Ontology

Tran Duc Minh

Outline

- Ontology
- Description Logic
- Sematics
- Knowledge Base
- Reasoning
- Semantic Web
- Semantic Web Rule Language
- Inductive Logic Programming

Ontology

- An ontology is a formal conceptualization of the world.
- An ontology specifies a set of **constraints**, which declare what should necessarily hold in any possible world.
- Any possible world should conform to the constraints expressed by the ontology.
- Given an ontology, **a legal world description** is a possible world satisfying the constraints.

Ontology languages (1)

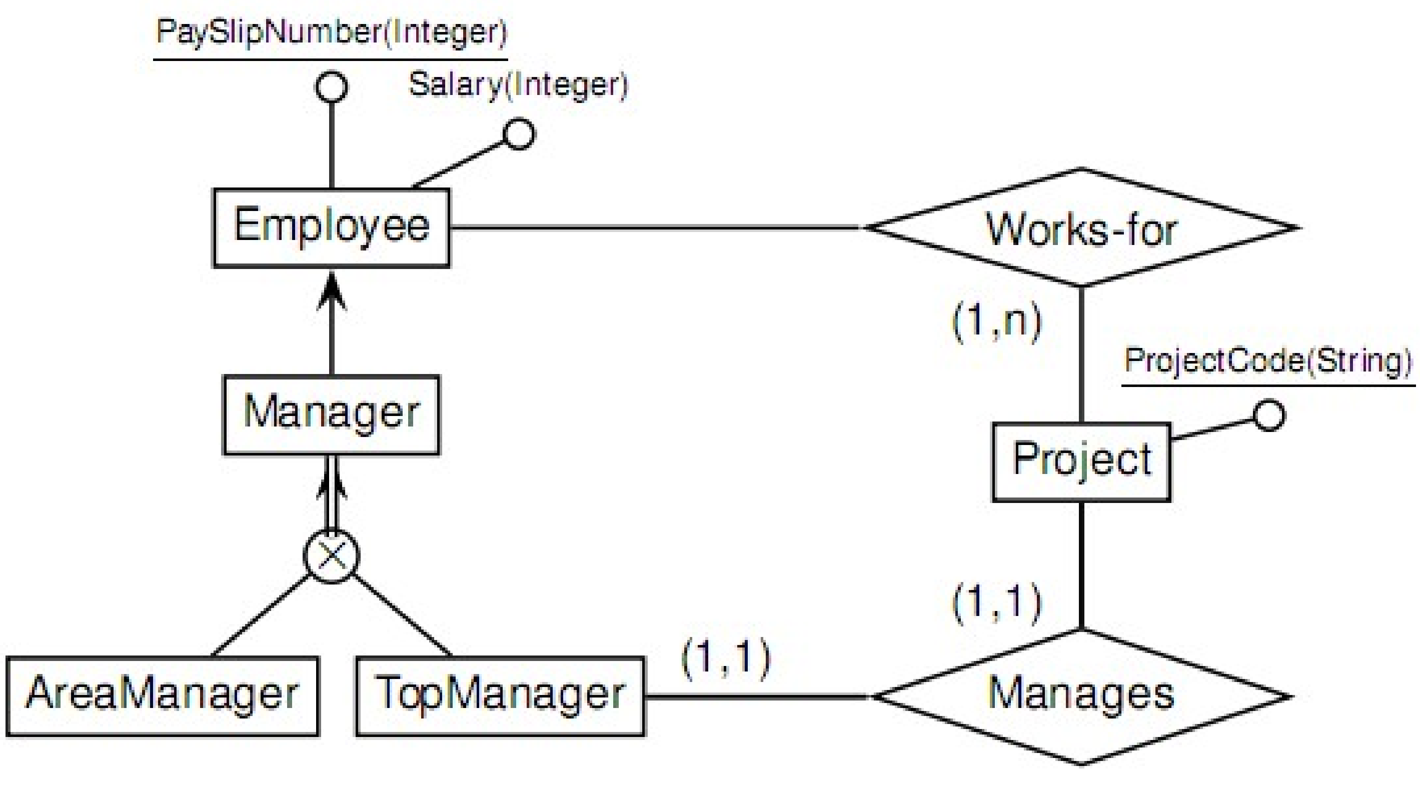
- An ontology language usually introduces **concepts** (aka classes, entities), **properties** of concepts (aka slots, attributes, roles), **relationships** between concepts (aka associations), and additional **constraints**.
- Ontology languages may be ***simple*** (e.g., having only concepts), ***frame-based*** (having only concepts and properties), or ***logic-based*** (e.g. Ontolingua and DAML+OIL).
  - Ontolingua is mechanism for writing ontologies in a canonical format, such that they can be easily translated into a variety of representation and reasoning systems.
  - DAML (DARPA Agent Markup Language)
  - OIL (the Ontology Inference Layer)

Ontology languages (2)

- Ontology languages are typically expressed by means of diagrams.
- **The Entity-Relationship conceptual data model** and **UML Class Diagrams** can be considered as ontology languages.

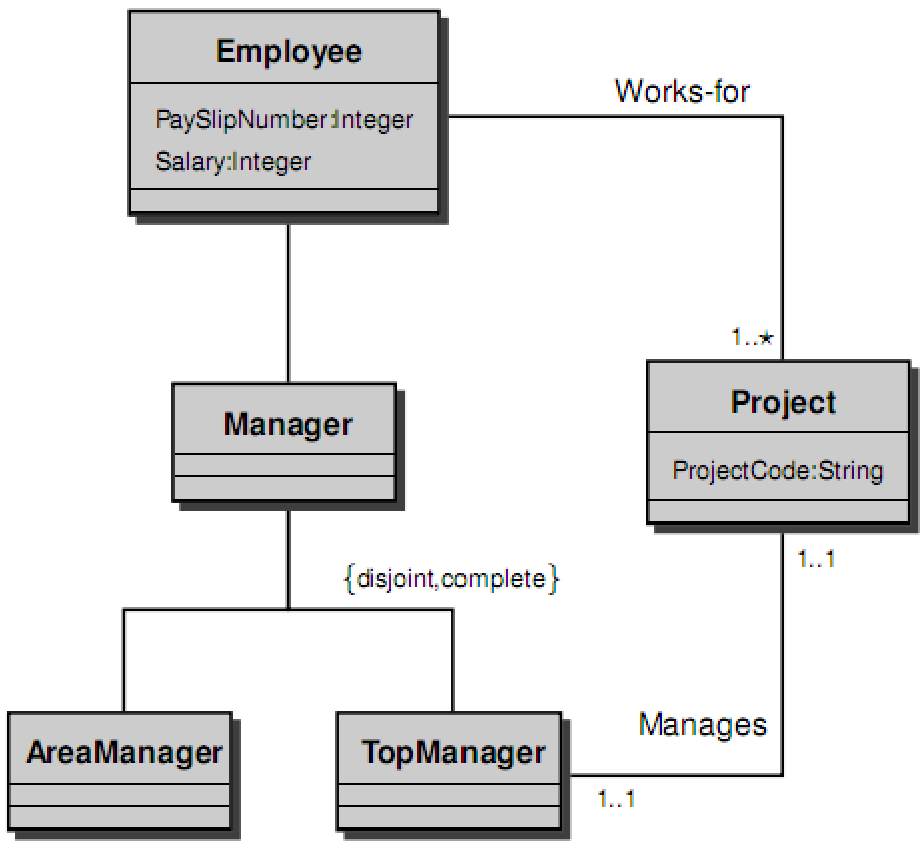
Ontology languages (3)

