Intelligent Systems

ONTOLOGY WEB LANGUAGE AND PROTÉGÉ

Intelligent Systems Lab

Universidade do Minho

Tiago Oliveira, Paulo Novais and José Neves

Summary

- **Ontologies**
  - What is an Ontology?
  - Developing Ontologies – Considerations
  - Ontology Web Language
- **Protégé**
  - Installing Protégé
  - Creating a new Ontology
  - Entities
  - Ontology Definition
  - Practical Example – Family Ontology
- **Bibliography**

ONTOLOGIES

Ontologies

- What is an Ontology?

A formal explicit description of:

Concepts	Properties	Restrictions
• In a domain of discourse; • classes (sometimes called concepts).	• Of each concept describing various features and attributes of the concept; • Slots (sometimes called roles or properties).	• On slots (facets (sometimes called role restrictions).

Ontologies

- Developing Ontologies - Considerations

Why develop an Ontology?

- To share common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyse domain knowledge.

Ontologies

- Developing Ontologies - Considerations

What questions should the Ontology answer?

- Ontology as a Knowledge Base.

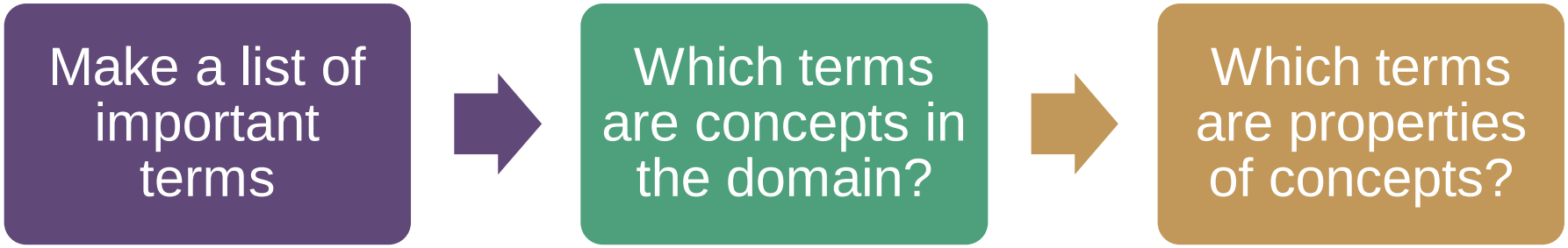
What is the underlying domain?

- Limiting a domain is complex.
- May change during the Ontology development process.

Ontologies

- Developing Ontologies - Considerations

Preparing the Domain



Ontologies

- Developing Ontologies - Considerations

Standards for Class Development

- Classes should be either **singular** or plural.
  - *(ex: ~~Man is a Persons~~)*
- *Exception:* When a concept only has one form.
- **Class** represents the concept, not the word used.
  - *(ex: Pen and Ballpoint refer to the same concept)*
- Synonyms and Translations may be added as annotations to classes.

Ontologies

- Developing Ontologies - Considerations

Define Hierarchy of Classes

- Hierarchies are created by **is-a** and **kind-of** relations.
  - *(ex: Man is-a Person; Dog kind-of Animal)*
- Subclasses should be at the same level of generalization.
  - *(ex: Caucasian and Portuguese should not be siblings)*
- Number of siblings on a class should be between 2 and 12.

Ontologies

- Developing Ontologies - Considerations

Define Hierarchy of Classes

- Hierarchy should be Transitive.
  - *(ex: If C is-a B and B is-a A then C is-a A)*
  - *(ex: Dog is-a Pet and Pet is-a Animal, then Dog is-a Animal)*
- Multiple Inheritance: a class may be a cubclass of several classes.
  - *(ex: Chicken is-a Animal and Chicken is-a Food)*
- Disjoint Classes: when classes should not share instances.
  - *(ex: Man and Woman are commonly Disjoint)*

Ontologies

- Developing Ontologies - Considerations

Is X a new Class or a Subclass?

- Does X have properties which its superclass does not?
- Does X have different restrictions from its superclass?
- Is X involved in relations in which its superclass is not?
- If yes to any or yes to all, than X is a new class.
- Otherwise …. *COMMON SENSE HELPS!*

Ontologies

- Developing Ontologies - Considerations

Is X a Class or a Property?

- Is X an important term in the Domain?
- It depends on the Domain!
- Properties usually represent the most generic concepts?
- Again …. *COMMON SENSE HELPS!*

Ontologies

- Developing Ontologies - Considerations

Is X a Class or an Instance?

- It depends on the Granularity of the representation.
- It depends on the application of the Ontology.
- May X have instances? Then it is a Class.
  - *(ex: Maria cannot have instances - Instance)*
  - *(ex: Car can have instances (BMW) - Class)*

Ontologies

- Ontology Web Language (OWL)

- Standard developed by the World Wide Web Consortium (W3C).

- OWL 2

- This formalism facilitates machine interpretability and is built upon other technologies such as XML, RDF and RDF-schema.

Ontologies

- Ontology Web Language (OWL)

❑ Sublanguages of OWL 2:

