

Artificial Intelligence II

2013/2014 - Prof: Daniele Nardi, Joachim Hertzberg

Exercitation 5 - Roberto Capobianco

Description Logics and Semantic Web

Semantic Web (1)

- ▶ The World Wide Web is limited by its reliance on languages such as HTML;
 - ▶ HTML is focused on presentation rather than content.
 - ▶ e.g.:

HTML: <P>The Beatles was a popular band from Liverpool.</P>

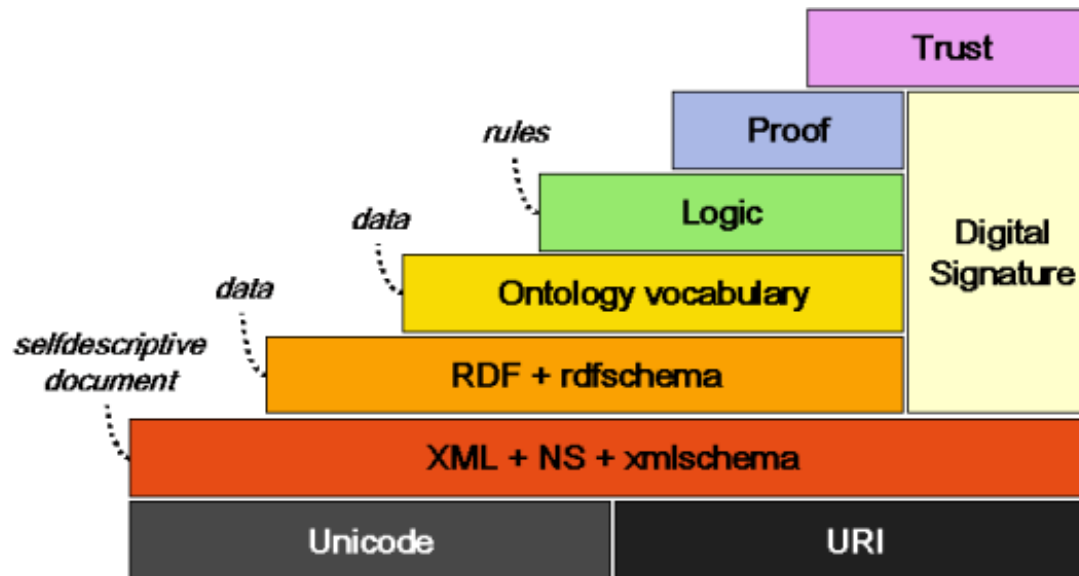
Man: Where is “The Beatles” band from?

Machine: ???????????

- ▶ The Semantic Web augments the existing data through a well formed and formal semantics (a meaning), so that information is machine understandable.

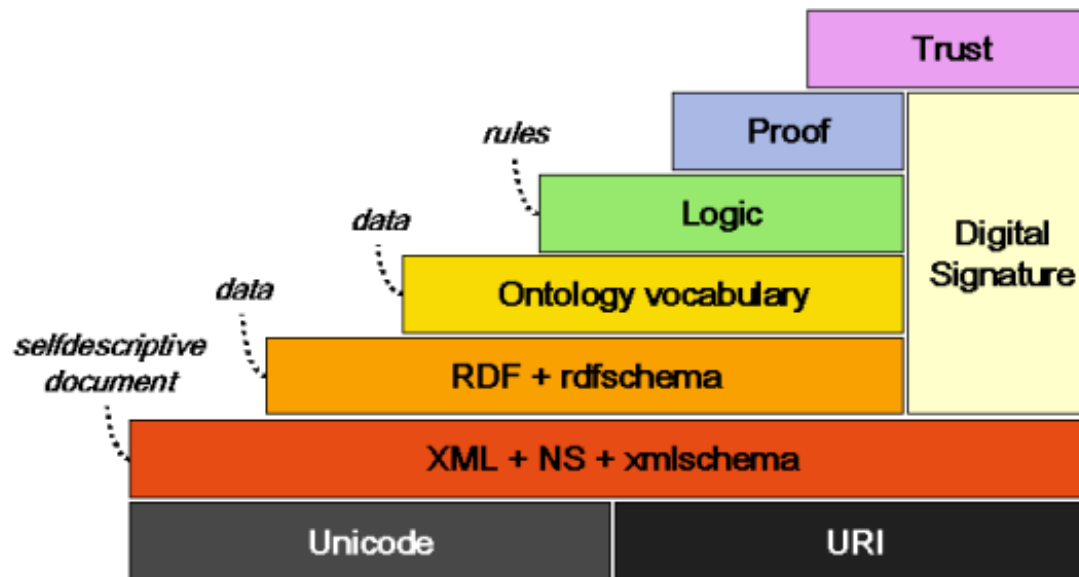
Semantic Web (2)

