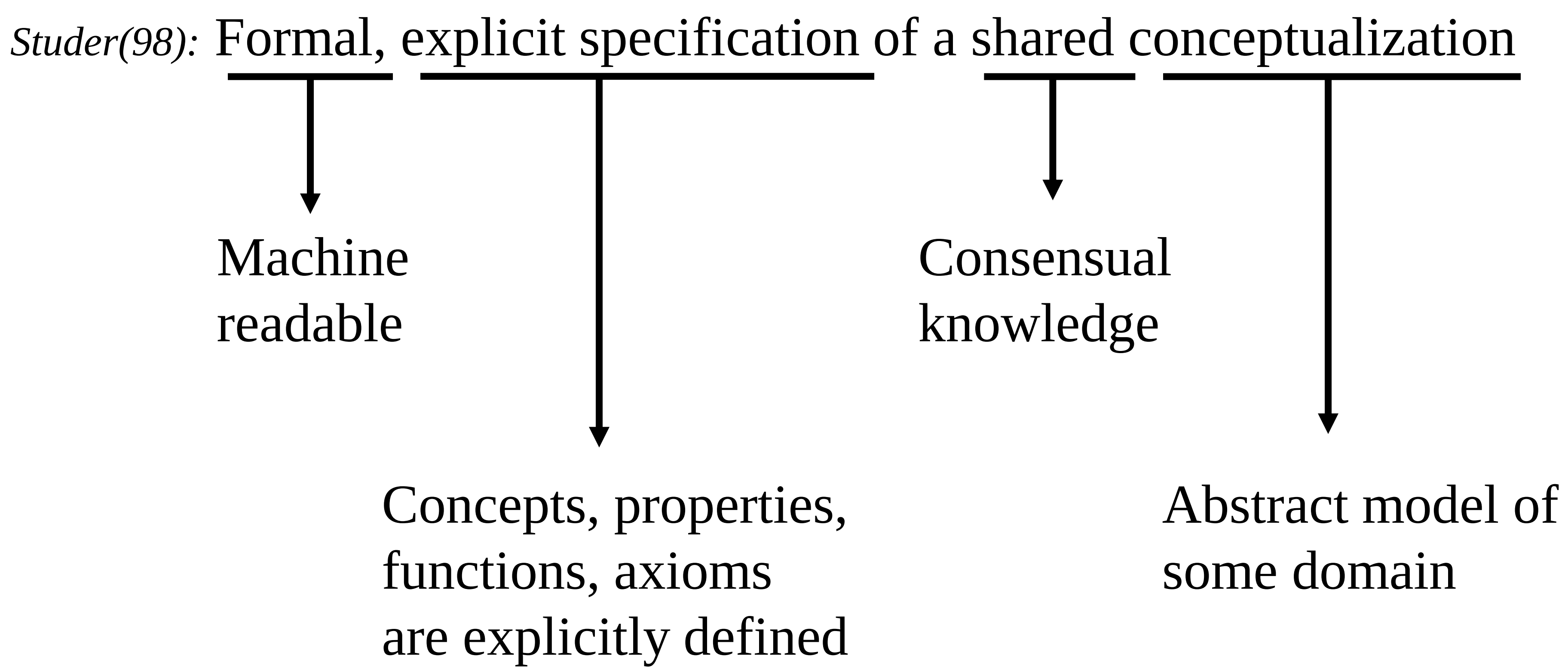
Ontologies are a popular research topic in Artificial Intelligence (AI), e.g.:

- Knowledge Engineering, Natural Language Processing, Intelligent Information Integration and Multi-agent systems.

By attaching information to data that describe its contents and meaning, web resources can be used by machines.

The data can be used not just for display purposes, but also for automation, integration and reuse of data across various applications.

What is an ontology?


Studer(98): Formal, explicit specification of a shared conceptualization
Machine readable
Concepts, properties, functions, axioms are explicitly defined
Consensual knowledge
Abstract model of some domain

Ontology

What exists in a domain and how they relate with each other.

- In general, an ontology formally describes a domain of discourse.
- An ontology consists of a finite list of terms (i.e. concepts) and the relationships between the terms (i.e. properties).
- The terms denote important concepts (classes of objects) of the domain.
- For example, in a university setting: staff members, students, courses, modules, lecture theatres, and schools are some important concepts.

For what is an ontology good for?

Enable knowledge sharing and knowledge reuse.

- Ontologies capture general knowledge about a domain that is changing rarely and specify concepts and relations about which knowledge is to be accumulated and processed.

- For information systems or for the Internet, ontologies can be used to organize keywords and database concepts by capturing the semantic relationships among the keywords or among the tables and fields in a database.