PROTÉGÉ

Protégé

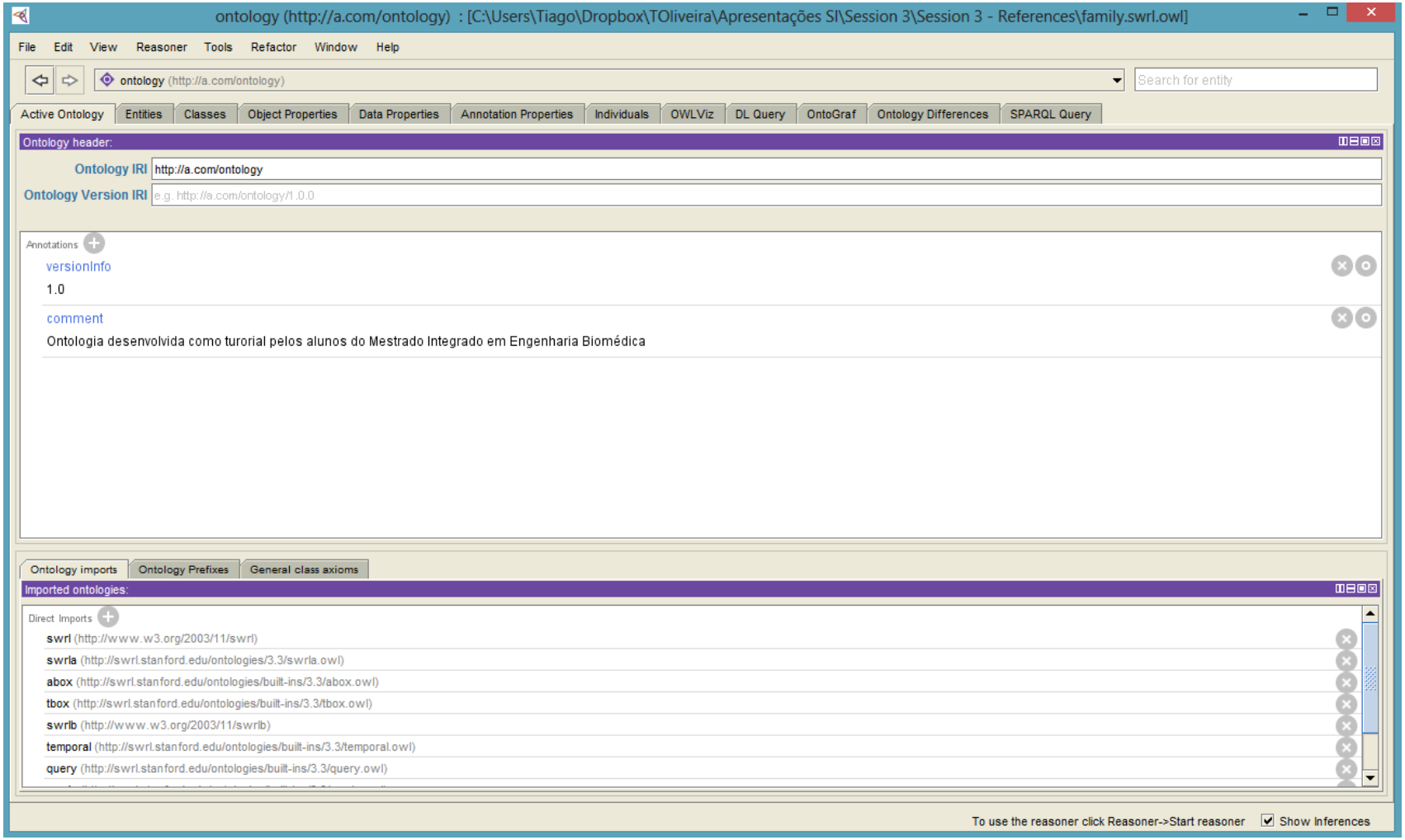
- Installing Protégé
  - Go to: http://protege.stanford.edu/
  - Then go to Downloads
  - Protégé 4.1 release
    - Platform Independent
    - Select the OS version
    - Include the JVM, if **necessary**.

Protégé

- Installing Protégé
  - Launch Protégé.
  - Select “Create New OWL ontology”.
  - Choose the URI.
  - Choose a directory to store the Ontology.

Protégé

- Creating a new Ontology



Protégé

- Creating a new Ontology

$$O=(C,H,I,R,P,A)$$

O O C_c R P C_i
H I I
A

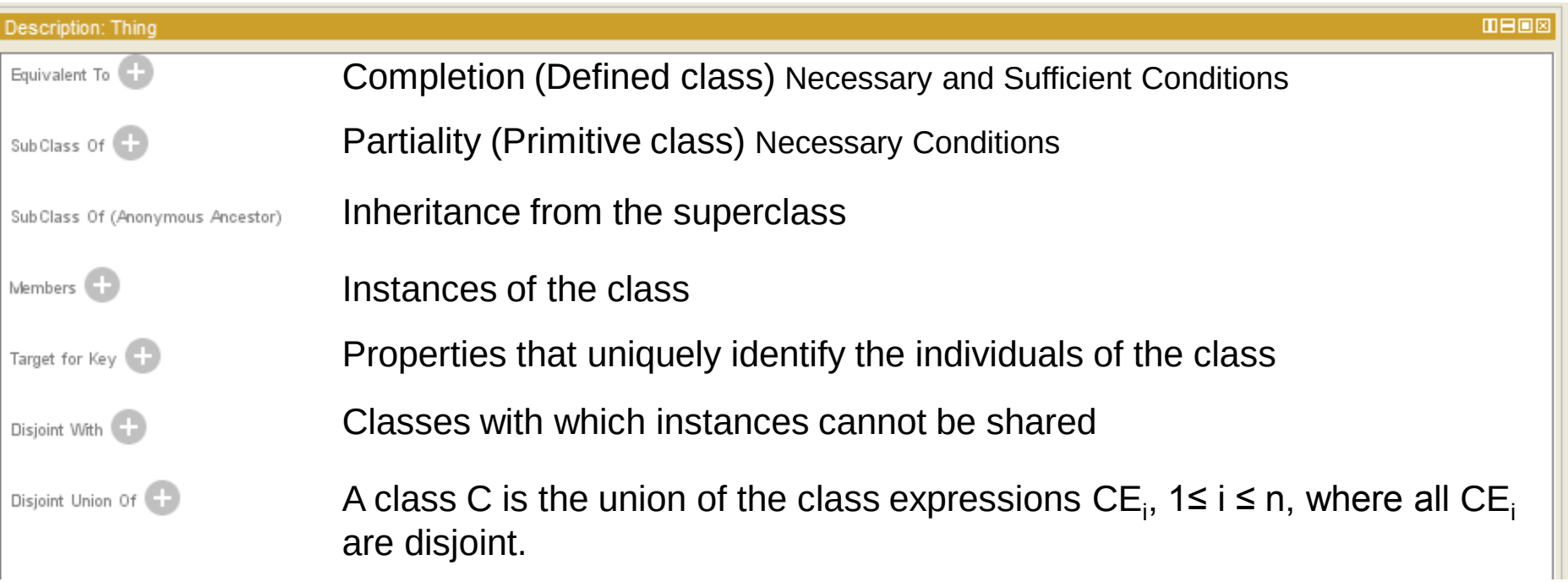
Protégé

- Entities
  - Classes/ Concepts
    - Restrictions, Enumerations and Partitions
  - Individuals/Instances
  - Properties/Relations (Object and Data)

Protégé

- Entities

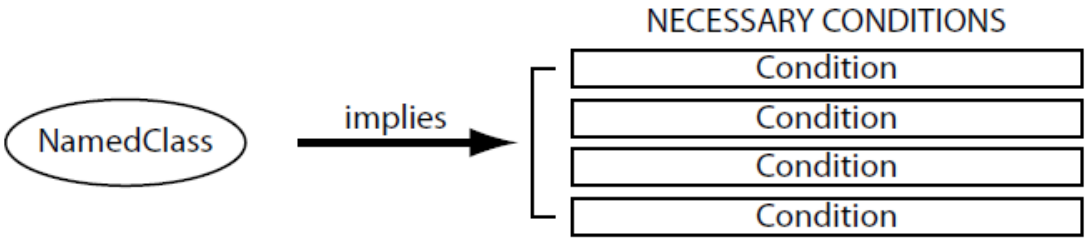
Classes



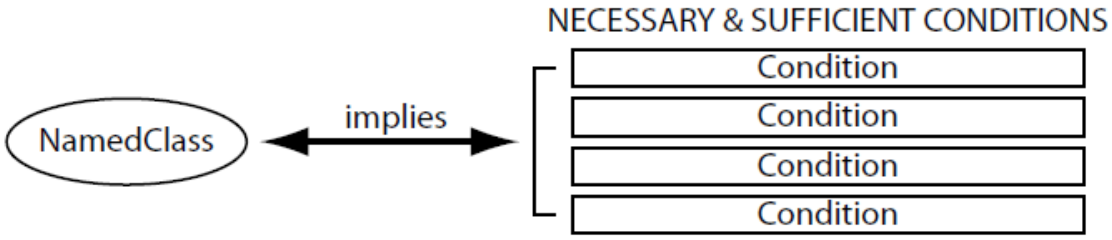
Protégé

- Entities

Classes



If an individual is a member of 'NamedClass' then it must satisfy the conditions. However if some individual satisfies these necessary conditions, we cannot say that it is a member of 'Named Class' (the conditions are not 'sufficient' to be able to say this) - this is indicated by the direction of the arrow.