- ▶ Existing formalisms for knowledge representation;
- ▶ Existing web content languages (XML, RDFS);
- ▶ Existing languages for ontologies (DAML+OIL);



Existing Formalisms for KR

- ▶ Existing formalisms for knowledge representation;
- ▶ Existing web content languages (XML, RDFS);
- ▶ Existing languages for ontologies (DAML+OIL);



Description Logics

- ▶ \mathcal{AL} : **A**tributive language;
- ▶ \mathcal{C} : **C**oncept negation:
 - ▶ Syntax: $\neg C$;
 - ▶ Semantics: $\Delta^I \setminus C^I$;
- ▶ **T**ransitive role:
 - ▶ Syntax: $R \in \mathbf{R}^+$;
 - ▶ Semantics: $R^I = (R^I)^+$
- ▶ \mathcal{S} is defined as the family \mathcal{ALC} with transitive closed roles;
- ▶ (D) indicates the possibility to express *datatypes*, i.e., standard types like int, char, string:
`code(ROBERTO, "CPBRRT89P05")`

Concept Constructors (1)

- ▶ \mathcal{N} : Number restriction:

- ▶ Syntax:

- ▶ $\geq nR$;

- ▶ $\leq nR$;

- ▶ Semantics:

- ▶ $\{x \mid \#\{y.(x, y) \in R^I\} \geq n\}$;

- ▶ $\{x \mid \#\{y.(x, y) \in R^I\} \leq n\}$;

- ▶ Example:

Mother with many sons:

Female $\sqcap \geq 3$ *hasChild*

Concept Constructors (2)

- ▶ **Q: Qualifying number restriction:**

- ▶ **Syntax:**

- ▶ $\geq nR.C$;

- ▶ $\leq nR.C$;

- ▶ **Semantics:**

- ▶ $\{x \mid \#\{y.(x, y) \in R^I \text{ and } y \in C^I\} \geq n\}$;

- ▶ $\{x \mid \#\{y.(x, y) \in R^I \text{ and } y \in C^I\} \leq n\}$;

- ▶ **Example:**

Mother with at least two sons who are engineer:

Female $\sqcap \geq 3$ hasChild.Engineer

Concept Constructors and Functional Roles

- ▶ **O : Nominal:**

- ▶ Syntax: I ;

- ▶ Semantics: $I^I \subseteq I$ with $\#\{I^I\} = 1$;

- (singleton sets, consisting of one element of the domain)

- ▶ **F : Functional Role:**

- ▶ Syntax: F ;

- ▶ Semantics: $\forall a, b, c . \langle a, b \rangle \in R^I \wedge \langle a, c \rangle \in R^I \rightarrow b = c$

Role Constructors

▶ *I*: Inverse role:

- ▶ Syntax: R^- ;
- ▶ Semantics: $\{(x, y) \mid (y, x) \in R^I\}$
- ▶ Example:
The sons of engineers
 $\exists \textit{hasChild}^-.\textit{Engineer}$