- For example: **Entity-Relationship Schema**



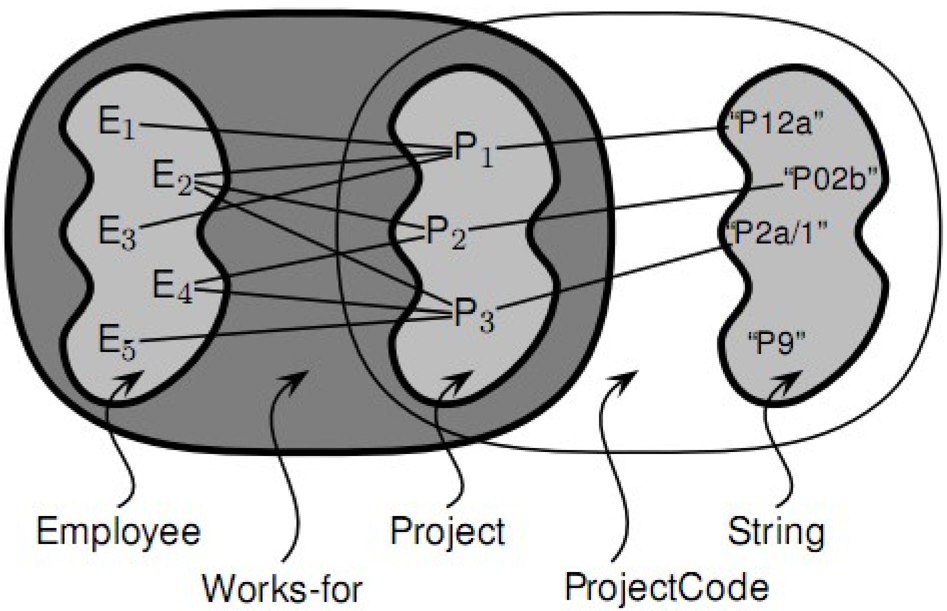
Ontology languages (4)

- **UML Class Diagram**

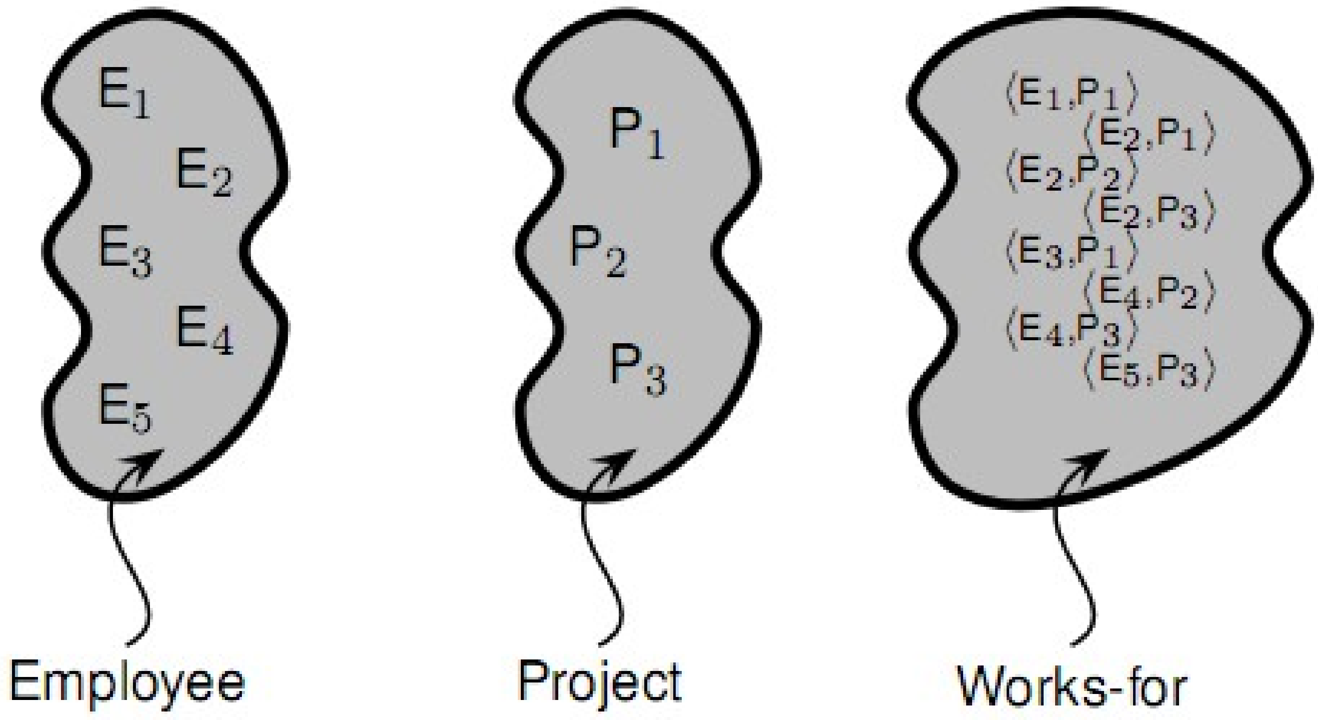


Meaning of basic constructs

- An **entity/class** is a set of instances
- An association (n-ary relationship) is a **set of pairs (n-tuples) of instances**
- An attribute is a **set of pairs of an instance and a domain element**



A world is described by sets of instances



The relational representation

Employee

employeeId
E_1
E_2
E_3
E_4
E_5

Project

projectId
P_1
P_2
P_3

String

anystring
"P12a"
"P02b"
"P2a/1"
"P9"
. . .