Why do I need to develop an ontology?

1. To provide more precise definition to resources and make them more suitable to machine processing;
2. To define explicit assumptions for a domain:
   a) Easier to exchange domain assumptions;
   b) Flexible to change/update the assumptions;
   c) Easier to interpret and update legacy data.
3. To separate domain knowledge from operational knowledge
   a) Re-use domain and operational knowledge separately;
4. A common reference to define concepts and relationships between concepts and other axioms in a domain;
5. To share a consistent understanding of a domain.

Criteria for Introducing Ontologies

- Large amounts of data
  - Data available on the Web
  - Data acquired or generated by new techniques
- Complex data structures
  - Inheritance, containment and other hierarchies
  - Many relationships
- Diverse sources
  - Many legacy systems
  - Sources on the Web using different formats
- Requirement for formal proofs
  - Contracts and policy enforcement

Terminological Systems

Terminological systems can be seen as basic examples of ontologies; for example:

Terminologies	list of terms referring to concepts in a particular domain;
Thesaurus	terms are ordered alphabetically and concepts maybe described by synonymous terms;
Vocabulary	concepts are defined in formal or free text form;
Classification	concepts are organized using generic (i.e. is_a) relationships;
Coding systems	codes designate concepts.

Types of ontologies

A common way to classify ontologies is depending on their level of generality:

- **Domain ontologies** contain knowledge that is valid for a particular type of domain (e.g. electronic, medical, mechanic);
- **Meta data ontologies** provide a vocabulary for describing the content of information sources in the World Wide Web;
- **Generic or top-level ontologies** (aka common-sense ontologies) capture general knowledge. The conceptualizations defined here are valid across several domains.
- **Method and task ontologies** can provide terms for particular tasks or problem-solving methods. They are used for reasoning on domain knowledge.

Ontology Elements

- Concepts(classes) + their hierarchy
- Concept properties (slots/attributes) + their hierarchy
- Property restrictions (type, cardinality, domain ...)
- Relations between concepts (disjoint, equality ...)
- Instances

Ontology Languages

- Ontologies are formal theories about a certain domain of discourse and therefore require a formal logical language to express them.
- Languages for defining ontologies are syntactically and semantically rich languages, e.g. richer than common approaches for databases.

The OWL Language

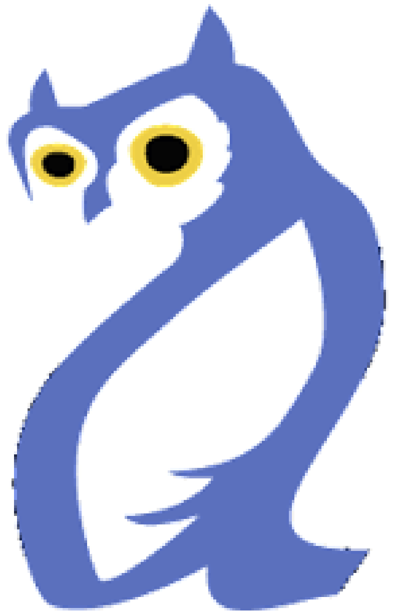
- Web Ontology Language (OWL) is the W3C recommendation for representing ontologies on the Web;
- OWL is a semantic markup language for defining, publishing and sharing ontologies in the World Wide Web;
- OWL contains specific constructs to represent the domain and range of properties, subclass and other axioms and constraints on the values that can be assigned to the property of an object;
- OWL is based on Description Logics knowledge representation formalism.

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

The OWL Language

- OWL is a W3C Recommendation
  - OWL was published in 2004
  - OWL 2 was published in 2012

- Motivations:
  - A well-defined syntax
  - A formal semantics
  - Efficient reasoning support

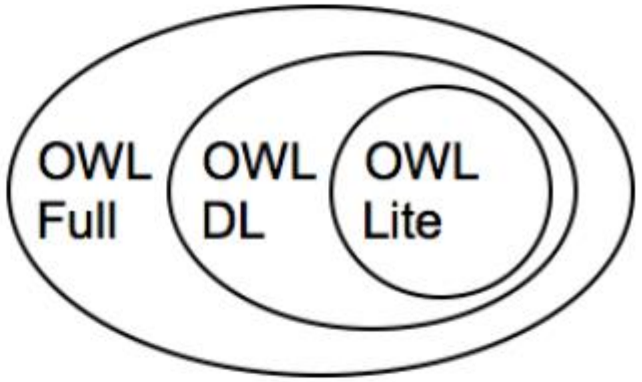


OWL and Description Logic

- OWL (DL) benefits from extensive work on DL research:
  - Well defined semantics
  - Formal properties (describing complexity, and providing decidability)
  - Known reasoning algorithms
  - Implemented systems