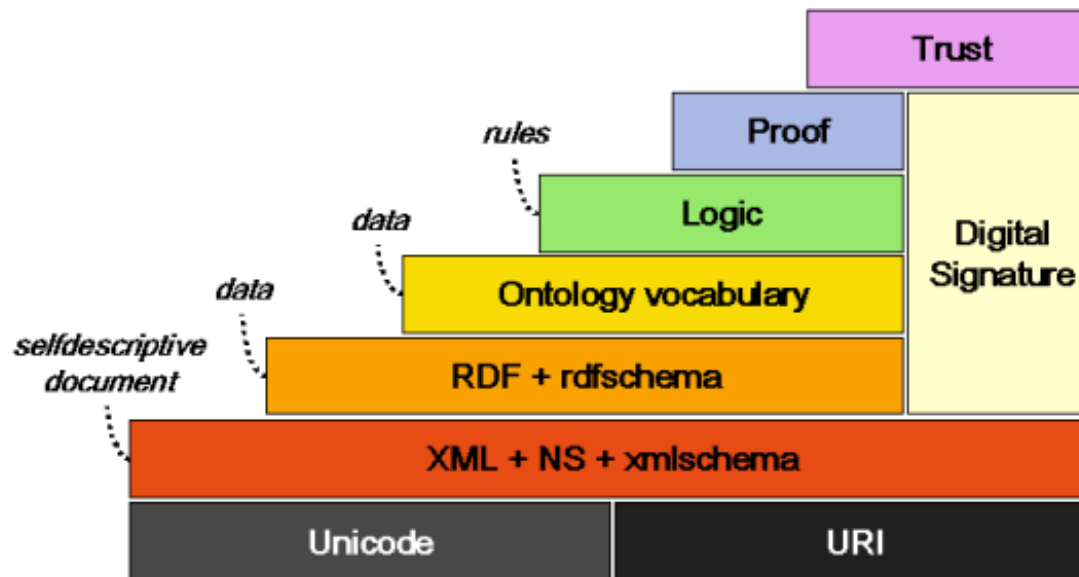
▶ *H*: Role hierarchy:

- ▶ Syntax: $R \sqsubseteq S$;
- ▶ Semantics: $R^I \subseteq S^I$;
- ▶ Example:

In graphs' theory, given the concept *Node* and roles *edge* and *connected* defined as $\text{Node} \times \text{Node}$, then $\textit{edge} \sqsubseteq \textit{connected}$

Existing Web Content Languages (1)

- ▶ Existing formalisms for knowledge representation;
- ▶ Existing web content languages (XML, RDFS);
- ▶ Existing languages for ontologies (DAML+OIL);

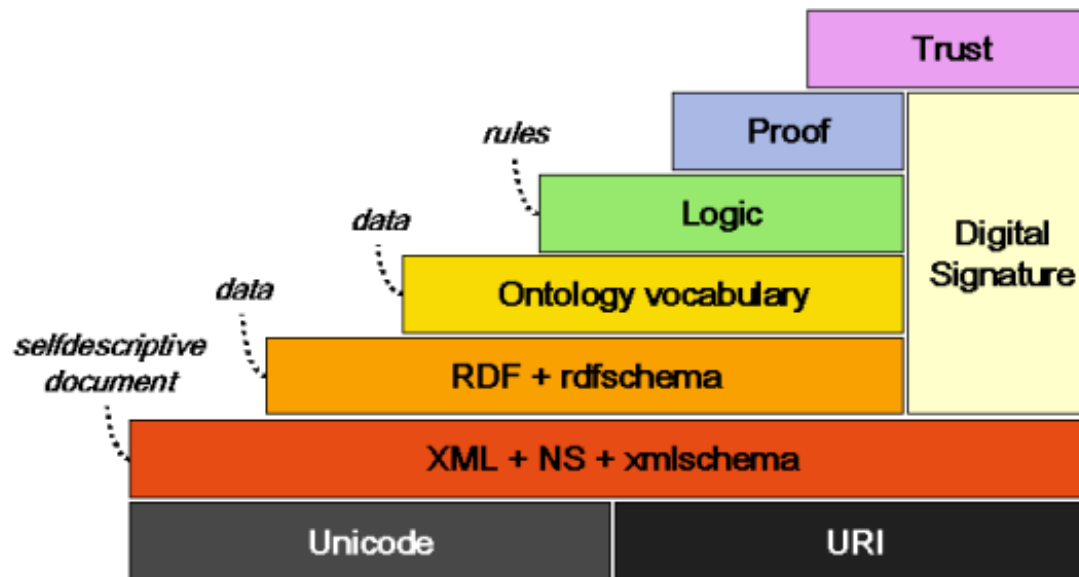


Existing Web Content Languages (2)