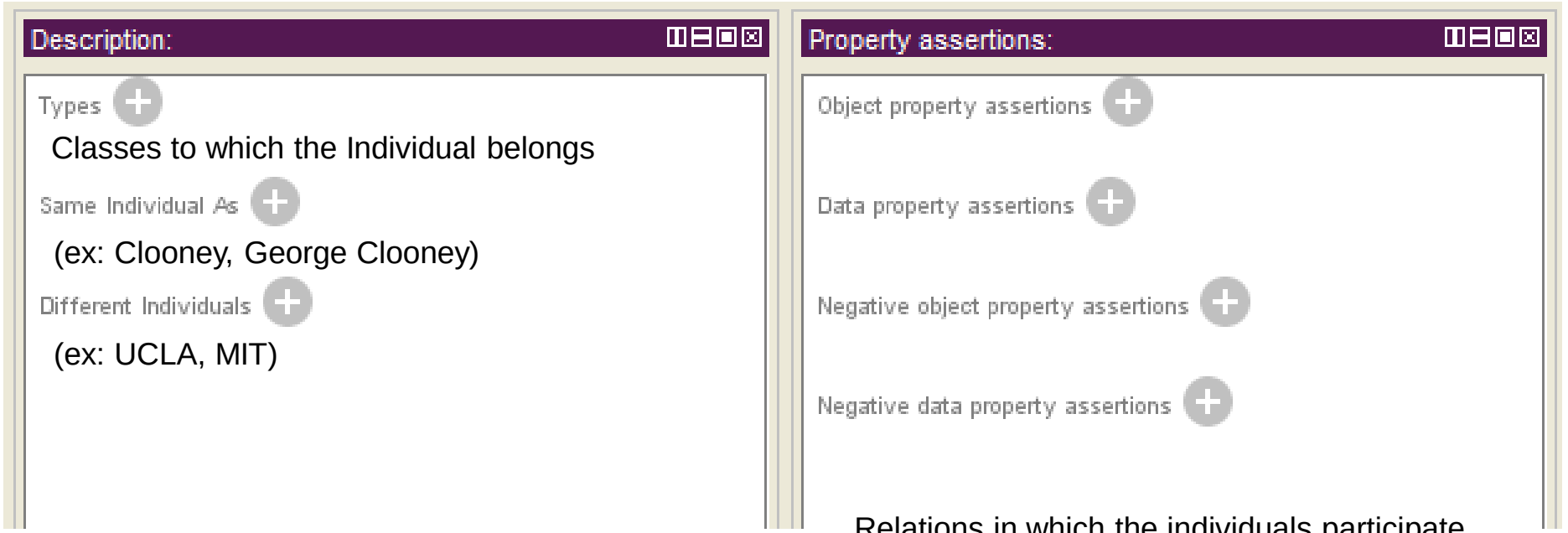


If an individual is a memeber of 'NamedClass' then it must satisfy the conditions. If some individual satisfies the condtions then the individual must be a member of 'NamedClass' - this is indicated by the double arrow.

Protégé

- Entities

Individuals

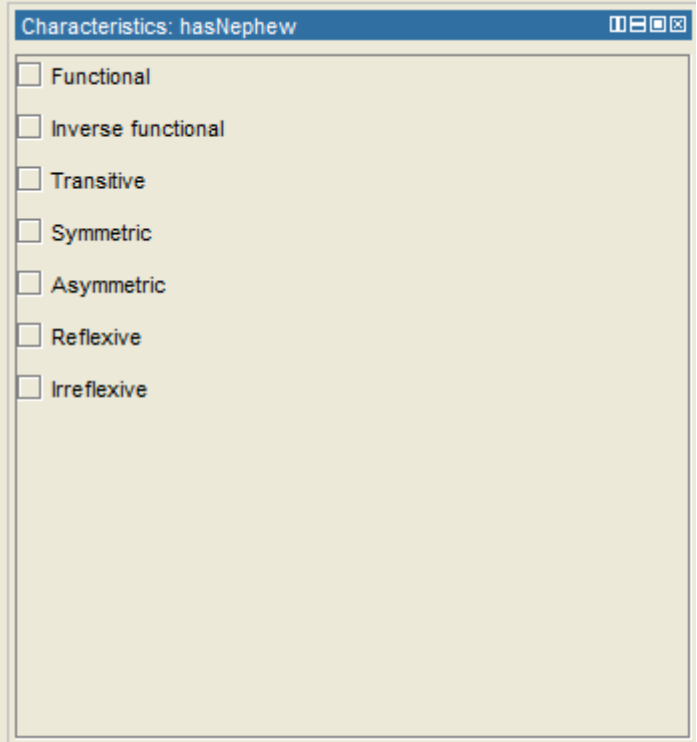


Relations in which the individuals participate, with respective concrete values.

Protégé

- Entities

Object Properties



Functional: X can only be related with one Y
(ex: R = hasMother | X hasMother Y)

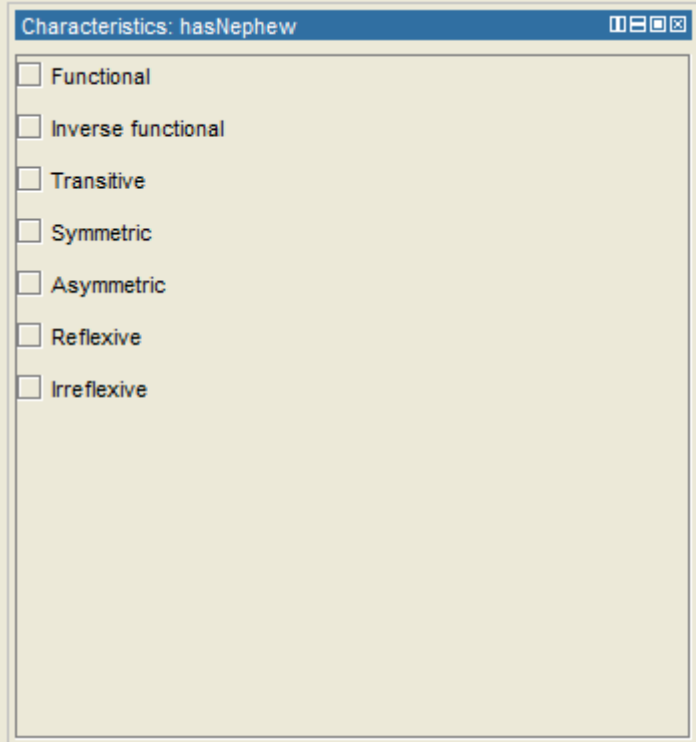
Inverse functional: the inverse property is functional
(ex: R = IsMotherOf | its inverse is hasMother)

