Protégé

Ontology Building Environment

Prof Christophe Roche

ERA Chair Holder - University of Crete

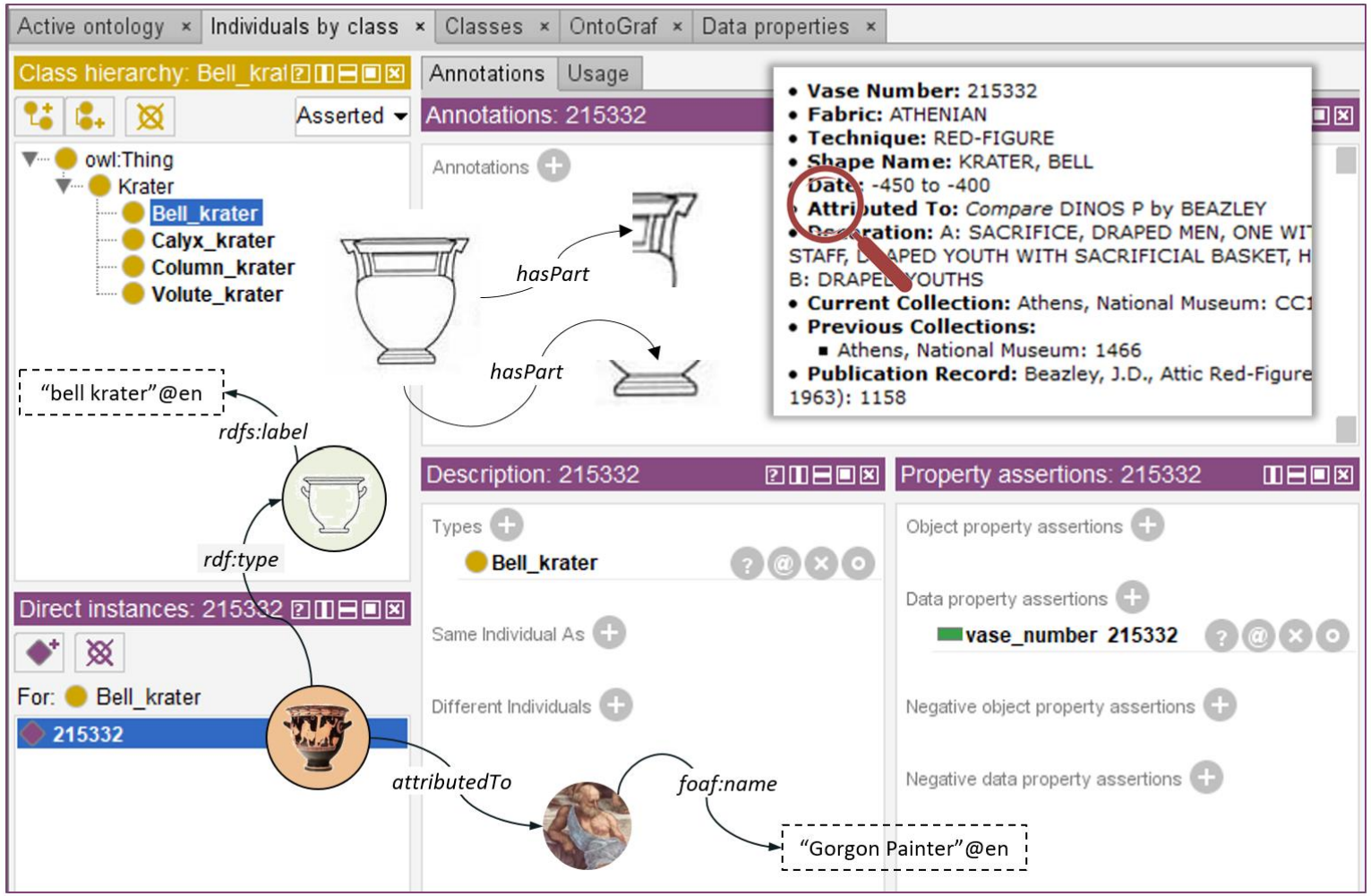
https://christophe-roche.fr/

May 2025

Horizon ERA Chair TALOS AI4SSH Project funded by the European Commission
Grant Agreement n° 101087269, https://cordis.europa.eu/project/id/101087269

Funded by the European Union

Contents



0. Introduction

https://protege.stanford.edu/

✓ Protégé is a free, open-source ontology editor written in Java developed at Stanford University

✓ More than 300,000 users are registered.

protégé

- ✓ *W3C standards compliant*
- ✓ *Customizable user interface*
- ✓ *Visualization support*
- ✓ *Ontology refactoring support*
- ✓ *Direct interface to reasoners*
- ✓ *Highly pluggable architecture*
- ✓ *Cross compatible with WebProtégé*

WHY PROTÉGÉ

Protégé's plug-in architecture can be adapted to build both simple and complex ontology-based applications. Developers can integrate the output of Protégé with rule systems or other problem solvers to construct a wide range of intelligent systems. Most important, the Stanford team and the vast Protégé community are here to help.

ACTIVE COMMUNITY

Protégé is actively supported by a strong community of users and developers that field questions, write documentation, and contribute plug-ins.

W3C STANDARDS SUPPORT

Protégé fully supports the latest OWL 2 Web Ontology Language and RDF specifications from the World Wide Web Consortium.

EXTENSIBLE OPEN SOURCE ENVIRONMENT

Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development.