OWL "flavors"

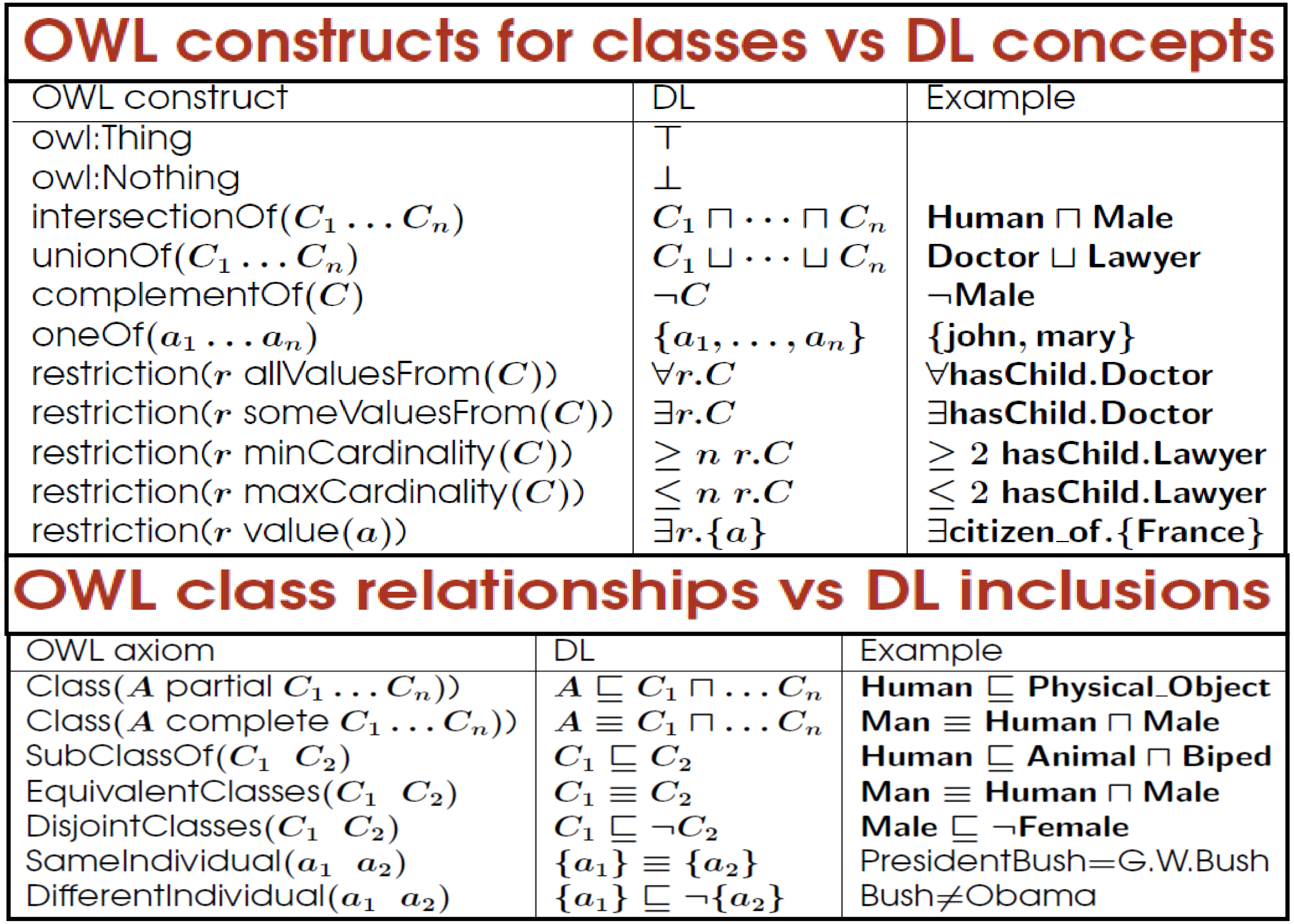
- W3C's Web Ontology Working Group defined OWL as three different sublanguages:
  - OWL Full
  - OWL DL
  - OWL Lite



- Each sublanguage geared toward fulfilling different aspects of requirements

OWL DL

- OWL DL (Description Logics)
  - Based on FOL semantics
- OWL DL permits efficient reasoning support
  - The most expressive decidable OWL sub-language
- But we lose full compatibility with RDF:
  - Every legal OWL DL document is a legal RDF document.
  - Not every RDF document is a legal OWL DL document.