- ▶ **Extensible Markup Language (XML):** domain-specific markup language for documents;
 - ▶ Tree-like structure;
 - ▶ No ontology primitives;
 - ▶ Feasible for closed collaborations;
- ▶ **Resource Description Framework (RDF):** data model represented as a set of triples;
 - ▶ Graph-like structure;
 - ▶ Incomplete information;
- ▶ **Resource Description Framework Schema (RDFS):** provides a vocabulary to structure RDF and defines some ontology primitives (e.g. class, subclass, ...)
 - ▶ Weak resource descriptions;
 - ▶ Shallow reasoning;

Existing languages for ontologies (1)

- ▶ Existing formalisms for knowledge representation;
- ▶ Existing web content languages (XML, RDFS);
- ▶ Existing languages for ontologies (DAML+OIL);



Existing languages for ontologies (2)

- ▶ In 1999, **OIL** was the first language based on Description Logics, specifically designed to represent contents in the world wide web;
- ▶ **DAML** at the same time was a language for agent communications, which used RDFS.
- ▶ Their fusion (**DAML+OIL**) constitutes the first example of a language for the semantic web. Its expressivity is *SHIQ(D)*.

OWL DL

- ▶ **OWL** is a W3C recommendation since February 2004;
- ▶ **OWL 2** extends OWL and is a recommendation since October 2009;
- ▶ As in the DL, the Unique Name Assumption is not granted;
- ▶ Open World Assumption (vs databases);
- ▶ It is equivalent to *SHOIN(D)*;
- ▶ Worst case reasoning: NExpTime.



OWL vs DL

DL:	Concept	Role	Individual
OWL:	Class	Property	Instance

► Verbosity:

► DL: $Student \equiv Person \sqcap \geq 1 \text{ enrolledIn}$

► OWL:

```
<owl:Class rdf:ID="Student">  
  <owl:intersectionOf rdf:parsetype="Collection">  
    <owl:Class rdfs:about="Person" />  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="enrolledIn" />  
      <owl:minCardinality rdfs:datatype="&xsd;Integer">  
        1  
      </owl:minCardinality>  
    </owl:Restriction>  
  </owl:intersectionOf>  
</owl:Class>
```

OWL Lite

- ▶ The main limitations of OWL Lite are that it disallows:
 - ▶ Cardinality constraints other than 0-1;
 - ▶ Creation of enumerated concepts (oneof);
 - ▶ Creation of concepts on the basis of the existence of a particular slot-filler;
 - ▶ Creation of defined concepts;
- ▶ OWL Lite belongs to $\mathcal{SHIF}(D)$;
- ▶ Worst case reasoning: ExpTime.



OWL Full

- ▶ OWL Full includes OWL DL but cannot be defined in terms of the DL semantics;
- ▶ The main extension consists in the possibility of defining a concept as an individual of another concept;
- ▶ Queries over an ontology OWL Full are, in general, undecidable.