Transitive: if a R b and b R c then a R c
(ex: R = hasAncestor | X hasAncestor Y and Y hasAncestor Z then X hasAncestor Z)

Protégé

- Entities

Object Properties



Symmetric: If a R b then b R a
(ex: R = hasSibling | X hasSibling Y and Y hasSibling X)

Asymmetric: If a R b then not b R a
(ex: R = hasFather | X hasFather Y and not Y hasFather X)

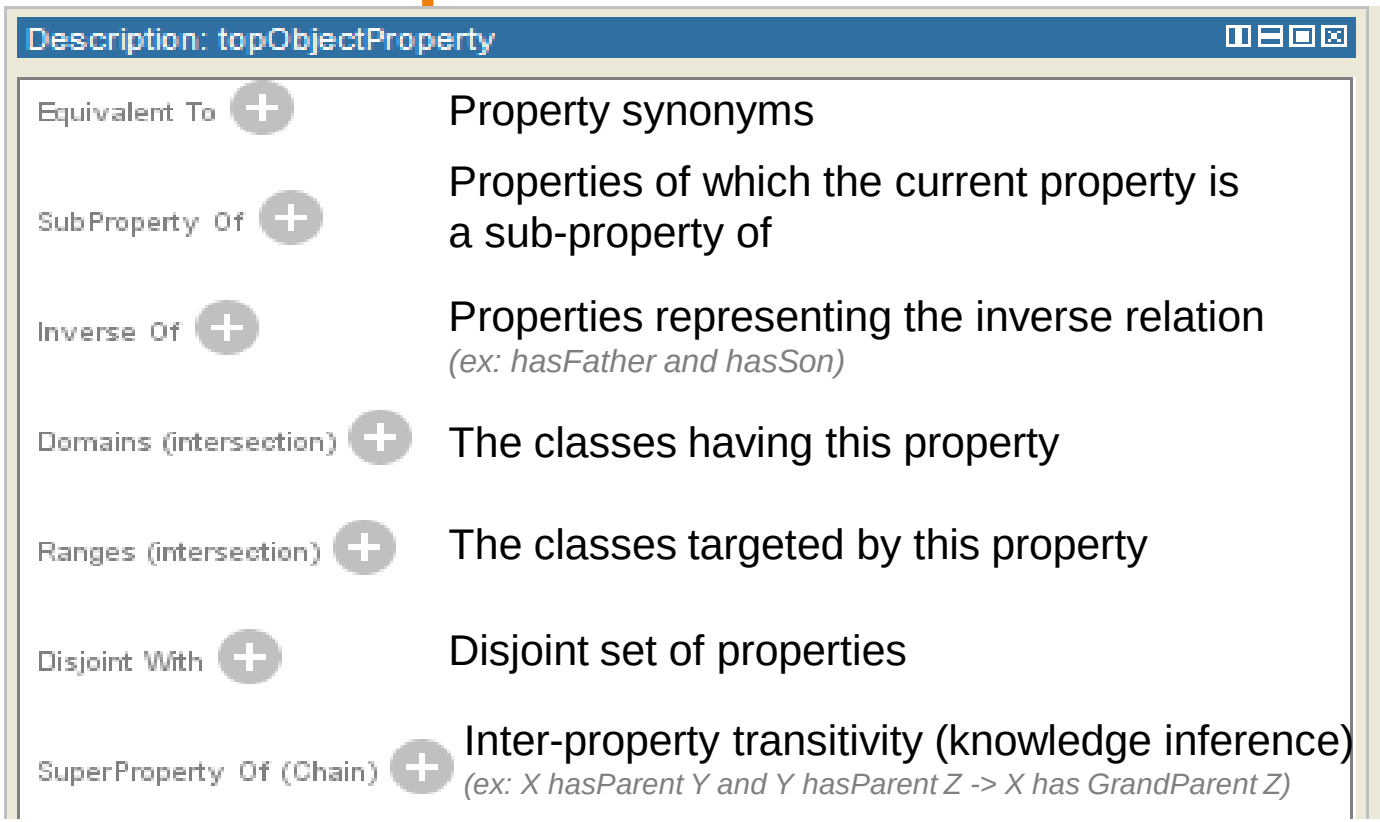
Reflexive: a R a
(ex: R = knows | X knows X)

Irreflexive: not a R a
(ex: R = hasBrother | ~~X hasBrother X~~)

Protégé

- Entities

Object Properties



Protégé

- Entities

Data Properties

- Mainly applied to **Individuals**
  - Normally it is **too general** to define a concrete value for a class.
    - *(ex: Woman hasAge 25::Integer)*
- Can only be Functional
  - *(ex: Mary hasAge 25::Integer)*

Protégé

- Entities

Partitions (Advanced Classes)

- Classes based on Partition
- Used to refine class descriptions
- Restrict the Range
- Partitions cover the possible Range of Values
  - *(ex: Child is covered by Daughter and Son)*
  - *A cover axiom is: Child is Daughter or Son*
  - *Daughter and Son are Disjoint*

Protégé

- Entities

Restrictions (Advanced Classes)

- Classes based on the properties their members may have

Protégé

- Entities

Restrictions (Advanced Classes)

- Quantifier Restrictions
  - Existential (some) or Universal (only)
    - *(ex: R **some** Y – means: class of individuals with **at least one** R relationship to Y)*
    - *(ex: R **only** Y – means: class of individuals with **all** R relationships to Y)*
- Cardinality Restrictions
  - Minimum (min), Maximum (max) Exact (exactly)
    - *(ex: R **min 3** Y – means: class of individuals with **minimum of 3 R** relationship to Y)*
    - *(ex: R **max 2** Y – means: class of individuals with **at most 2** R relationships to Y)*
    - *(ex: R **exactly 1** Y – means: class of individuals with **exactly 1** R relationships to Y)*

Protégé

- Entities

Restrictions (Advanced Classes)

- hasValue
  - Specifically Valued (value)
    - *(ex: R **value** Y – means: class of individuals with **at least one** R relationship with the value Y)*

Protégé

- Entities

Reasoning

- Reasoners that check the consistency of the Ontology
- Protégé uses Fact++ as default
- It helps on inferring new knowledge
  - *(ex: An individual that belongs to two disjoint classes)*