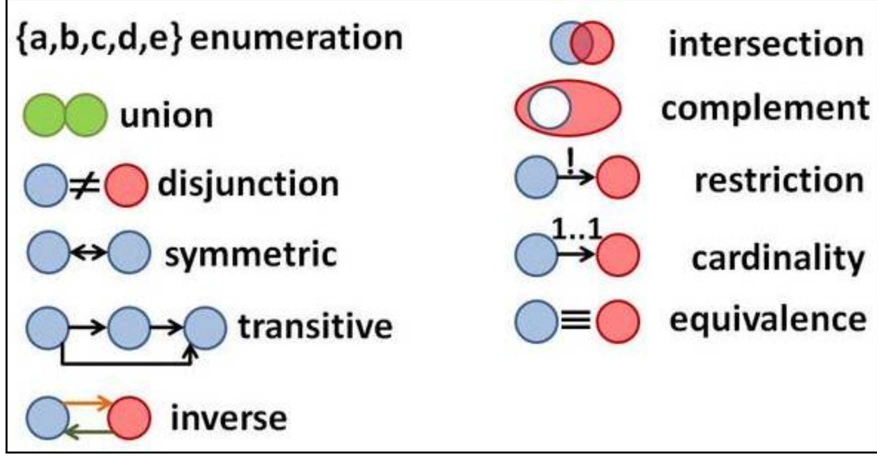
OWL constructs for classes vs DL concepts

OWL construct	DL	Example
owl:Thing	$\top$	
owl:Nothing	$\bot$	
intersectionOf$(C_1 \ldots C_n)$	$C_1 \sqcap \cdots \sqcap C_n$	**Human** $\sqcap$ **Male**
unionOf$(C_1 \ldots C_n)$	$C_1 \sqcup \cdots \sqcup C_n$	**Doctor** $\sqcup$ **Lawyer**
complementOf(C)	$\neg C$	$\neg$**Male**
oneOf$(a_1 \ldots a_n)$	$\{a_1, \ldots, a_n\}$	**{john, mary}**
restriction(r allValuesFrom(C))	$\forall r.C$	$\forall$**hasChild.Doctor**
restriction(r someValuesFrom(C))	$\exists r.C$	$\exists$**hasChild.Doctor**
restriction(r minCardinality(C))	$\geq n\ r.C$	≥ 2 **hasChild.Lawyer**
restriction(r maxCardinality(C))	$\leq n\ r.C$	≤ 2 **hasChild.Lawyer**
restriction(r value(a))	$\exists r.\{a\}$	$\exists$**citizen_of.{France}**

OWL class relationships vs DL inclusions

OWL axiom	DL	Example
Class(A partial $C_1 \ldots C_n$))	$A \sqsubseteq C_1 \sqcap \ldots C_n$	**Human** $\sqsubseteq$ **Physical_Object**
Class(A complete $C_1 \ldots C_n$))	$A \equiv C_1 \sqcap \ldots C_n$	**Man** $\equiv$ **Human** $\sqcap$ **Male**
SubClassOf$(C_1\ \ C_2)$	$C_1 \sqsubseteq C_2$	**Human** $\sqsubseteq$ **Animal** $\sqcap$ **Biped**
EquivalentClasses$(C_1\ \ C_2)$	$C_1 \equiv C_2$	**Man** $\equiv$ **Human** $\sqcap$ **Male**
DisjointClasses$(C_1\ \ C_2)$	$C_1 \sqsubseteq \neg C_2$	**Male** $\sqsubseteq \neg$**Female**
SameIndividual$(a_1\ \ a_2)$	$\{a_1\} \equiv \{a_2\}$	PresidentBush=G.W.Bush
DifferentIndividual$(a_1\ \ a_2)$	$\{a_1\} \sqsubseteq \neg\{a_2\}$	Bush≠Obama

OWL on one Slide



- **Symmetric**: if P(x, y) then P(y, x)
- **Transitive**: if P(x,y) and P(y,z) then P(x, z)
- **Functional**: if P(x,y) and P(x,z) then y=z
- **InverseOf**: if P1(x,y) then P2(y,x)
- **InverseFunctional**: if P(y,x) and P(z,x) then y=z
- **allValuesFrom**: P(x,y) and y=allValuesFrom(C)
- **someValuesFrom**: P(x,y) and y=someValuesFrom(C)
- **hasValue**: P(x,y) and y=hasValue(v)
- **cardinality**: cardinality(P) = N
- **minCardinality**: minCardinality(P) = N
- **maxCardinality**: maxCardinality(P) = N