OWL Reasoners

▶ Pellet

- ▶ Written in Java
- ▶ Open-source
- ▶ <http://clarkparsia.com/pellet/>

▶ Racer

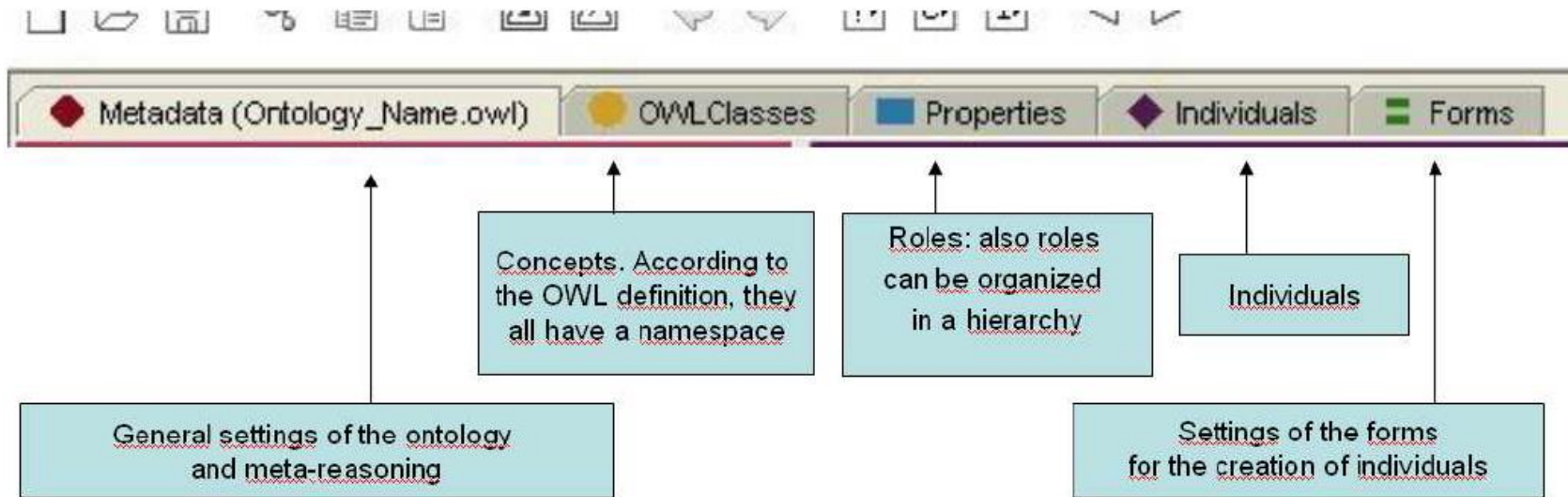
- ▶ Written in Lisp
- ▶ Commercial (free license for research purposes)
- ▶ <http://racer.sts.tuhh.de/>

▶ Other reasoners: <http://en.wikipedia.org/wiki/Reasoner>

Protege

- ▶ Protege is the most known editor for ontologies (use version 3.4!);
- ▶ <http://protege.stanford.edu/download/registered.html>
- ▶ Plug-in for Pellet reasoner;
- ▶ (ver. 3.4) Path for creation of an ontology with the description logics expressivity:
New Project → OWL/RDF Files → Next → Next → OWL DL → Logic View.

Use of Protege



Robot Scenario

- ▶ **What is a robot?**
 - ▶ A robot is modeled as a set of functionalities;
 - ▶ Robot functionality is complementary to the context in which the robot operates;
- ▶ **Notice:** It is a more general solution than modeling a robot as a collection of sensors. Each functionality can be achieved through different sensor types.

Primitive Concepts

- ▶ **Definition:** element of the ontology, in which only necessary conditions are specified.
 - ▶ Assertions over primitive concepts: populated like in a database (A-Box).
 - ▶ An individual X (e.g., LISA) belongs to a primitive concept A (e.g., Robot) if it is explicitly asserted (Robot(LISA)), or it is an individual of a sub-concept of A , e.g.:
 $B_Type \sqsubseteq \text{Robot}$
 $B_Type(\text{LISA})$.
 - ▶ **Open World Assumption:** if none of these two cases is specified, “ X is not an individual of A ” is not a correct conclusion, that means that X may still be an individual of A .

Primitive Concepts - Exercise

- ▶ A robot is a collection of functionalities. Functionalities are grouped in Actuation, Sensorial, Communicative classes.
- ▶ Three types of robots are defined (A,B,C).
 - ▶ A represents a robot having only actuation capabilities;
 - ▶ B has both actuation and sensorial capabilities;
 - ▶ C has all three types of functionalities.
- ▶ Soldatino is an A-type-robot, Lisa is a B-type, REDBACK is C-type.

Primitive Concepts - Solution

- ▶ Actuation, Sensorial, Communicative, Robot, A_Type, B_Type, C_Type are primitive concepts, while SOLDATINO, LISA and REDBACK are individuals.