Protégé

- Practical Example – Family Ontology

Terms

Gender
Sister
Child
Female
Mother
Daughter
Parent
Relative
Person
Father
Aunt
Brother
Son
Uncle
Nephew
Sibling
Niece
Male

Protégé

- Practical Example – Family Ontology

Hierarchies

Gender
- Male
- Female

Person
- Child
  - Daughter
  - Son

Parent
- Mother
- Father

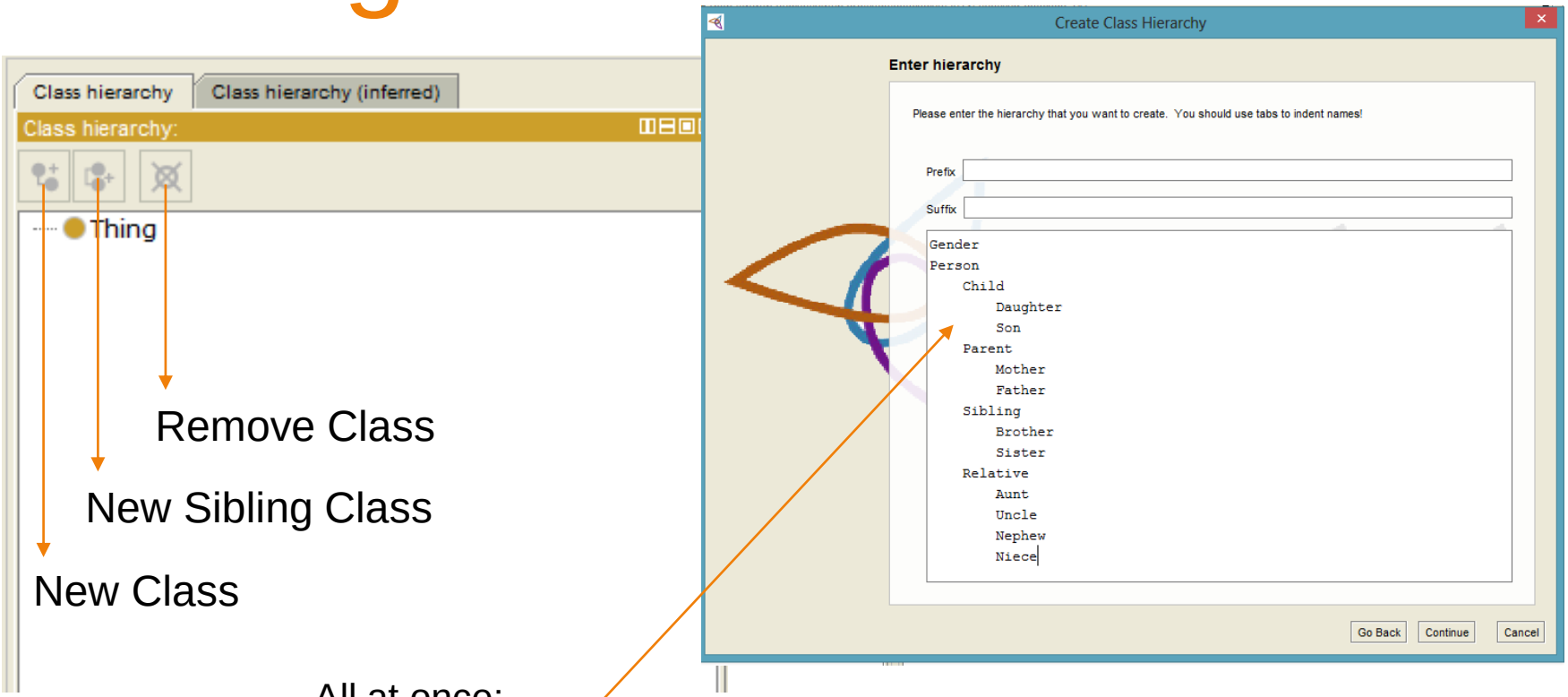
Sibling
- Brother
- Sister

Relative
- Aunt
- Uncle
- Nephew
- Niece

Protégé

- Practical Example – Family Ontology

Adding Classes



Protégé

- Practical Example – Family Ontology

Other Class Operations

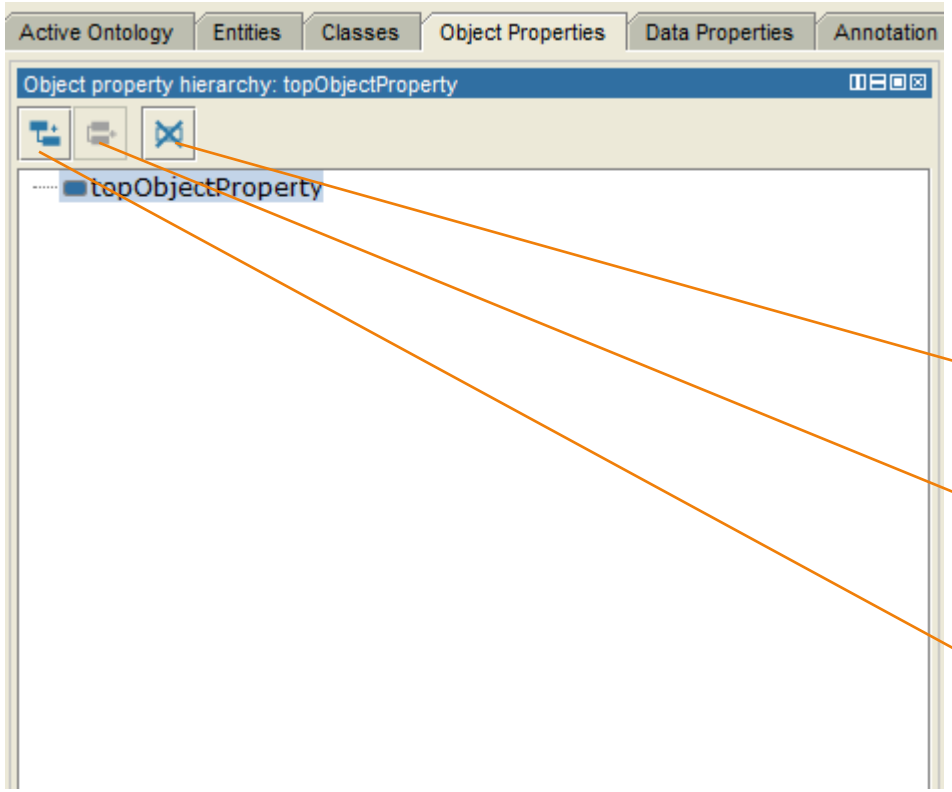
- Make Siblings Disjoint:
  - Select First Sibling and
  - Edit – Make primitive siblings disjoint

- Make Defined Class from Primitive
  - Select Primitive Class and
  - Edit – Convert to defined class