- **equivalentProperty**: P1 = P2
- **intersectionOf**: C = intersectionOf(C1, C2, …)
- **unionOf**: C = unionOf(C1, C2, …)
- **complementOf**: C = complementOf(C1)
- **oneOf**: C = one of(v1, v2, …)
- **equivalentClass**: C1 = C2
- **disjointWith**: C1 != C2
- **sameIndividualAs**: I1 = I2
- **differentFrom**: I1 != I2
- **AllDifferent**: I1 != I2, I1 != I3, I2 != I3, …
- **Thing**: I1, I2, …

Caption:
- Properties are indicated by: P, P1, P2, etc
- Specific classes are indicated by: x, y, z
- Generic classes are indicated by: C, C1, C2
- Values are indicated by: v, v1, v2
- Instance documents are indicated by: I1, I2, I3, etc.
- A number is indicated by: N
- P(x,y) is read as: “property P relates x to y”

An example

- Woman $\equiv$ Person $\sqcap$ Female
- Man $\equiv$ Person $\sqcap$ $\neg$Woman
- Mother $\equiv$ Woman $\sqcap$ $\exists$hasChild.Person
- Father $\equiv$ Man $\sqcap$ $\exists$hasChild.Person
- Parent $\equiv$ Father $\sqcup$ Mother
- Grandmother $\equiv$ Mother $\sqcap$ $\exists$hasChild.Parent

We can further infer (though not explicitly stated):

→ Grandmother $\sqsubseteq$ Person

Grandmother $\sqsubseteq$ Man $\sqcup$ Woman

etc.

A Sample Ontology

African Wildlife Ontology (AWO)

- AWO contains knowledge about wildlife:
  - Giraffes eat leaves and twigs;
  - Giraffes are herbivores;
  - Hervibores are animals, and so on.

A Sample Ontology

African Wildlife Ontology (AWO)

- A mathematician may prefer to represent such knowledge with first order predicate logic (FOL).
- Example:

$$\forall x(Lion(x) \rightarrow \forall y(eats(x,y) \wedge Herbivore(y)) \wedge \exists z(eats(x,z) \wedge Impala))$$

this states that ...

A Sample Ontology

African Wildlife Ontology (AWO)

- A mathematician may prefer to represent such knowledge with first order predicate logic (FOL).
- Example:

$$\forall x(Lion(x) \rightarrow \forall y(eats(x,y) \wedge Herbivore(y)) \wedge \exists z(eats(x,z) \wedge Impala))$$

this states that “all lions eat herbivores, and they also eat some impalas”.

A Sample Ontology

African Wildlife Ontology (AWO)

One can represent the same knowledge also in other logic languages.

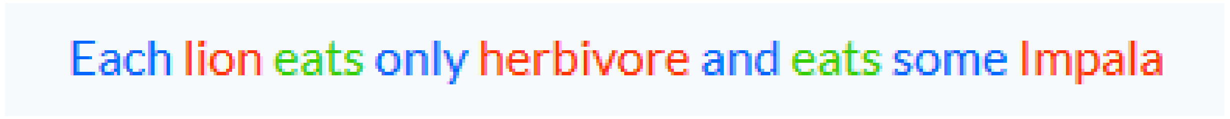
- Example, in Description Logic language:

$$Lion \sqsubseteq \forall eats.Herbivore \sqcap \exists eats.Impala$$

A Sample Ontology

African Wildlife Ontology (AWO)

- A domain expert will typically prefer a more user-friendly rendering, such as an automatically generated (pseudo-)natural language rendering, e.g.:

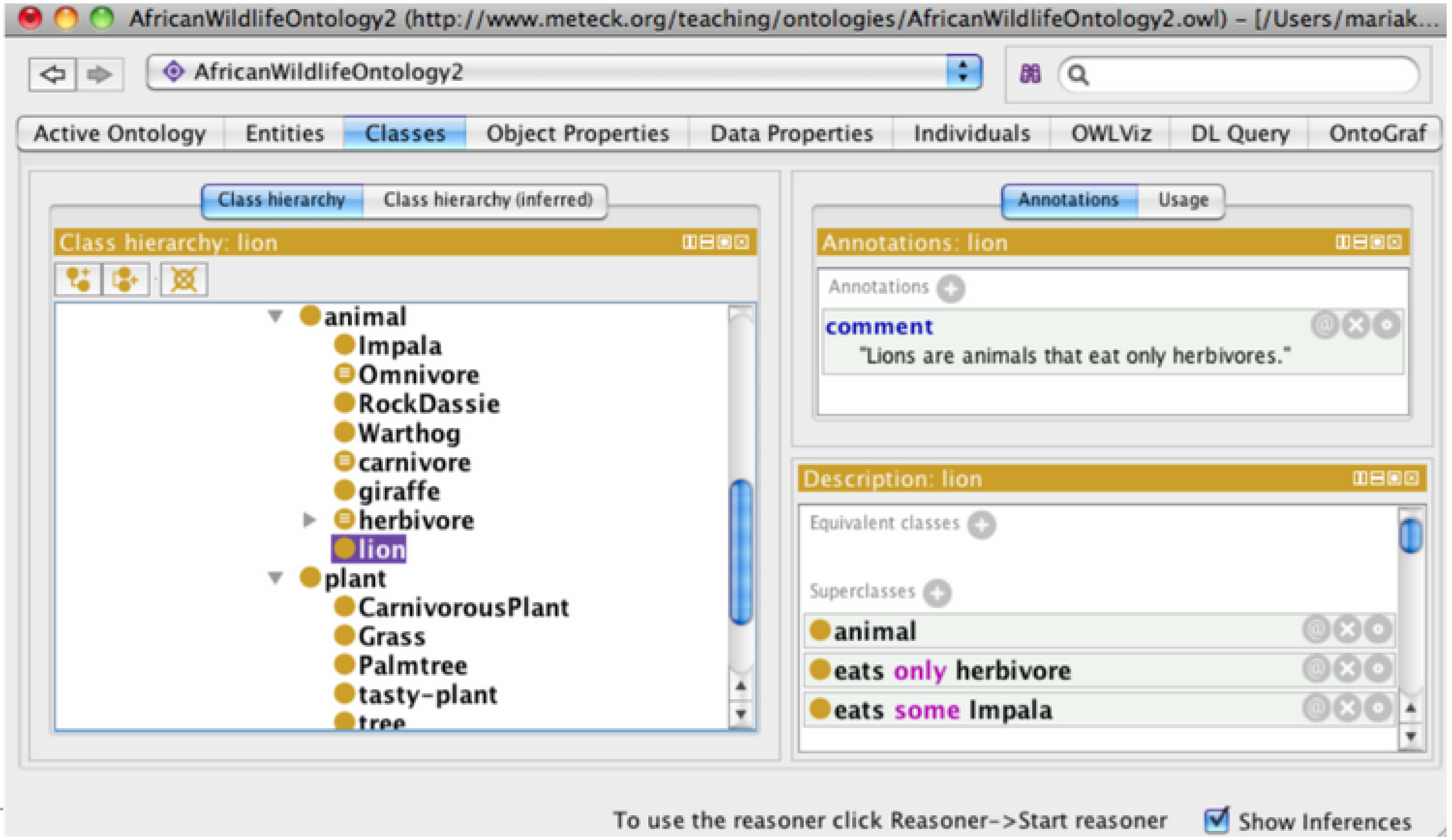

Each lion eats only herbivore and eats some Impala

where the first “∀” in first equation is verbalized as each and the second one as only, the “∧” as and, and the “∃” as some.

A Sample Ontology

African Wildlife Ontology (AWO)

In Protégé tool:


AfricanWildlifeOntology2 (http://www.meteck.org/teaching/ontologies/AfricanWildlifeOntology2.owl) – [/Users/mariak...

Class hierarchy: lion

- animal
  - Impala
  - Omnivore
  - RockDassie
  - Warthog
  - carnivore
  - giraffe
  - herbivore
  - lion
- plant
  - CarnivorousPlant
  - Grass
  - Palmtree
  - tasty-plant
  - tree

Annotations: lion

Annotations

comment
"Lions are animals that eat only herbivores."