Works-for

employeeId	*projectId*
E_1	P_1
E_2	P_1
E_2	P_2
E_2	P_3
E_3	P_1
E_4	P_2
E_4	P_3
E_5	P_3

ProjectCode

projectId	*pcode*
P_1	"P12a"
P_2	"P02b"
P_3	"P2a/1"

Meaning of the diagram

$\mathsf{Works\text{-}for} \subseteq \mathsf{Employee} \times \mathsf{Project}$

$\mathsf{Manages} \subseteq \mathsf{TopManager} \times \mathsf{Project}$

$\mathsf{Employee} \subseteq \{e \mid \sharp(\mathsf{PaySlipNumber} \cap (\{e\} \times \mathtt{Integer})) \geq 1\}$

$\mathsf{Employee} \subseteq \{e \mid \sharp(\mathsf{Salary} \cap (\{e\} \times \mathtt{Integer})) \geq 1\}$

$\mathsf{Project} \subseteq \{p \mid \sharp(\mathsf{ProjectCode} \cap (\{p\} \times \mathtt{String})) \geq 1\}$

$\mathsf{TopManager} \subseteq \{m \mid 1 \geq \sharp(\mathsf{Manages} \cap (\{m\} \times \Omega)) \geq 1\}$

$\mathsf{Project} \subseteq \{p \mid 1 \geq \sharp(\mathsf{Manages} \cap (\Omega \times \{p\})) \geq 1\}$

$\mathsf{Project} \subseteq \{p \mid \sharp(\mathsf{Works\text{-}for} \cap (\Omega \times \{p\})) \geq 1\}$

$\mathsf{Manager} \subseteq \mathsf{Employee}$

$\mathsf{AreaManager} \subseteq \mathsf{Manager}$

$\mathsf{TopManager} \subseteq \mathsf{Manager}$

$\mathsf{AreaManager} \cap \mathsf{TopManager} = \emptyset$

$\mathsf{Manager} \subseteq \mathsf{AreaManager} \cup \mathsf{TopManager}$

Description Logic

- **Description logics (DLs)** is a family of knowledge representation languages that are widely used in ontological modelling.
- DLs provide one of the main underpinnings for the **Web Ontology Language** as standardized by the **W3C**.
- DLs are equipped with a formal semantics: a precise specification of the meaning DLs ontologies.
- The formal sematics allows human and computer systems
  - Exchanging DL ontologies without ambiguity as to their meaning.
  - Using logical deduction to infer additional information from the facts stated explicitly in an ontology
- The computation of inferences is called **reasoning**.

The Description Logic ALC

- **The set N_I of individual names** contains all names used to denote singuler entities in our domain of interest.
  - E.g: sun, moon, ….
- **The set N_C of concept names** contains names refer to types, categories, classes of entities, usually characterized by common properties.
  - E.g: Mammal, Country, ..
- **The set N_R of role names** contains names that denote binary relationships which may hold between individuals of a domain.
  - E.g: fatherOf, marriedWith, ...

The Description Logic ALC

- (Complex) concept expressions are constructed as:

$$C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

- A **TBox** (Terminological) is a set of statements of the form C ≡ D or C ⊑ D, where C and D are concept expressions. They are called general inclusion axioms.
- An **ABox** (Assertion) consists of statements of the form C(a) or R(a,b), where C is a class expression, R is a role, and a, b are individuals.

The Description Logic ALC

- ABox expressions:
  - Individual assignments. E.g: Father(phong)
  - Property assignments. E.g: hasWife(phong,thuan)
- TBox expressions:
  - Subclass relationships. E.g: Father $\sqsubseteq$ Man
  - Equivalence relationships. E.g: Mouse ≡ Mice
  - Conjunction. E.g: $C \sqcap D$
  - Disjunction. E.g: $C \sqcup D$
  - Negation. E.g: $\neg C$
  - Top. E.g: $\top$
  - Bottom. E.g: $\bot$
  - Property restrictions. E.g: $\forall R.C$ $\exists R.C$

The Description Logic ALC

- (Complex) concept expressions are constructed as:

$$C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

- A TBox (Terminological) is a set of statements of the form C ≡ D or C ⊑ D, where C and D are concept expressions. They are called general inclusion axioms.
- An ABox (Assertion) consists of statements of the form C(a) or R(a,b), where C is a class expression, R is a role, and a, b are individuals.

The Description Logic ALC

$$\mathtt{Human} \sqsubseteq \exists \mathtt{hasParent.Human}$$

$$\mathtt{Orphan} \sqsubseteq \mathtt{Human} \sqcap \forall \mathtt{hasParent.} \neg \mathtt{Alive}$$

$$\mathtt{Orphan(harrypotter)}$$

$$\mathtt{hasParent(harrypotter, jamespotter)}$$

The Description Logic SROIQ(D)

- The Description Logic SROIQ(D) includes:
  - SR = ALC + role chains
    - E.g: hasParent o hasBrother $\sqsubseteq$ hasUncle

      FOL: $\forall x\, \forall y\, (\exists z\, ((\text{hasParent}(x, z) \land \text{hasBrother}(z, y)) \rightarrow \text{hasUncle}(x, y)))$
    - includes S = ALC + transitivity
      - E.g: hasAncestor o hasAncestor $\sqsubseteq$ hasAncestor
    - includes SH = S + role hierarchies
      - E.g: hasFather $\sqsubseteq$ hasParent

The Description Logic SROIQ(D)

- The Description Logic SROIQ(D) includes:
  - **O** - nominals (closed concepts)
    - E.g: MyBirthdayGuests ≡ {lành, thơm, nụ}
  - **I** - inverse roles
    - E.g: hasParent ≡ hasChild^{-}

      Orphan ≡ ∀hasChild^{-}.Dead
  - **Q** - qualified cardinality restrictions
    - HappyFather ≡ ≥2 hasChildren.Female
    - Car ≡ =4 hasTyre. ⊤

The Description Logic SROIQ(D)