Actuation \sqsubseteq Functionality

Sensorial \sqsubseteq Functionality

Communicative \sqsubseteq Functionality

Robot \sqsubseteq Functionality

A_Type \sqsubseteq Robot \sqcap Actuation

B_Type \sqsubseteq Robot \sqcap Actuation \sqcap Sensorial

C_Type \sqsubseteq Robot \sqcap Actuation \sqcap Sensorial \sqcap Communicative

A_Type(SOLDATINO)

B_Type(LISA)

C_Type(REDBACK)

Roles and Constraints

- ▶ With respect to relations of a database, roles are “oriented”, even if their internal representation is very similar.
- ▶ **Exercise:**
 - ▶ Inside the arena YELLOW I there are two victims Pippo and Pluto.
 - ▶ A robot possesses the functionality ImageAcquisition if it is able to communicate with another robot with the functionality ImageAcquisition.
 - ▶ Inside each arena there are at least two victims.
 - ▶ The stairs which are present inside an orange arena can not be spiral.

Roles and Constraints - Solution

contains(YELLOW1, PIPPO)

contains(YELLOW1, PLUTO)

Roles and Constraints - Solution

contains(YELLOW1, PIPPO)

contains(YELLOW1, PLUTO)

ImageAcquisition \sqsubseteq Communicative \sqcap \exists

connected.(Communicative \sqcap ImageAcquisition)

Roles and Constraints - Solution

contains(YELLOW1, PIPPO)

contains(YELLOW1, PLUTO)

ImageAcquisition \sqsubseteq Communitative \sqcap \exists
connected.(Communicative \sqcap ImageAcquisition)

Arena \sqsubseteq ≥ 2 containsVictim

Roles and Constraints - Solution

contains(YELLOW1, PIPPO)


contains(YELLOW1, PLUTO)

ImageAcquisition \sqsubseteq Communitative \sqcap \exists
connected.(Communicative \sqcap ImageAcquisition)

Arena \sqsubseteq ≥ 2 containsVictim

OrangeArena \sqsubseteq \forall ContainsStairs.(\neg Spiral)

Defined Concepts

- ▶ They are typically populated indirectly, individuals are retrieved through classification (subsumption).
 - ▶ Defined concepts do not exist in databases;
 - ▶ A defined concept has a specification of necessary and sufficient conditions (definition) for an individual to belong to the concept;
 - ▶ The reasoner “populates” the concept;
 - ▶ Primitive classes are defined if they have a defined sub-concept;
- ▶ *In Protegé, defined concepts are denoted with the symbol* 

Defined Concepts - Exercise

- ▶ **3 types of arena exist: red, orange and yellow;**
 - ▶ An arena is red iff it has rough terrain;
 - ▶ An arena is orange iff it has stairs but not rough terrain;
 - ▶ An arena is yellow iff it has more than two victims, but neither stairs nor rough terrain.

Defined Concepts - Solution

- ▶ *Arena, RedArena, RoughTerrain, Stair, Victim* are primitive concepts.
- ▶ *contains* is a role $Arena \times T$
- ▶ *containsVictim* is a role $Arena \times Victim$

$RedArena \equiv Arena \sqcap \exists \text{ contains. } RoughTerrain$

$OrangeArena \equiv Arena \sqcap \neg \exists \text{ contains. } RoughTerrain \sqcap \exists \text{ contains. } Stair$

$\text{containsVictim} \sqsubseteq \text{contains}$

$YellowArena \equiv Arena \sqcap \neg \exists \text{ contains. } (RoughTerrain \sqcup Stair) \sqcap \geq 2 \text{ containsVictim}$

Defined Concepts - Exercise

- ▶ A team of robots contains one or more robots. The team **BARNEYTEAM** contains robots **SOLDATINO** (A-type) and **LISA** (B-type). The team **BARNEYTEAM** is into the yellow arena **YELLOW I**. The C-type robot **REDBACK** is inside the red arena **RED I**. Retrieve all the robots who belong to the team **BARNEYTEAM**.

Defined Concepts - Solutions

Team(BARNEYTEAM)

YellowArena(YELLOW1)

RedArena(RED1)

member(BARNEYTEAM, SOLDATINO)

member(BARNEYTEAM, LISA)

contains(YELLOW1, BARNEYTEAM)

contains(RED1, REDBACK)

MembersBarneyTeam $\equiv \exists \text{member}^- . \text{BARNEYTEAM}$

Reasoning with Concepts (1)

- ▶ **Classification** is the process of reasoning only over concepts (Tbox). Applies subsumption over concepts and builds an inferred taxonomy (adding new subset relationships).

OWL Menu → "Classify Taxonomy".