1. Definitions

Ontology

An ontology is a formally-defined vocabulary for a particular domain of interest. Ontologies are typically based on a class hierarchy (asserted and/or inferred), supplemented by assorted properties.

https://protegewiki.stanford.edu/wiki/Pr4_UG_mi_Glossary#Ontology

☞ **Ontologies are used to capture knowledge about some domain of interest. An ontology describes the concepts in the domain and also the relationships that hold between those concepts.**

A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools - Edition 1.3

Language

OWL provides the theoretical basis for Protégé ontologies.

https://protegewiki.stanford.edu/wiki/Pr4_UG_mi_Glossary#Ontology

Different ontology languages provide different facilities. The most recent development in standard ontology languages is OWL from the World Wide Web Consortium (W3C).

A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools - Edition 1.3

Reasoner

The logical model allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognise which concepts fit under which definitions. The reasoner can therefore help to maintain the hierarchy correctly.

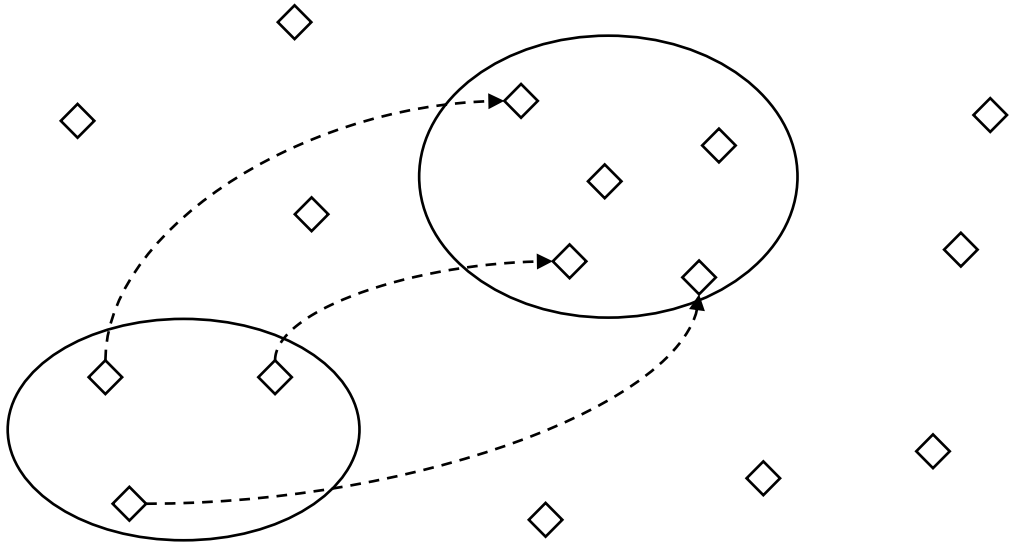
A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools - Edition 1.3

☞ **an inconsistent class is a class which cannot contain any individual because of its definition**

2. Principles

Extensional Logic

Goal: **Organising** the objects which populate the world into **classes** according to the **relationships** that linked objects together



⇨ An object is not defined by its “nature“, but by its relations with other objects

2. Principles

Components of OWL Ontologies

1) Individuals

Individuals, represent objects in the domain in which we are interested

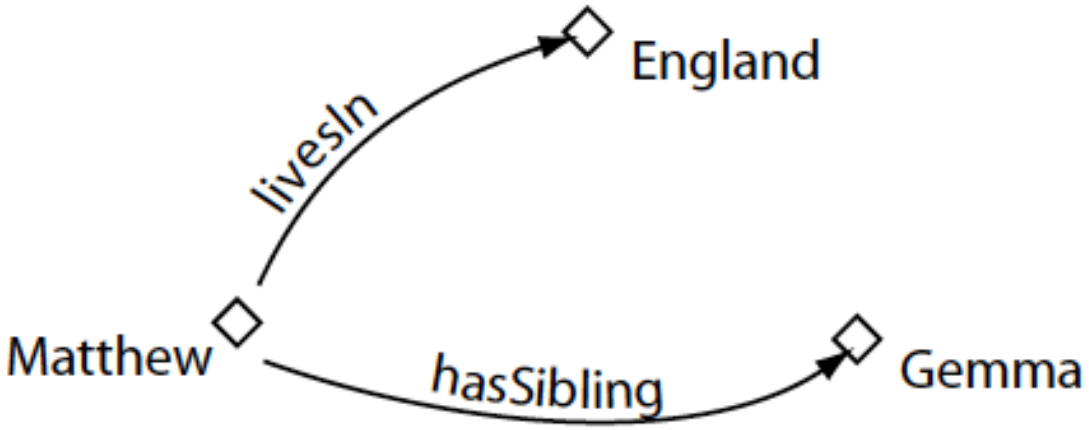
<u>Terminology</u>: « individual », « instance », « object »

2. Principles

Components of OWL Ontologies

2) Object Properties

Properties are binary relations on individuals, i.e. properties link two individuals together.