- Property characteristics: Properties can be declared to be
  - Transitive
    - E.g: hasAncestor R(a,b) and R(b,c) => R(a,c)
  - Symmetric
    - E.g: hasSpouse R(a,b) => R(b,a)
  - Asymmetric
    - E.g: hasChild R(a,b) => not R(b,a)
  - Reflexive
    - E.g: hasRelative R(a,a) for all a

The Description Logic SROIQ(D)

- Property characteristics: Properties can be declared to be
  - Irreflexive
    - E.g: parentOf   not R(a,a) for all a
  - Functional
    - E.g: hasHusband   R(a,b) and R(a,c) => b = c
  - InverseFunctional
    - E.g: hasHusband   R(a,b) and R(c,b) => a = c

The Description Logic SROIQ(D)

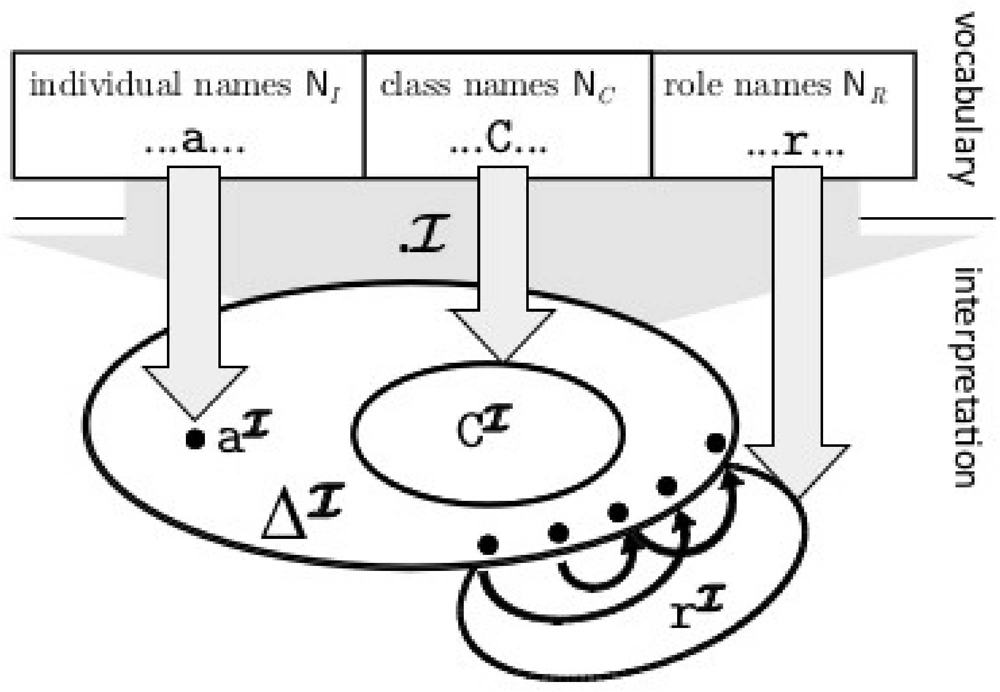
- The Description Logic SROIQ(D) includes:
  - **(D)** – datatypes
    - properties with datatype literals in second argument are called **data properties** or concrete roles
      - E.g 1: hasAge(john, "51"^^xsd:integer)
      - E.g 2: Teenager ≡ Person ⊓ ∃hasAge.(xsd:integer: ≥12 and ≤19)
    - note: this is not standard DL notation! It‘s really only used in OWL.

Semantics

- The semantics of description logic is defined in a model-theoretic way.
- **An interpretation** (normally denoted with **I**) provides:
  - A nonempty set $\boldsymbol{\Delta^I}$, called the ***domain*** which can be understood as the entire of individuals that **I** represents.
  - A function $\cdot^I$, called ***interpretation function*** which connects the vocabulary elements to Δ^I, by providing:

Semantics

- for each individual name **a** $\in$ $\mathbf{N_I}$ a corresponding individual $\mathbf{a^I}$ $\in$ $\mathbf{\Delta^I}$ from the domain.
- for each concept name **A** $\in$ $\mathbf{N_C}$ a corresponding set $\mathbf{A^I}$ $\in$ $\mathbf{\Delta^I}$ of the domain elements.
- for each role name **r** $\in$ $\mathbf{N_R}$ a corresponding set $\mathbf{r^I}$ $\subseteq$ $\mathbf{\Delta^I}$ **x** $\mathbf{\Delta^I}$ of ordered pairs of domain elements.



Semantics

- Interpretation function extends to concept expressions in the obvious way

$$
\begin{aligned}
&(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}\\
&(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}\\
&(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\\
&\{x\}^{\mathcal{I}} = \{x^{\mathcal{I}}\}\\
&(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\\
&(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}\\
&(\leqslant nR)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \leqslant n\}\\
&(\geqslant nR)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \geqslant n\}
\end{aligned}
$$

Semantics

- E.g 1:

$$\mathtt{Bird} \doteq \mathtt{Animal} \sqcap \forall \mathtt{SKIN}.\mathtt{Feather}$$

$$\begin{aligned}
&\Delta^{\mathcal{I}} = \{\mathtt{tweety}, \mathtt{goofy}, \mathtt{fea1}, \mathtt{fur1}\} \\
&\mathtt{Animal}^{\mathcal{I}} = \{\mathtt{tweety}, \mathtt{goofy}\} \\
&\mathtt{Feather}^{\mathcal{I}} = \{\mathtt{fea1}\} \\
&\mathtt{SKIN}^{\mathcal{I}} = \{\langle \mathtt{tweety}, \mathtt{fea1}\rangle, \langle \mathtt{goofy}, \mathtt{fur1}\rangle\} \\
&\Longrightarrow \quad \mathtt{Bird}^{\mathcal{I}} = \{\mathtt{tweety}\}
\end{aligned}$$

Semantics

- E.g 2:

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be defined by setting

- $\Delta^{\mathcal{I}} = \{a, b, c, d\}$;
- $A^{\mathcal{I}} = \{b, d\}$, $B^{\mathcal{I}} = \{c\}$;
- $r^{\mathcal{I}} = \{(a, b), (a, c)\}$, $s^{\mathcal{I}} = \{(a, b), (a, d)\}$.