Protégé

- Practical Example – Family Ontology

Adding Properties



Remove Property

New Sibling Property

New Property

Protégé

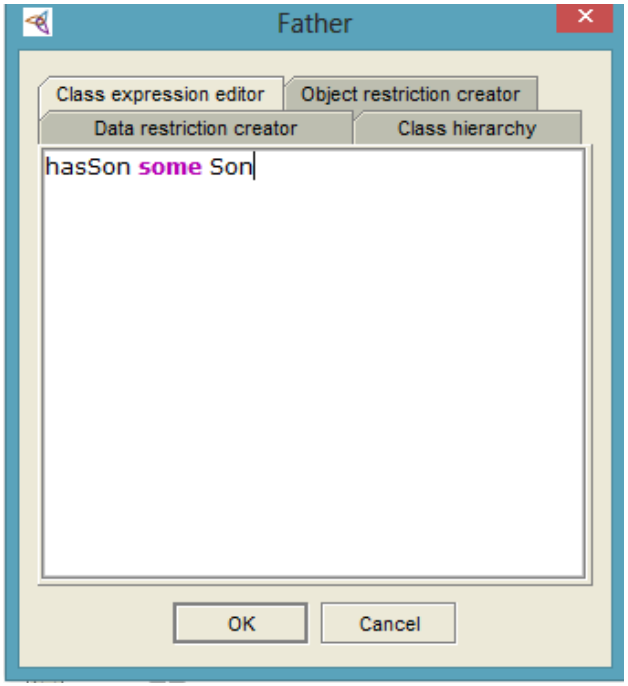
- Practical Example – Family Ontology

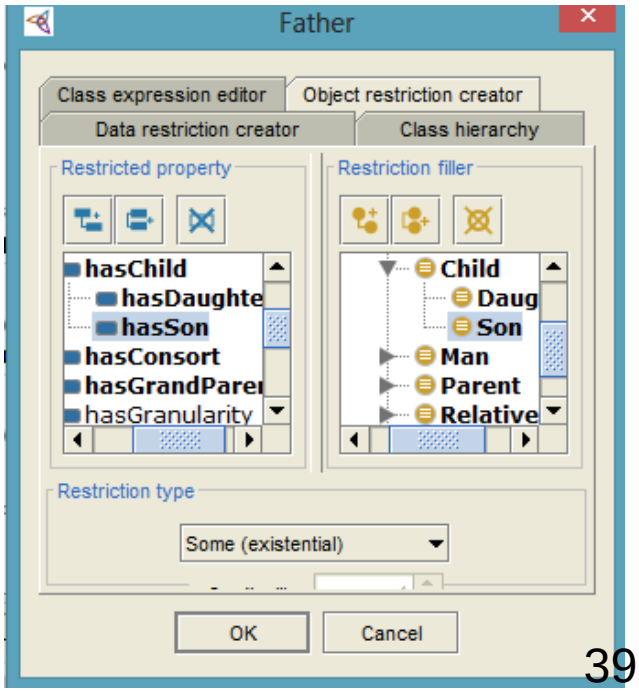
Class Restrictions

SubClass Of → Restriction Creator Trigger

Restriction Operators: some, only | min x, max x, exactly x | value

Set Operators: or, and, not



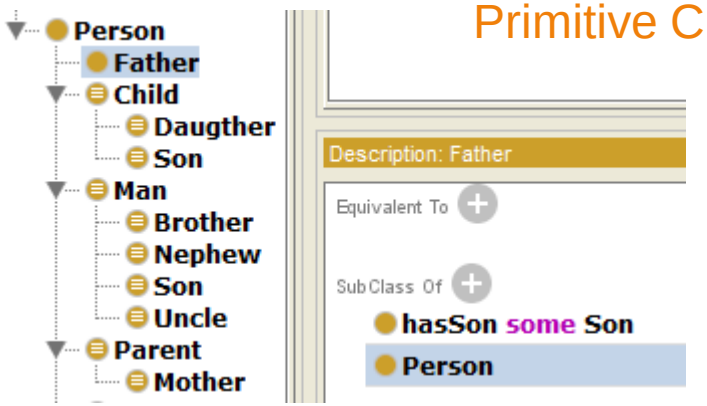


Protégé

- Practical Example – Family Ontology

Necessary and Sufficient Conditions

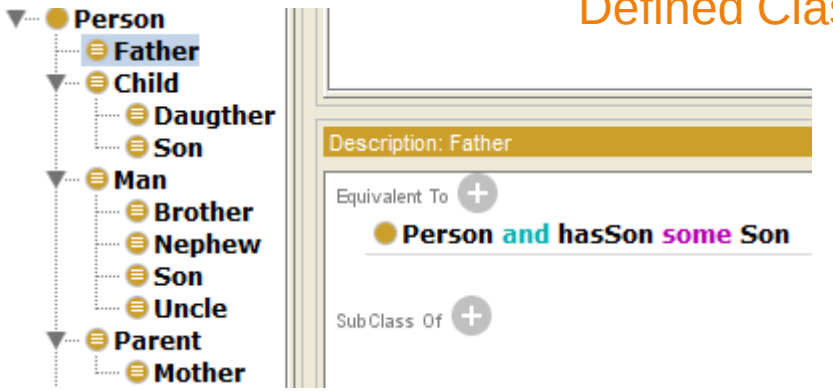
Primitive Class



Individuals of Father **are** a Person and are in 'hasSon' relationship with **at least one** individual Son.

X implies Y
or
X necessarily holds Y

Defined Class



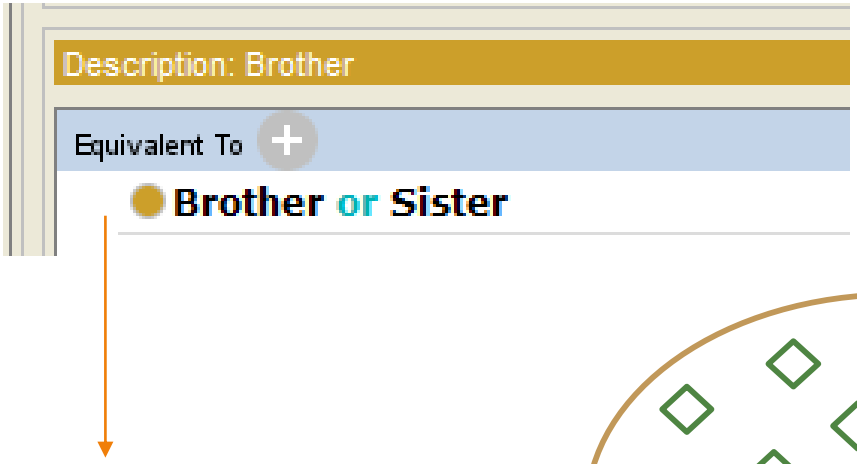
Any individual which **is** a Person and is in 'hasSon' relationship with **at least one** individual Son, **is also** a Father.