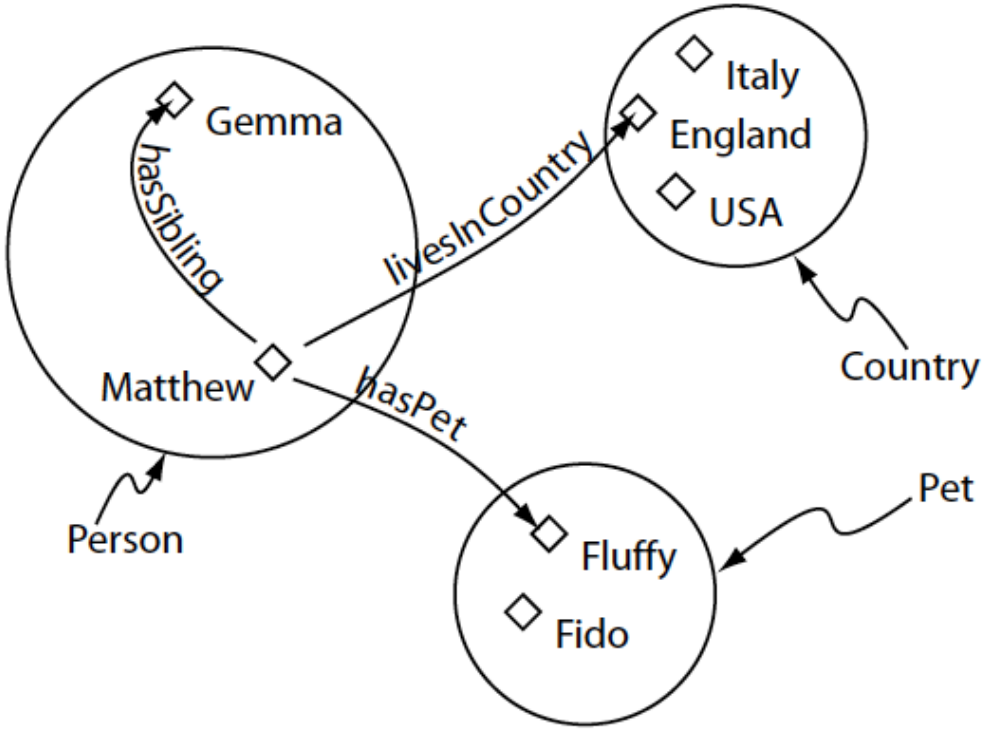
<u>Terminology</u>: « properties », « slots » (Protégé), « roles » (DL), « relations », « attributes »

2. Principles

Components of OWL Ontologies

3) Classes

OWL classes are interpreted as sets that contain individuals.



Classes are a concrete representation of concepts.

Classes are <u>defined</u> using <u>formal descriptions</u> that state precisely the requirements for membership of the class.

Person = ?

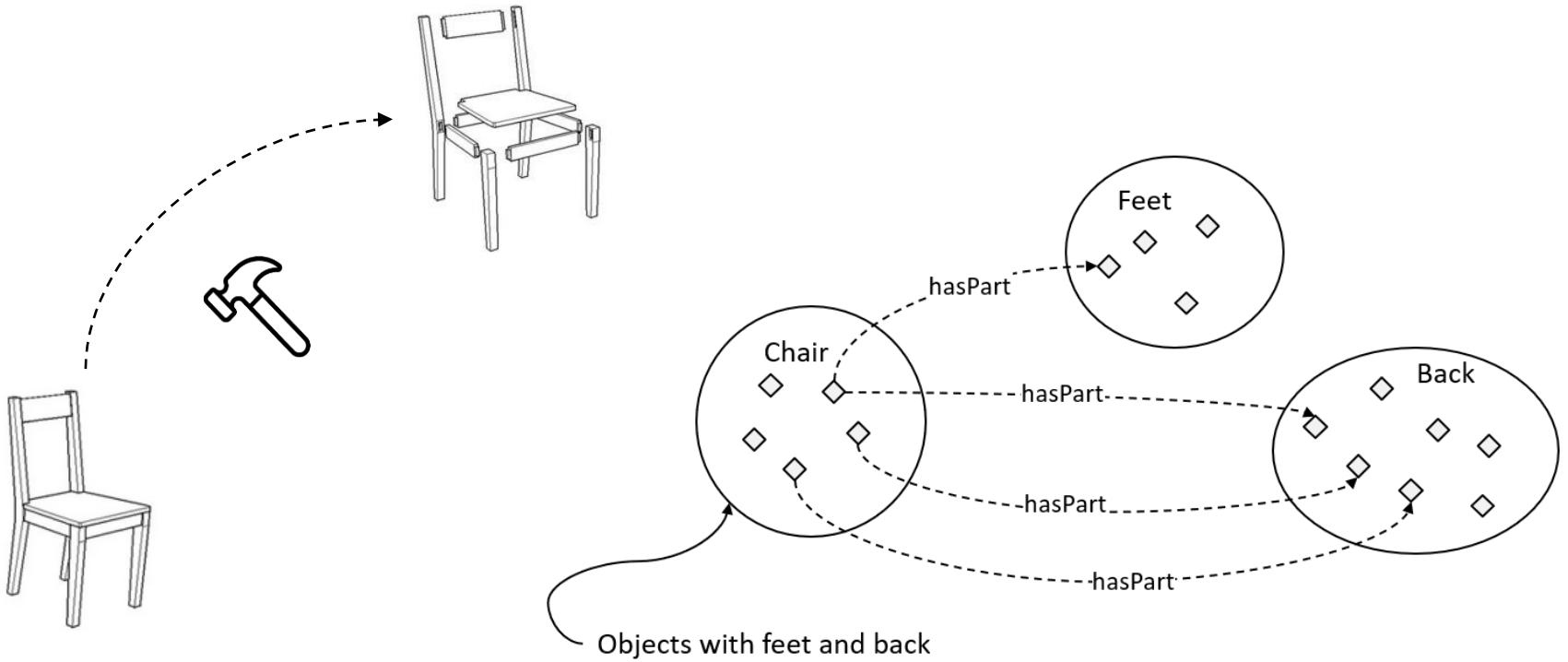
Person = { x / ∃ y Country(y) ∧ livesInCountry (x,y) }