Then

- $(\forall r.A)^{\mathcal{I}} = \{b, c, d\}$, $(\forall s.A)^{\mathcal{I}} = \{a, b, c, d\}$;
- $(\exists r.A \sqcap \forall r.A)^{\mathcal{I}} = \emptyset$, $(\exists s.A \sqcap \forall s.A)^{\mathcal{I}} = \{a\}$;
- $(\exists r.B \sqcap \exists r.A)^{\mathcal{I}} = \{a\}$, $(\exists r.(A \sqcap B))^{\mathcal{I}} = \emptyset$;
- $(\forall r.\neg A)^{\mathcal{I}} = \{b, c, d\}$, $(\forall s.\neg A)^{\mathcal{I}} = \{b, c, d\}$.

Knowledge Base

$\sum$ = (Tbox, Abox)

- E.g:

TBox:

- $Person \sqsubseteq Animal \sqcap Biped$
- $Woman \doteq Person \sqcap Female$
- $Mother \doteq Woman \sqcap \exists ParentOf.Person$
- $Parent \doteq Mother \sqcup Father$
- $Man \doteq Person \sqcap \neg Woman$
- $MotherWithoutDaughter \doteq Mother \sqcap \forall ParentOf.\neg Female$
- $GrandMother \doteq Woman \sqcap \exists ParentOf.Parent$

ABox:

- $GrandMother(Sally)$
- $(Person \sqcap Male)(John)$

Reasoning

- Sentences: Axioms or assertions
- **I** is a model for a sentence **S** iff **S** is true in **I**
- **I** is a model for a DL knowledge base **KB** iff it is a model for every sentence in **KB**
- Models of **KB** are denoted by **[KB]**
- **S** is entailed by **KB** , written **KB ⊨ S** iff **[KB]** ⊆ **[S]** (i.e. every model of K is a model of S)

Types of reasoning

KB a DL knowledge base

C and D are concepts

R is a role

a and b are individual names

- Instance checking: **KB $\models$ C(a)** or **KB $\models$ R(a, b)**
- Subsumption checking: **KB $\models$ C $\sqsubseteq$ D**
- Equivalence checking: **KB $\models$ C $\equiv$ D**
- Consistency checking: **KB $\not\models$ $\top \sqsubseteq \bot$**
- Concept satisfiability: **KB $\not\models$ C $\sqsubseteq \bot$**
- Disjoint concept: **KB $\models$ C $\sqcap$ D $\sqsubseteq \bot$**

Semantic Web

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation.“

Tim Berners-Lee - 2001

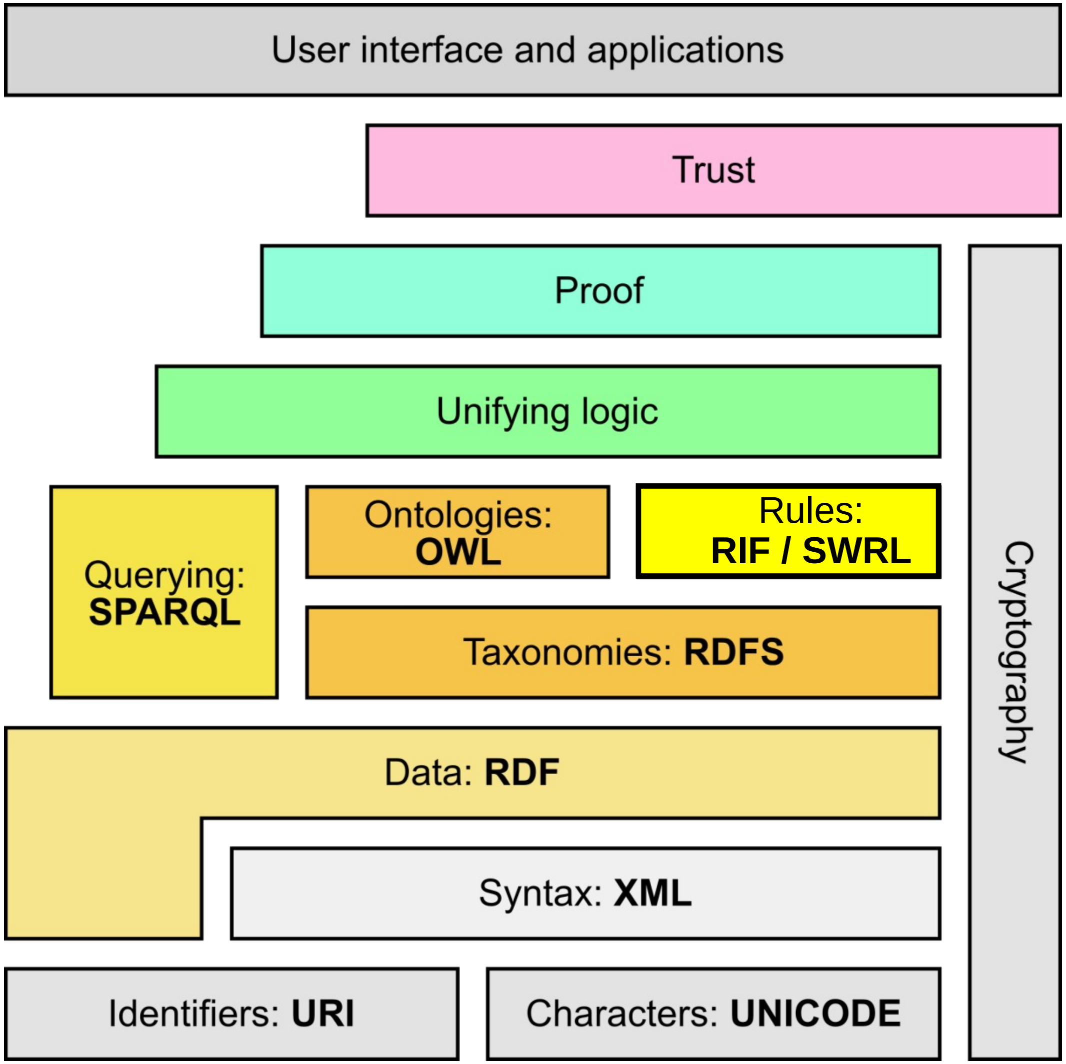
Semantic Web

- The content of the present Word Wide Web is nowadays only accessible and can be elaborated only by people.
- The Semantic Web is an enlargement of the WWW with semantic information that can be used by computers
- With the help of semantic information the content of pages could be processed automatically and computers could make inferences about a search.