Description: lion

Equivalent classes

Superclasses

animal

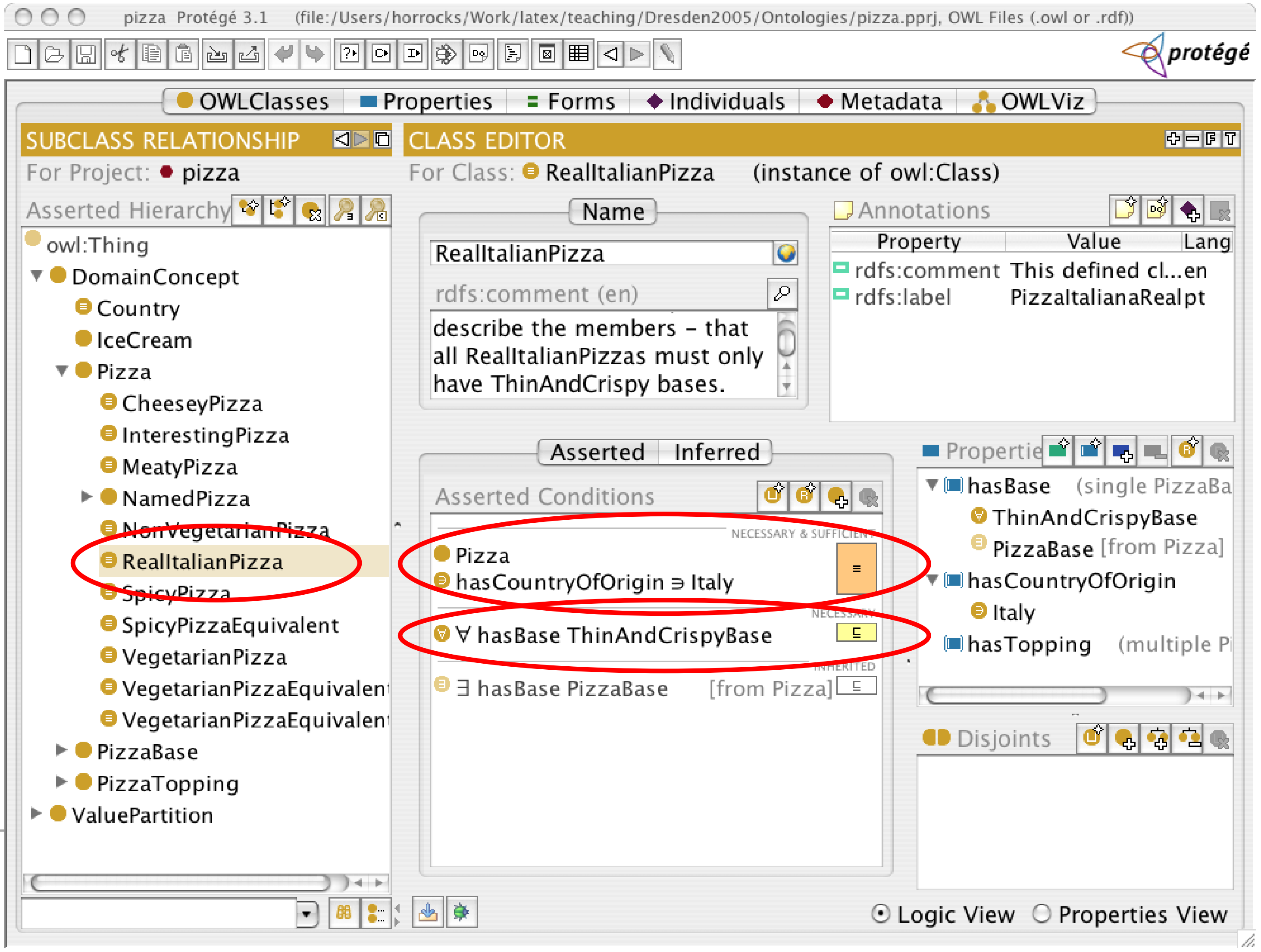
eats only herbivore

eats some Impala

To use the reasoner click Reasoner->Start reasoner

Show Inferences

Example Ontology (Protégé)



pizza Protégé 3.1 (file:/Users/horrocks/Work/latex/teaching/Dresden2005/Ontologies/pizza.pprj, OWL Files (.owl or .rdf))

For Project: pizza

For Class: RealItalianPizza (instance of owl:Class)

owl:Thing
- DomainConcept
  - Country
  - IceCream
  - Pizza
    - CheeseyPizza
    - InterestingPizza
    - MeatyPizza
    - NamedPizza
    - NonVegetarianPizza
    - RealItalianPizza
    - SpicyPizza
    - SpicyPizzaEquivalent
    - VegetarianPizza
    - VegetarianPizzaEquivalent
    - VegetarianPizzaEquivalent
  - PizzaBase
  - PizzaTopping
- ValuePartition

Name: RealItalianPizza

rdfs:comment (en)

describe the members – that all RealItalianPizzas must only have ThinAndCrispy bases.

Property	Value	Lang
rdfs:comment	This defined cl...en	
rdfs:label	PizzaItalianaRealpt	

NECESSARY & SUFFICIENT
Pizza
hasCountryOfOrigin ∋ Italy
NECESSARY
∀ hasBase ThinAndCrispyBase
INHERITED
∃ hasBase PizzaBase [from Pizza]

- hasBase (single PizzaBa
  - ThinAndCrispyBase
  - PizzaBase [from Pizza]
- hasCountryOfOrigin
  - Italy
- hasTopping (multiple P

OWL
Simple Classes and Individuals

- An important use of an ontology will depend on the ability to reason about individuals.
  - This requires a mechanism to describe the classes that individuals belong to and the properties that they inherit by virtue of class membership.
  - We can always assert specific properties about individuals, but much of the power of ontologies comes from class-based reasoning.
- Sometimes we want to emphasize the distinction between a class as an object and a class as a set containing elements.
  - The set of individuals that are members of a class are called the extension of the class.