Person = { x / ∃ y Pet(y) ∧ hasPet (x,y) }

Person = { x / ∃ y hasSibling(x,y) ∨ hasSibling (y,x) }

2. Principles

Change your way of thinking

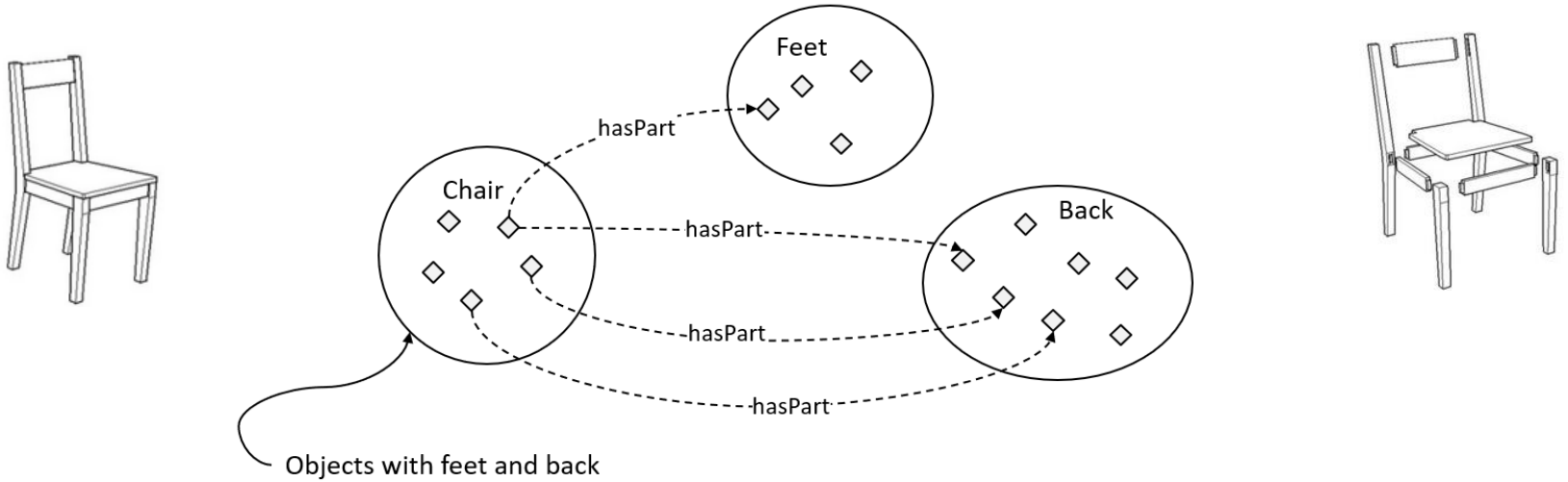


2. Principles

4) Property restriction

A means to define classes of individuals

a) Existential Restrictions: describes (anonymous) classes of individuals that participate in *at least one* (some) relationship along a specified property to individuals that are members of a specified class.



b) Universal Restrictions: describes (anonymous) classes of individuals that for a given property *only* (only) have relationships along this property to individuals that are members of a specified class (all values of the property must be of a certain type)

c) Has value: at least one of the values of the property is a certain value

3. Example

CLASSICAL ART RESEARCH CENTRE

UNIVERSITY OF OXFORD

Beazley Archive: https://www.carc.ox.ac.uk/carc/Home

The Krater Ontology

"The term **'krater**' suggests a mixing-vessel (compare Greek kerannumi - to mix), and we know that the wine served at the symposium was mixed with water.

Column-krater: Named for its column-like handles, the column-krater is first known from Corinthian examples dated to the late seventh century. It is regularly produced by Athenian potters from the first half of the sixth-century until the third quarter of the fifth. It seems from graffiti on Athenian red-figure examples that the vessel was referred to as *Korinthios* or *Korinthiourges*.