X if and only if Y
or
It suffices Y for X

Protégé

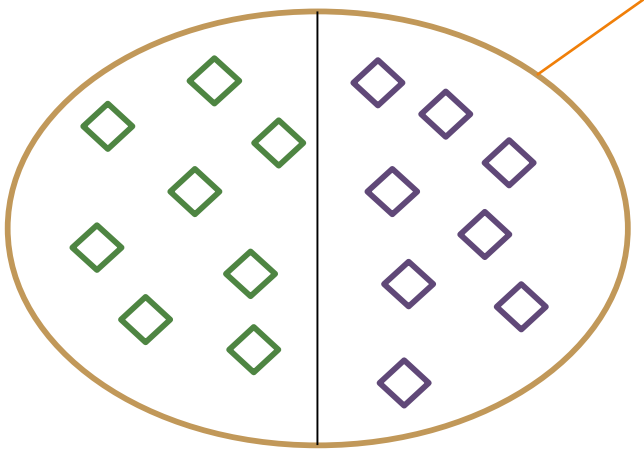
- Practical Example – Family Ontology

Partitions



Cover Axiom

Siblings



Disjoint Classes

Brother

Sister

Protégé

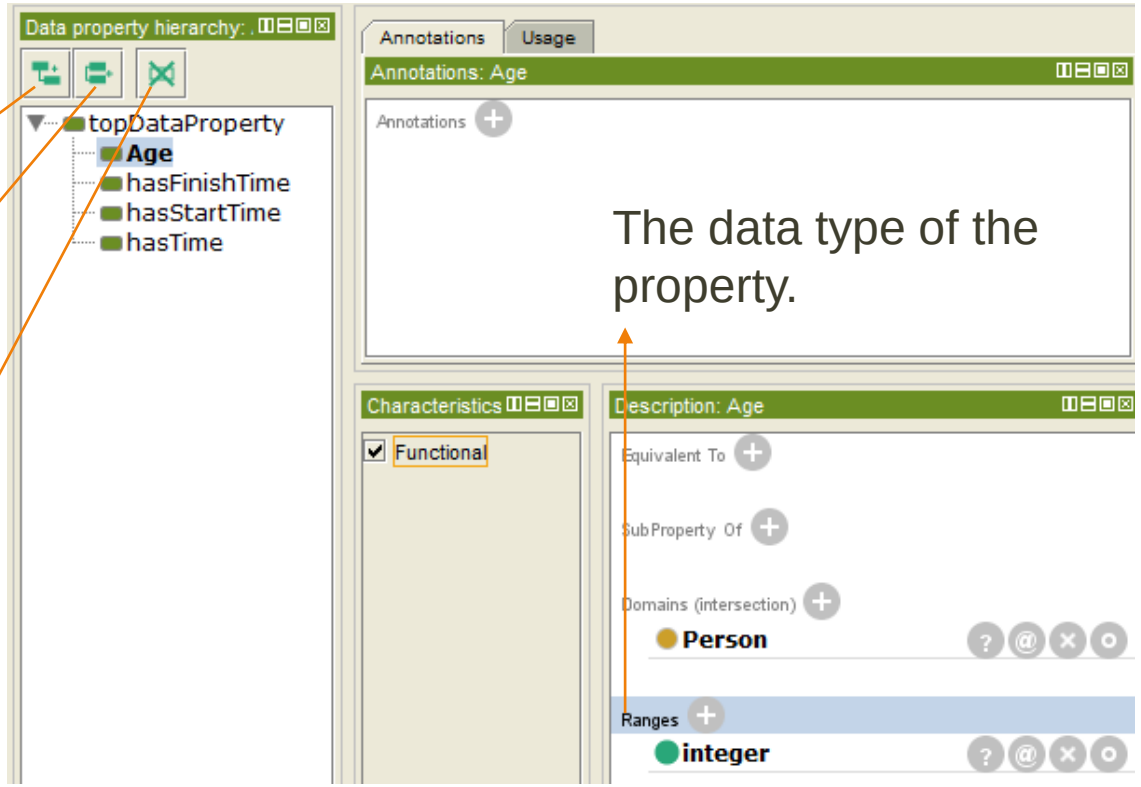
- Practical Example – Family Ontology

Data Properties

New Data Property

New Sibling Property

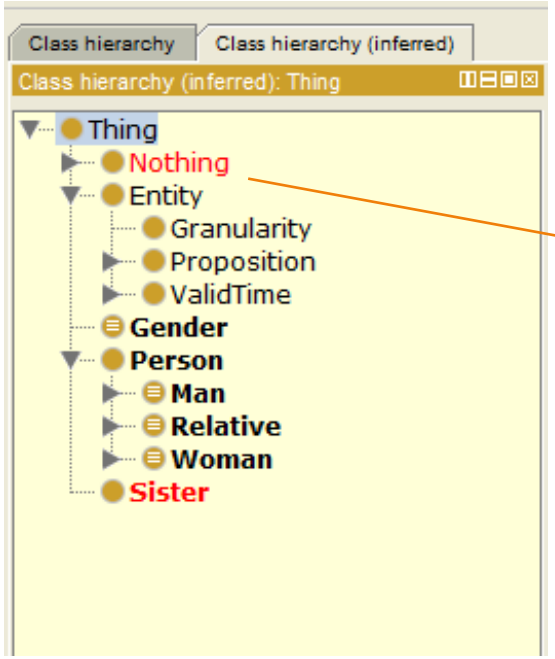
Remove Data Property



Protégé

- Practical Example – Family Ontology

Reasoner



Reasoner -> Fact++
Reasoner -> Classify

Inconsistency Problems are marked in red and are placed under the Nothing Class.