Simple Named Classes

- Every individual in the OWL world is a member of the class owl:Thing.
- Named classes
  - Example:

```
<owl:Class rdf:ID="Staff"/>
<owl:Class rdf:ID="Researcher"/>
<owl:Class rdf:ID="Academic"/>
```

- The fundamental taxonomic constructor for classes is rdfs:subClassOf.
  - Example:

```
<owl:Class rdf:ID="Researcher">
<rdfs:subClassOf rdf:resource="#Staff" />
...
</owl:Class>
```

Individuals

- Individuals enable us to describe members of a class.
- An individual is minimally introduced by declaring it to be a member of a class.

```
<owl:Thing rdf:ID="Postdoc" />
  <owl:Thing rdf:about="#Postdoc">
        <rdf:type rdf:resource="#Researcher"/>
  …
 </owl:Thing>
```

- rdf:type is an RDF property that ties an individual to a class of which it is a member.

Simple Properties

- *Properties* allow asserting general facts about the members of classes and defining specific facts about individuals.
- A property is a binary relation.
- Two types of properties are distinguished:
  - *datatype properties*, relations between instances of classes and RDF literals and XML Schema datatypes
  - *object properties*, relations between instances of two classes.

Simple Properties- Example

```
<owl:ObjectProperty rdf:ID=“hasSupervisor">
  <rdfs:domain rdf:resource="# PhDStudent "/>
  <rdfs:range rdf:resource="#Academic"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID=“demonstratedBy">
  <rdfs:domain rdf:resource="#Lab" />
  <rdfs:range rdf:resource="#PhDStudent" />
</owl:ObjectProperty>
```

Simple Properties- Example

Using DatatypeProperty

```
<owl:Class rdf:ID="hasName" />
<owl:DatatypeProperty rdf:ID="nameValue">
  <rdfs:domain rdf:resource="#PhDStudent" />
  <rdfs:range rdf:resource="&xsd; string"/>
</owl:DatatypeProperty>
```

Simple Properties- Constraints

```
<owl:Class rdf:ID=“PhDStudent">
  <rdfs:subClassOf rdf:resource=“#DeptMembers"/> <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasSupervisor"/>
        <owl:minCardinality
      rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
      </owl:Restriction>
  </rdfs:subClassOf>
...
</owl:Class>
```

Ontology design
Philosophical principles

1. Clarity: understandable not only for machines but also for humans.
2. Coherence: consistency of formal and informal layers of ontology (axioms vs. natural language documentation and labels).
3. Extendibility
4. Minimal coding bias: specification of ontologies should remain at the knowledge level (if it is possible) without depending on a particular symbol-level encoding.
5. Minimal ontological commitment: defining only those terms that are essential to the communication of knowledge consistent theory.
6. Proper sub-concept taxonomies

Ontology design
Technical principles

1. Define and use of naming conventions
   - Capitalisation: it is a common convention to begin concept names with capital, instance and property names with non-capital letters.
   1. Delimiters: common conventions are using space or “-” or writing names in CamelCase which eliminates the need for delimiters.
   2. Singular or plural: it is common to use the singular form in the concept names.
2. Scoping the ontology
   1. Introducing new entities: introduce a new concept only if it is significant for the problem domain.

Ontology design principles
Technical principles (cont.)

3. Optimal number of sub-concepts;
   - New concept or property value - concept or instance?
     - If it is meaningful to speak of a “kind of X” in the target domain i.e. the entity represents a set of something, make X a concept. Otherwise X should be an instant.
4. Document your ontologies;
5. Represent disjoint and exhaustive knowledge explicitly.

Ontologies as Part of a Solution to Other Problems

Some examples

Deep Question Answering with Watson

- Watson is a sophisticated question answering engine that finds answers to trivia/general knowledge questions for the Jeopardy! TV quiz.
- In the end, Watson did consistently outperform the human experts of the game.

Deep Question Answering with Watson

Further Reading OWL

- Industry-scale Knowledge Graphs: Lessons and Challenges

https://queue.acm.org/detail.cfm?ref=rss&id=3332266

- Protégé Tutorials:

1. https://www.emse.fr/~zimmermann/Teaching/SemWeb/Practice/ProtegeTutorial.pdf
2. https://protege.stanford.edu/publications/ontology_development/ontology101.pdf