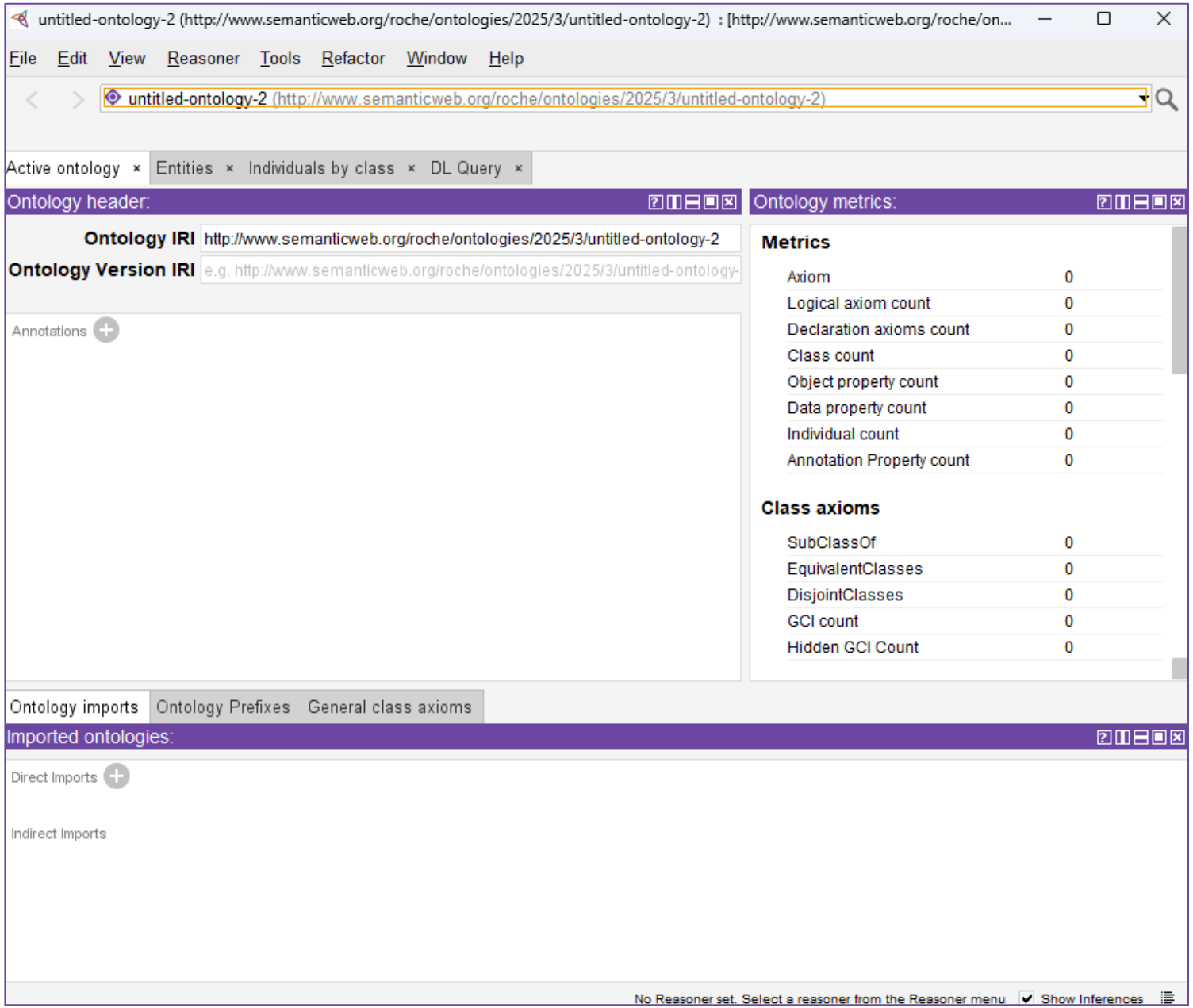
Volute-krater: The volute-krater is named after its handles. The François Vase is a famous and early example, but the typical Athenian form occurs only later in the sixth century, with the handles tightly curled so that they look like the volutes on Ionic columns. The shape is also found in metal. Over the course of the fifth and fourth centuries, examples become slimmer, and Apulian volute-kraters from South Italy are particularly elaborate.

Calyx-krater: The handles of the calyx-krater are placed low down on the body, at what is termed the cul. Their upward curling form lends the shape an appearance reminiscent of the calyx of a flower, hence the name. The earliest known example was possibly made by Exekias in the third quarter of the sixth century. It continues to be produced, mainly in red-figure, becoming more elongated over the course of the fifth and fourth centuries.

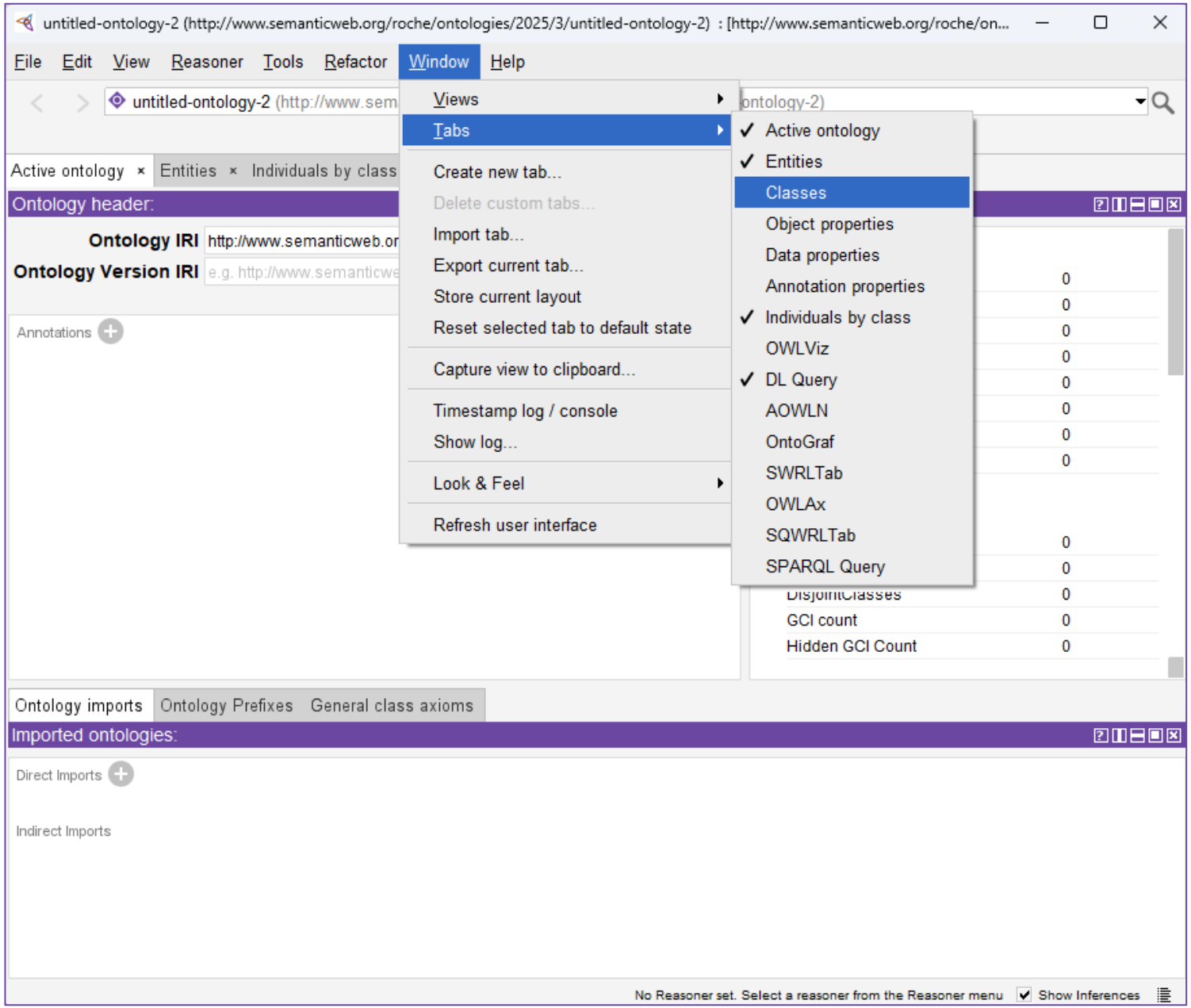
Bell-krater: The latest of the four krater-types, it first occurs in the early fifth century, and is not found decorated in black-figure. It is named for its bell-like shape, perhaps originating in wood. It has small horizontal upturned handles just over halfway up the body. Some do not have a foot, and earlier examples may have lugs for handles.