Semantic Web

- The semantic web is not different from the www, is actually a developing part of it.
- The infrastructures and characteristics should be common
  - Use URI (Uniform resource Identifiers) addressing
  - Use protocols that a have a small and universally understood set of commands (like HTTP: Hypertext Transfer Protocol)

Semantic Web Vision



RESOURCE, URIs and NAMESPACES

- A resource is anything that has an identity
  - Digital (i.e an electronic document)
  - Physical (i.e. a book)
- A URI (Uniform Resource Identifier) is a character string that identifies a resource on the Web
  - URIs can follow different schemes
    - FTP (File Transfer Protocol)
    - HTTP (Hypertext Transfer Protocol
    - http://www.mysite.com/food.html

RESOURCE, URIs and NAMESPACES

- Namespaces are contexts, the domain of specific elements
- Namespaces are identified by a URI
- URIref: It is a URI with an optional fragment identifier attached to it, preceded by #

XML: Extensible Markup Language

- It is a general purpose markup Language for creating specific purpose mark-up languages
- With XML the single users can create their own tags (which is not possible with HTML)
- Differences between HTML and XML
  - HTML (Hypertext Markup Language)
    - Has a fixed set of tag
    - It is most frequently used to define the lay-out
    - Does not focus on the logical content or on the structur
  - XML
    - It is possible to personally define the tags
    - Tags reflect a content
    - The layout is defined in a separate document (stylesheet)

RDF: Resource Description Framework

- RDF is a general-purpose language for representing information in the web
  - Useful to represent metadata about Web resources
- RDF describes resources (Both abstract or concrete subjects) identifiable via an URI
- The syntax of RDF is based on XML
- RDF-documents are written as XML-documents with the tag rdf:RDF

RDF: Resource Description Framework

- A RDF-statement is described by a triple (Subject, Predicate, Object)
  - **Subject** is a resource
  - **Property** is a property of a resource
  - **Object** is the value of a property of a resource