- ▶ **Satisfiability** of concepts verifies if a concept admits at least one individual; it is implemented as the non-subsumption (where the subsumer is \perp).

Right click on the concept and select "Check Concept Consistency";

OWL menu → "Check Consistency": verifies satisfiability for all the concepts.

Reasoning with Concepts (2)

- ▶ **Equivalence between concepts**

Can be checked using the definition: Run classification and verify whether one subsumes the other and viceversa.

- ▶ **Disjointness between concepts**

Apply the definition: define the intersection concept and verify if it is not consistent.

- ▶ **Protege has also some API JAVA, which allow to operate on the ontology from a JAVA program.**

Reasoning over Individuals

- ▶ Instance checking verifies whether an individual is instance of a concept
- ▶ *Right click on the individual in the "Individuals" tab → "Compute types" (instance checking of the individual on all concepts);
OWL menu → "Compute inferred types": instance checking of all individuals on all concepts.*
- ▶ Retrieval finds all individuals belonging to a concept.
Right click on the concept rightarrow "Compute individuals belonging to class".
- ▶ Consistency check (Abox)
It is executed automatically while making assertions.

Which Robot for Which Arena?

- ▶ **Given the definition:**

$\text{RescueTeam} \equiv \text{Team} \sqcap \exists \text{member.Exploration} \sqcap \exists \text{member.}$

$\text{Localization} \sqcap \exists \text{member.Mapping} \sqcap \exists \text{member.VictimDetection}$

- ▶ “A Team is a RescueTeam if it can perform exploration, mapping, localization, victim detection”.
- ▶ 3 Actuation functionalities exist: Basic, OnStairs, OnEachTerrain, stating in which arenas a robot is able to move.
- ▶ Specify that a robot has the capability Mobility if it can move in the arena in which its team is.
- ▶ Specify that a Team, in order to be a RescueTeam must be into an arena and at least one robot of the team must be able to move.

Which Robot for Which Arena? - Solution

Basic \sqsubseteq Actuation

OnStairs \sqsubseteq Basic

OnEachTerrain \sqsubseteq OnStairs

RedMobility \equiv OnEachTerrain \sqcap (\exists member $^-$.(\exists contains $^-$.RedArena))

OrangeMobility \equiv OnStairs \sqcap (\exists member $^-$.(\exists contains $^-$.OrangeArena))

YellowMobility \equiv Basic \sqcap (\exists member $^-$.(\exists contains $^-$.YellowArena))

Mobility \equiv RedMobility \sqcup OrangeMobility \sqcup YellowMobility

Mobility \sqsubseteq Actuation

TeamRescue \equiv ... \sqcap \exists contains $^-$.Arena \sqcap \exists member.Mobility

Teams of Robots

- ▶ **Query the system asking:**
 - ▶ Which teams are able to move in the red arena;
 - ▶ Which arena can face the team BARNEYTEAM;
 - ▶ Whether RedArena and YellowArena are disjoint concepts;
- ▶ **Remember that a team can face a certain characteristic of the arena if at least one robot is able to do it.**

Teams of Robots – How to Proceed

- ▶ Create the concept

$\text{TeamForRedArena} \equiv \exists \text{member.RedMobility}$

- ▶ and ask for retrieval.

- ▶ It is an instance checking query for **BARNEYTEAM** on the concept **RescueTeam**. Insert the Team inside the different arenas and perform the instance checking operation.

- ▶ Create the concept

$\text{Intersection} \equiv \text{RedArena} \sqcap \text{YellowArena}$

- ▶ and verify the consistency.

Exercises

- ▶ May an arena $W1$ be classified autonomously as an OrangeArena?
- ▶ If yes, how? If no, why?
- ▶ If I want to allow that the entry and exit door of an arena may be a unique door, should I modify the defined constraint in the previous slides ?
- ▶ If the definition of RescueTeam was
 $\text{TeamRescue} \equiv \text{Team} \sqcap \exists \text{member.}(\text{Exploration} \sqcap \text{Localization} \sqcap \text{Mapping} \sqcap \text{VictimDetection})$
- ▶ What would the difference with the previous definition be?
- ▶ Ask for the content of the arena YELLOW I.

Thank you!