4. Ontology Building

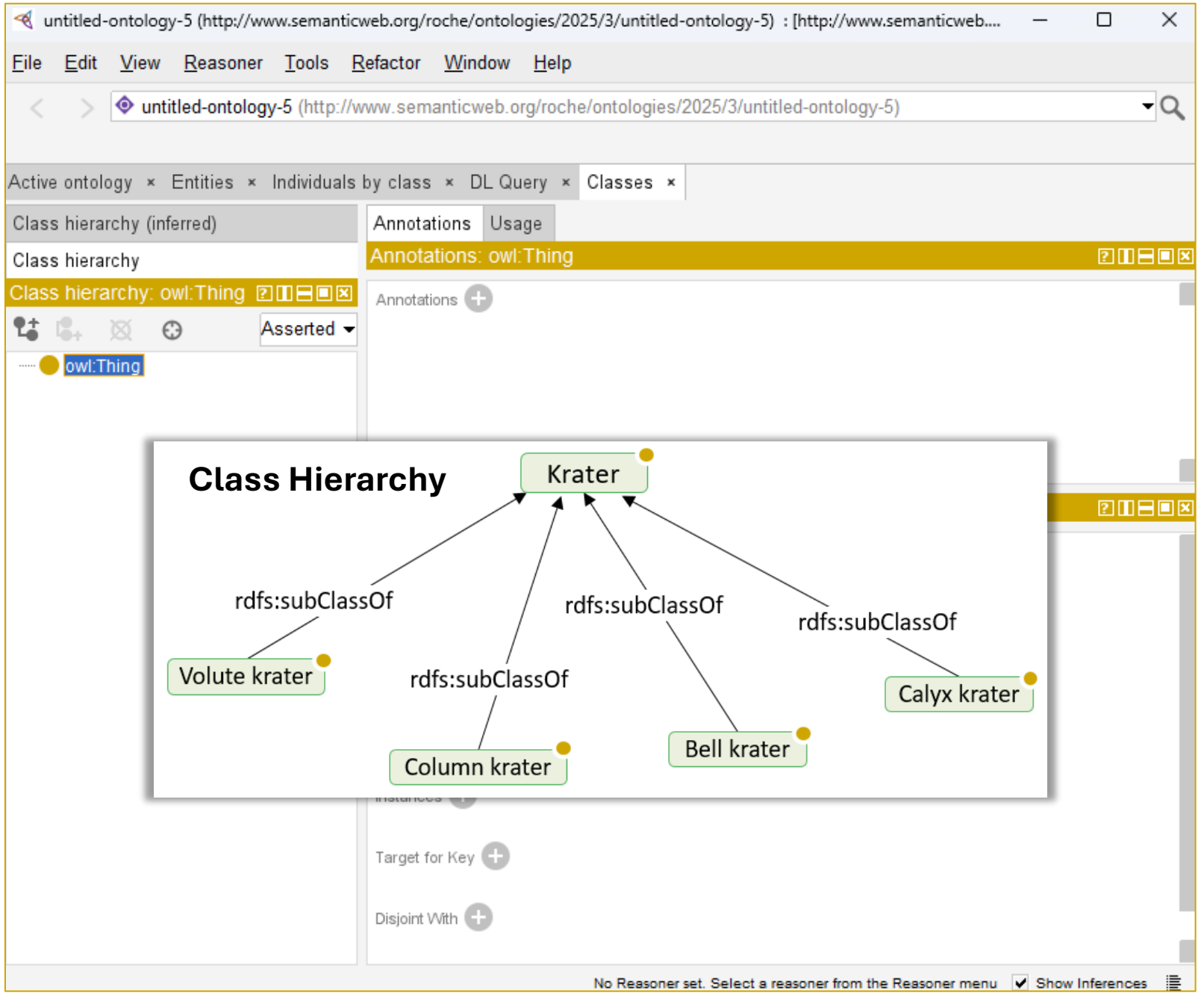
protégé



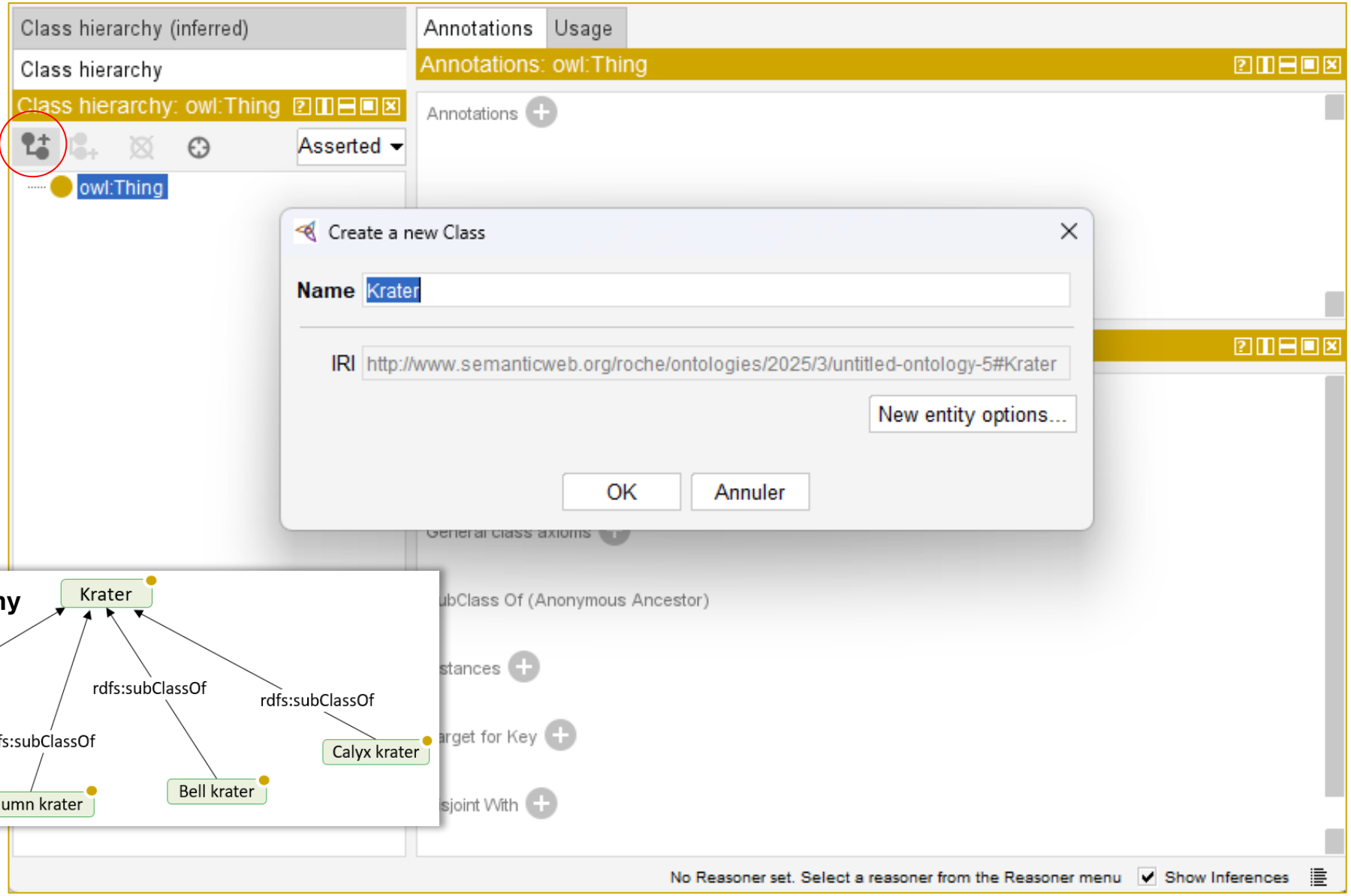
4. Ontology Building