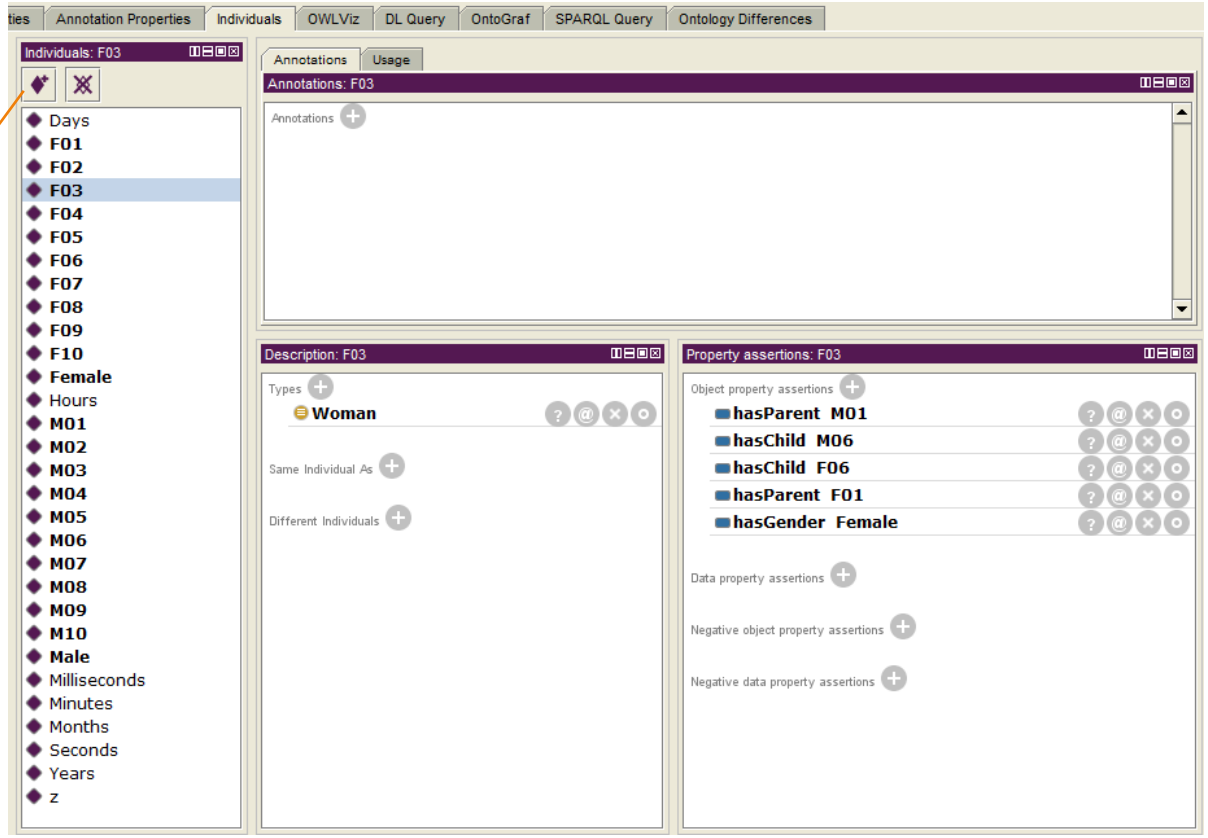
Asserted Classes: User-defined classes
Inferred Classes: Reasoner-discovered classes and hierarchy.

Protégé

- Practical Example – Family Ontology

Individuals

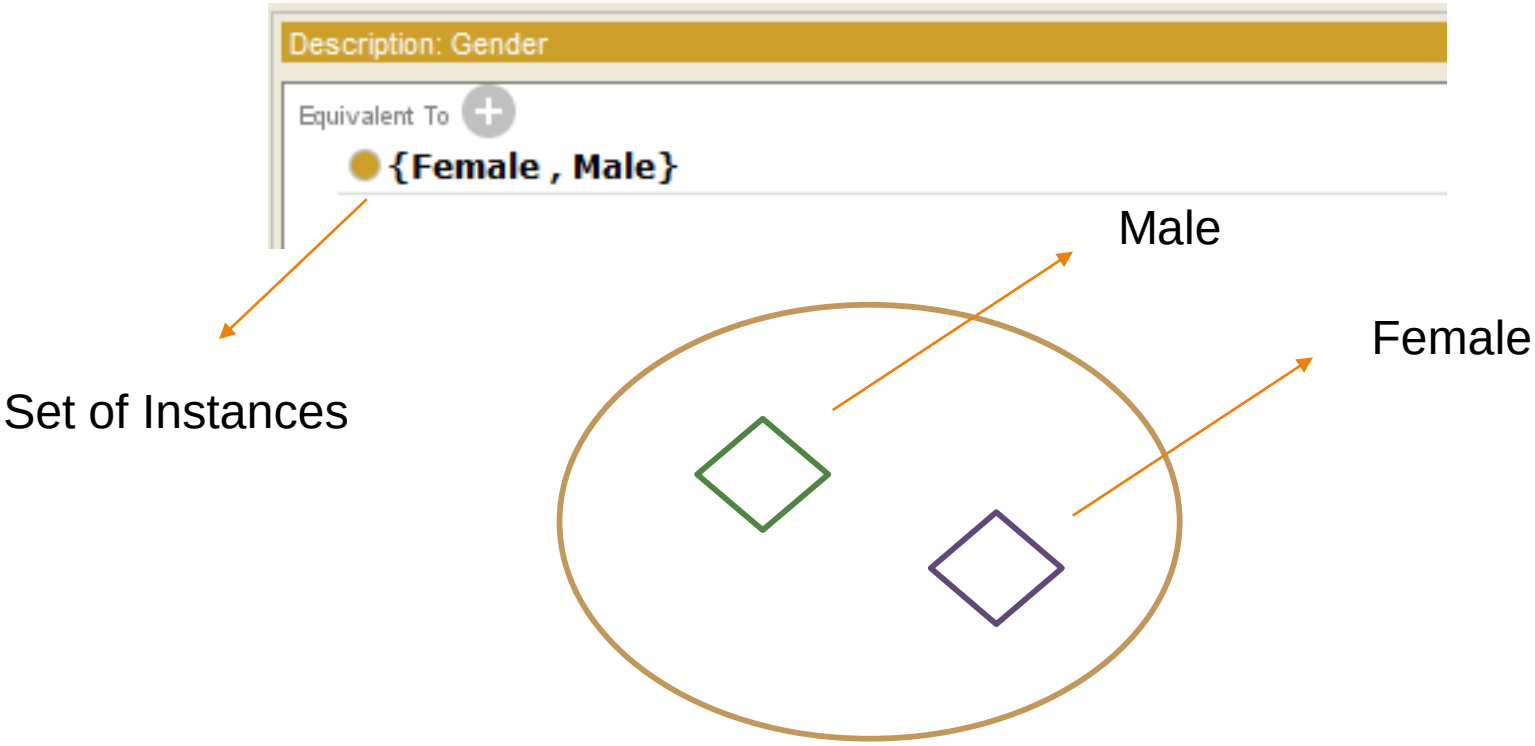
Add New Individuals



Protégé

- Practical Example – Family Ontology

Enumerations



Bibliography

1. Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. In Development, vol. 32, Nr. 1, pp. 1-25. 2001.

1. Ivo Serra and Rosario Girardi. A Process for Extracting Non-Taxonomic Relations of Ontologies from Text. In Intelligent Information Management, vol. 3, Nr. 4, pp. 119-124. July, 2009.

2. M. Horridge and H. Knublauch and A. Rector and R. Stevens and C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools. The University Of Manchester, vol 27, pp. 0-117. 2004.

1. OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features/ . 2004.

Acknowledgements

- Nuno Oliveira, Doctoral Student at the High-Assurance Software Laboratory (HASLab), University of Minho.

In the next session…

- CompGuide – ontology for Computer-Interpretable Guidelines

Intelligent Systems

ONTOLOGY WEB LANGUAGE AND PROTÉGÉ

Intelligent Systems Lab

Universidade do Minho

Tiago Oliveira, Paulo Novais and José Neves