RDF: Resource Description Framework

- E.g:
  - *A Fact: Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila*

```
<rdf:RDF>
  <rdf:Description about=
      "http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

How to represent above fact with DL ?

RDF: Resource Description Framework

- RDF gives a formalism for meta data annotation, and a way to write it down in XML, but it does not give any special meaning to vocabulary such as **subClassOf** or **type**

RDF Schema (RDFS)

- RDF Schema allows you to define vocabulary terms and the relations between those terms
  - it gives “extra meaning” to particular RDF predicates and resources
  - this “extra meaning”, or semantics, specifies how a term should be interpreted

RDF Schema (RDFS)

- RDF Schema terms
  - Class
  - Property
  - type
  - subClassOf
  - range
  - domain
- These terms are the RDF Schema building blocks (constructors) used to create vocabularies:
  - <Person,type,Class>
  - <hasColleague,type,Property>
  - <Professor,subClassOf,Person>
  - <Carole,type,Professor>
  - <hasColleague,range,Person>
  - <hasColleague,domain,Person>

RDF Schema (RDFS)

- :mary rdf:type :Person
  - Person(mary)
- :Mother rdfs:subClassOf :Woman
  - Mother $\sqsubseteq$ Woman
- :john :hasWife :Mary
  - hasWife(john,mary)
- :hasWife rdfs:subPropertyOf :hasSpouse
  - hasWife $\sqsubseteq$ hasSpouse
- :hasWife rdfs:range :Woman
  - $\top \sqsubseteq \forall$hasWife.Woman
- :hasWife rdfs:domain :Man
  - $\exists$hasWife.$\top \sqsubseteq$ Man

Web Ontology Languages (OWL)

- They are designed to define ontologies
- They are based on RDF and RDF-schema
- OWL (Web Ontology Language)

  http://www.w3.org/TR/owl-features/
  - It is an ontology description language
  - It is a standard language for the modeling of ontologies
  - Has additional vocabulary based on description logic

Web Ontology Languages (OWL)

- **OWL** was based on SHOIN
  - only simple role hierarchy, and unqualified Nrs
- **OWL 2** based on SHOIQ
  - ALC extended with transitive roles, a role box nominals, inverse roles and qualified number restrictions

Web Ontology Languages (OWL)

- Class/Concept Constructors

OWL Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer
maxCardinality	$\leqslant nP$	$\leqslant$1hasChild
minCardinality	$\geqslant nP$	$\geqslant$2hasChild

Web Ontology Languages (OWL)

- Ontology Axioms: OWL ontology is a mixed set of TBox and ABox axioms

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child

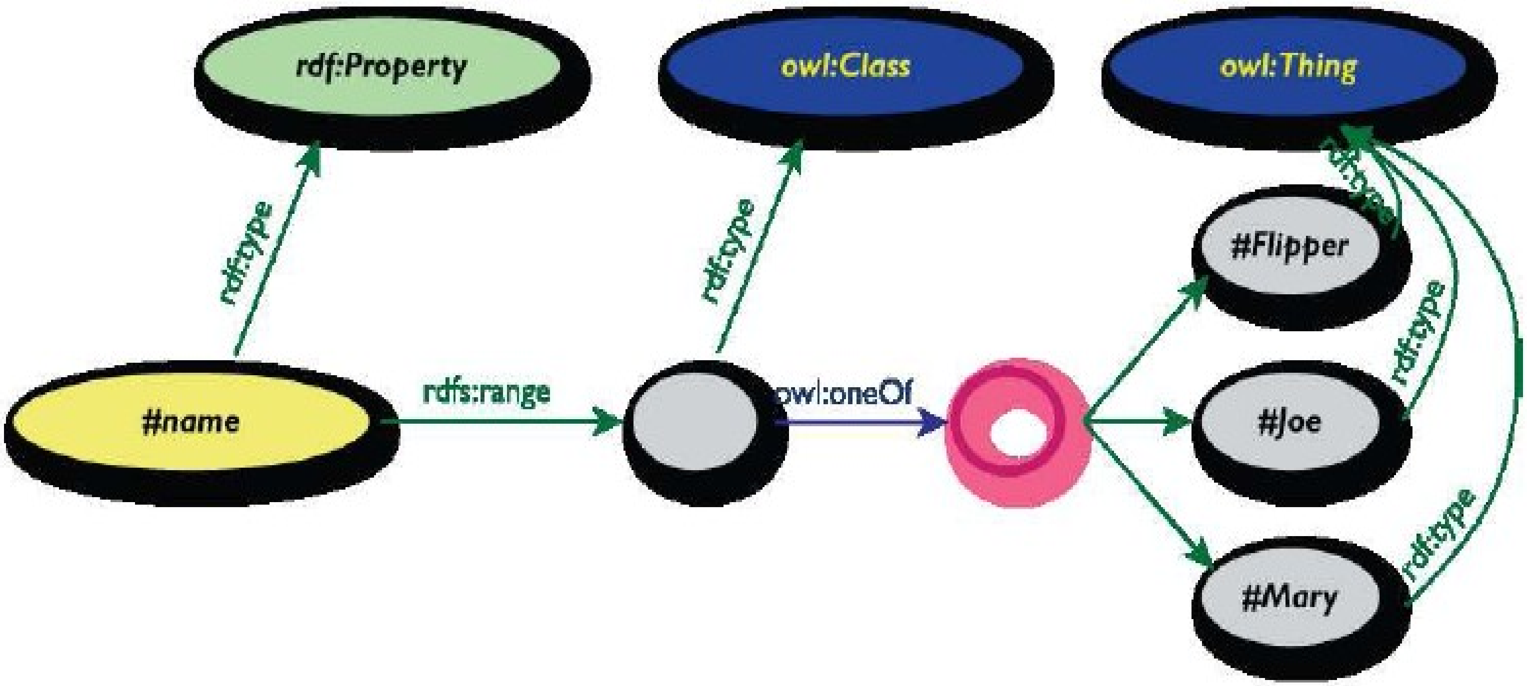
Web Ontology Languages (OWL)

E.g., Person $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor):

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

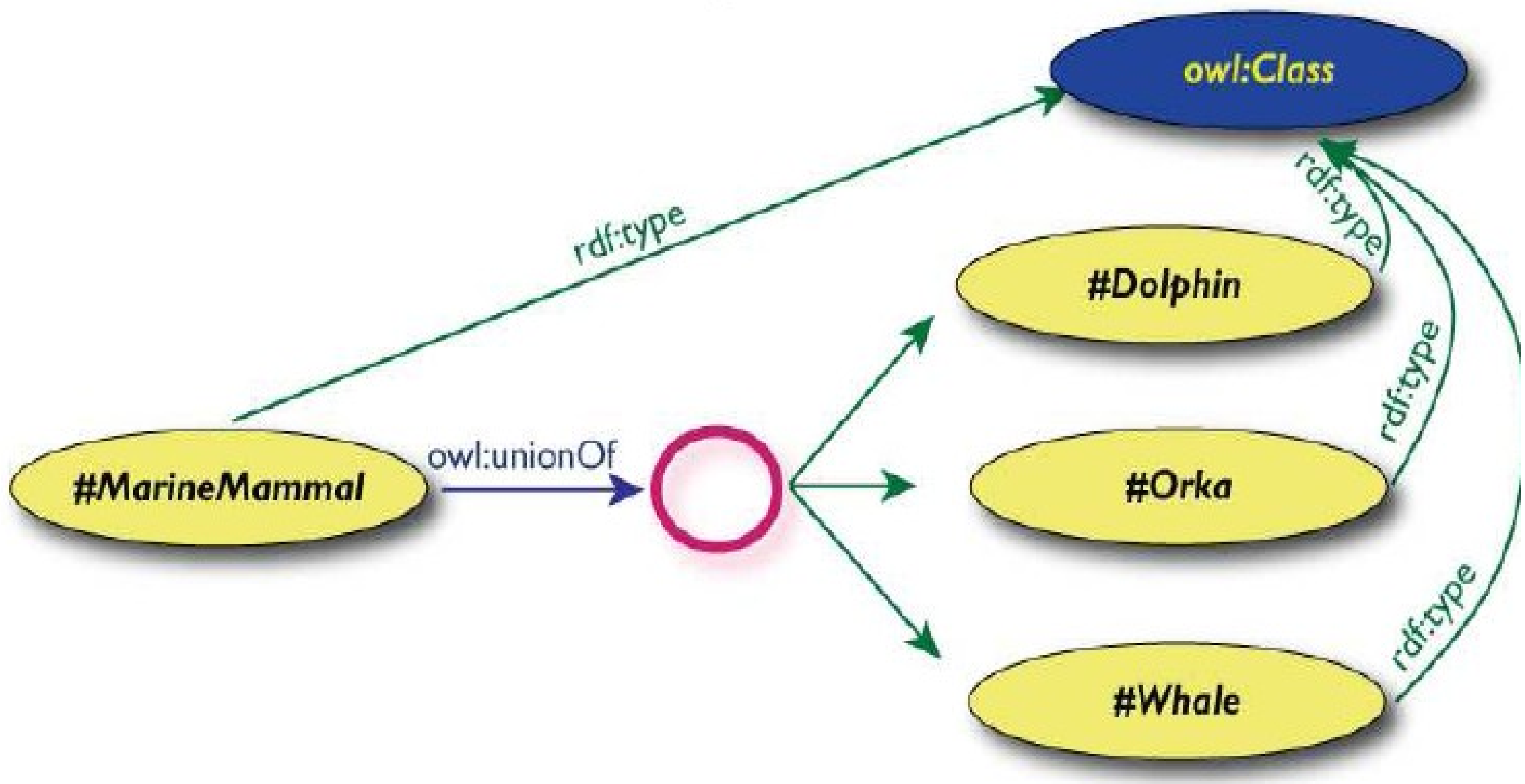
Web Ontology Languages (OWL)

- Definition of “name”



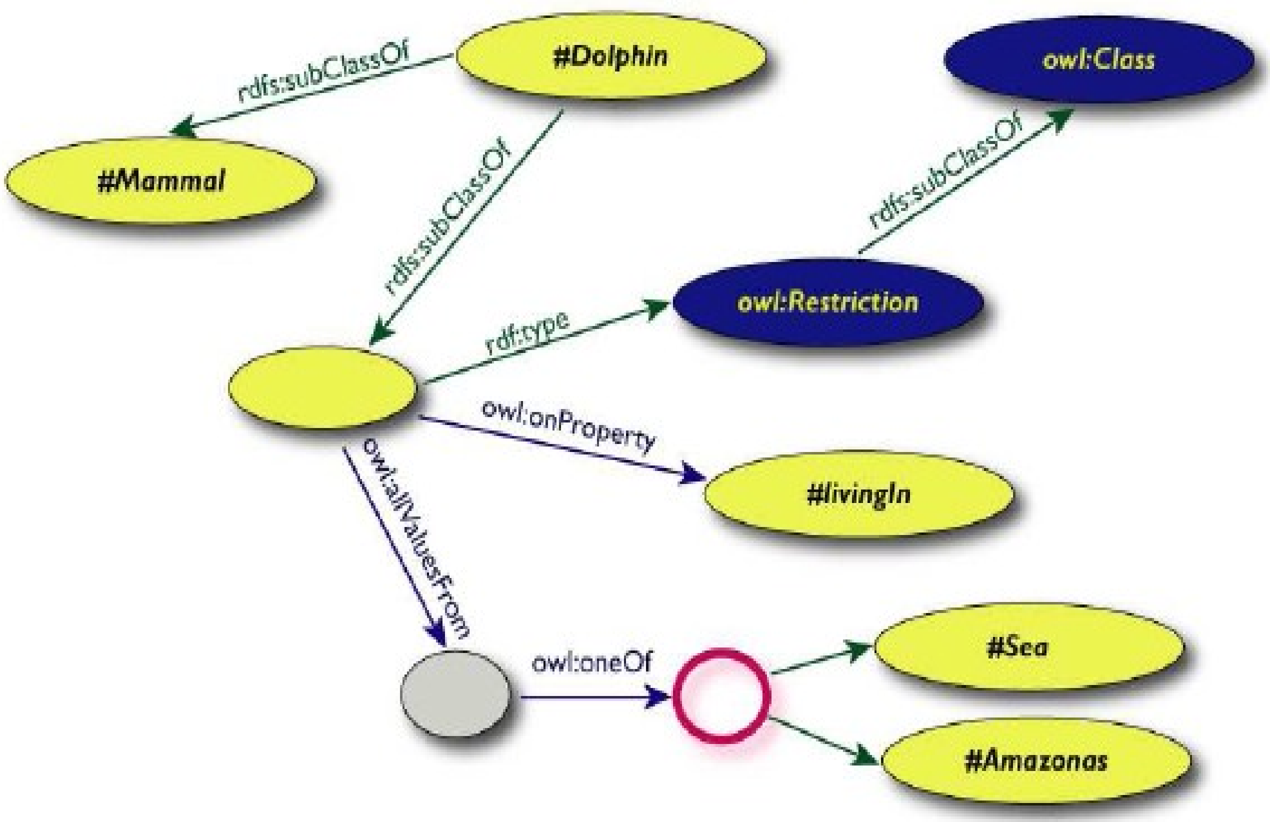
Web Ontology Languages (OWL)

- Definition of “marine mammal”



Web Ontology Languages (OWL)

- What is dolphin?