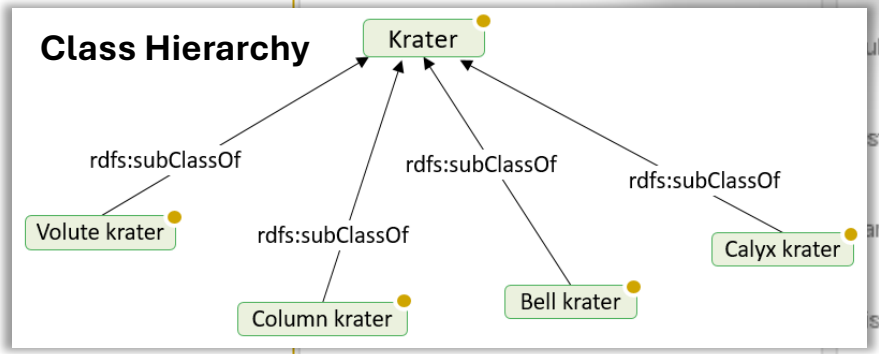


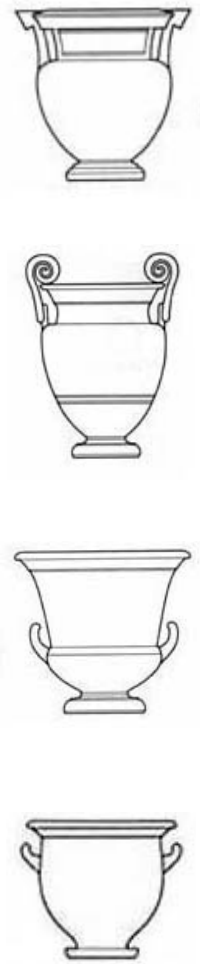
4. Ontology Building