Semantic Web Rule Language (SWRL)

- The SWRL rule syntax follows:

  $a \Leftarrow b_1 \wedge b_2 \wedge \ldots. \wedge b_n$ where

  a: head (an atom) b_s: body (all atoms)

- A SWRL knowledge base (K) is defined as follows:

  $K = (\sum, P)$ where

  $\sum$ = Knowledge base of SROIQ(D)

  P = A finite set of rules

Semantic Web Rule Language (SWRL)

- SWRL atoms are defined as follows:

Atom <= C(i) | D(v) | R(i, j) | U(i, v) |

builtIn(p, v_1, ..., v_n) | i = j | i <> j

C: Class **D:** Data type

R: Object Property **U:** Data type Property

i, j: Object variable names or Object individual names

v_1, …, v_n: Data type variable names or Data type value names

p: Built-in names

Semantic Web Rule Language (SWRL)

- SWRL Semantics

Let **$I = (\Delta^I, \Delta^D, \cdot^I, \cdot^D)$** where

I: interpretation

Δ^I : Object Interpretation domain

Δ^D : Datatype Interpretation domain

$\cdot^I$: Object Interpretation function

$\cdot^D$: Datatype Interpretation function

$\Delta^I \cap \Delta^D = \varnothing$

such that $V_{IX} \Rightarrow P(\Delta^I)$ $V_{DX} \Rightarrow P(\Delta^D)$

V_{IX} : object variables V_{DX} : datatype variables

P : the powerset operator

Semantic Web Rule Language (SWRL)

- The following table shows Binding B(I) for the SWRL atoms:

SWRL Atoms	**Condition on Interpretation**
$C(i)$	$i^I \in C^I$
$R(i, j)$	$(i^I, j^I) \in R^I$
$U(i, v)$	$(i^I, v^D) \in U^I$
$D(v)$	$v^D \in D^D$
$builtIn(p, v_1, \ldots, v_n)$	$(v_1^D, \ldots, v_n^D \in p^D)$
$i = j$	$i^I = j^I$
$i \neq j$	$i^I \neq j^I$

Semantic Web Rule Language (SWRL)

- SWRL atoms in the antecedent are satisfied
  - if it is empty (trivially true)
  - or every atom of it is satisfied
- SWRL atom in the consequent is satisfied,
  - if it is not empty
  - and it is satisfied
- A rule is satisfied by an interpretation of I iff
  - every binding B(I) that satisfies the antecedent
  - B(I) satisfies the consequent

Semantic Web Rule Language (SWRL)

- A rule asserting a fast computer:

FastComputer(?c) <= Computer(?c) ^ hasCPU(?c, ?cpu) ^ hasSpeed(?cpu, ?sp) ^ HighSpeed(?sp)

- The above rule using only DL can be expressed as follows:

$$\text{Computer} \sqcap \exists \text{hasCPU}.\exists \text{hasSpeed}.\text{HighSpeed} \sqsubseteq \text{FastComputer}$$

Semantic Web Rule Language (SWRL)

- The translating of rules from SWRL to DL, depends on the number of variables based on shared variables between the consequent and antecedent.
- The number of variables shared between consequent and antecedent:
  - translating is possible, if 0 variable is shared, but at least one individual is shared
  - translating is possible, if 1 variable is shared
  - translating not possible, if 2 or more variables are shared

Semantic Web Rule Language (SWRL)

- A Rule in SWRL but not in DL:

  hasUncle(?x, ?y) <= hasParent(?x, ?z) ^ hasBrother(?z, ?y)

- The above rule cannot be translated into DL:
  - two different variables in the consequent