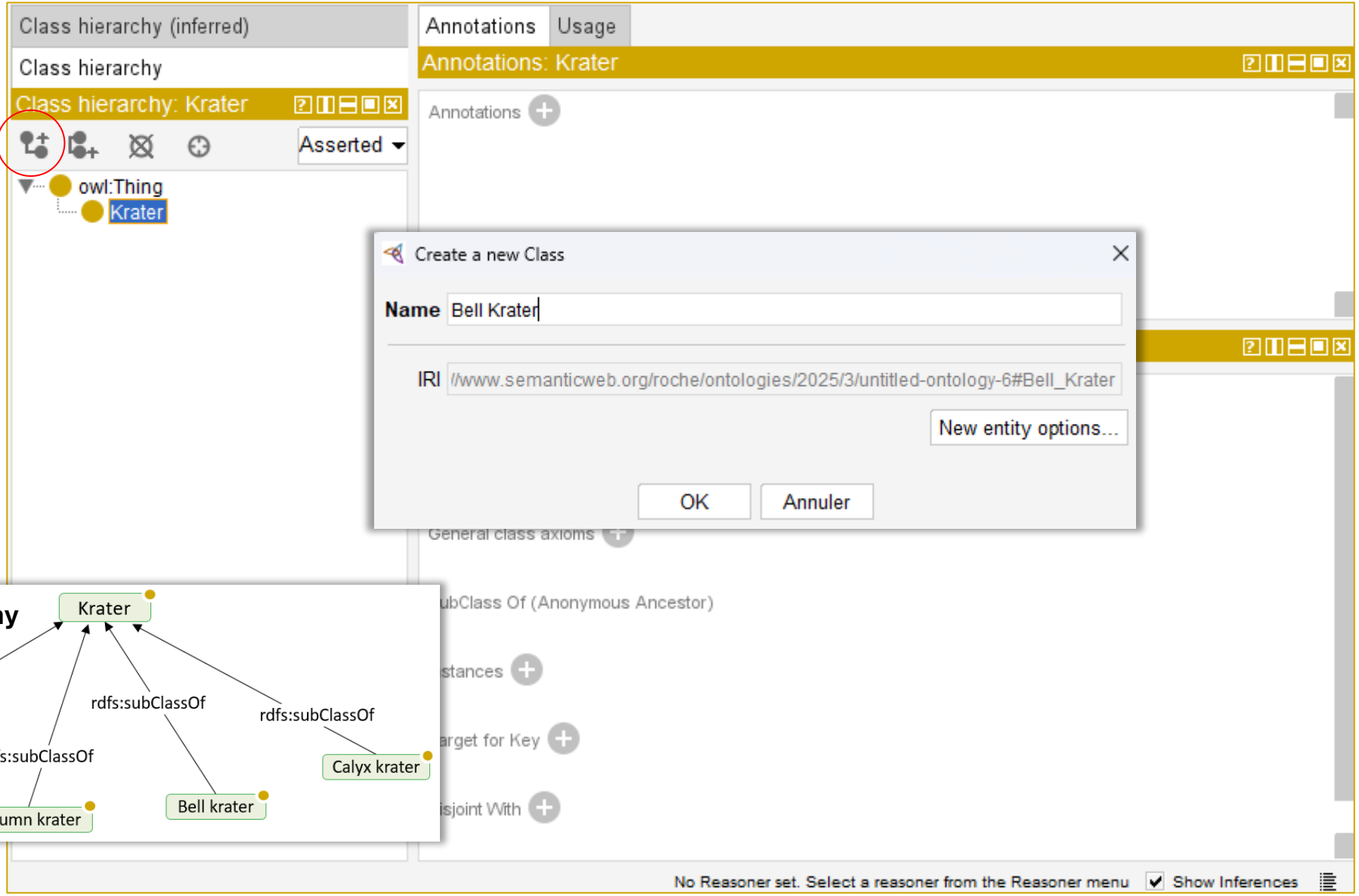
4. Ontology Building

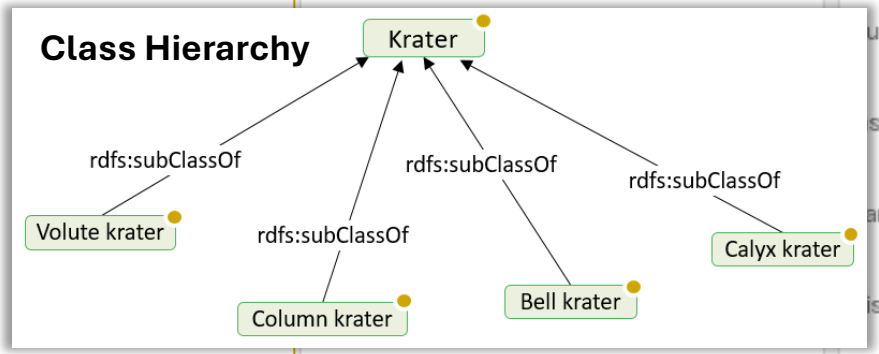


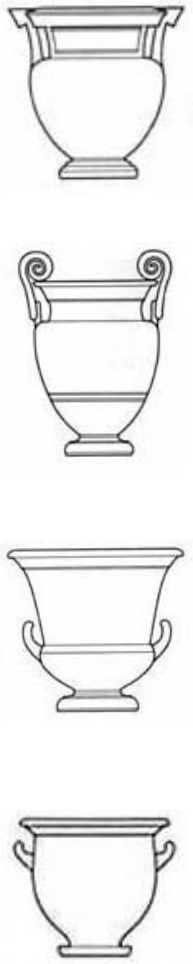




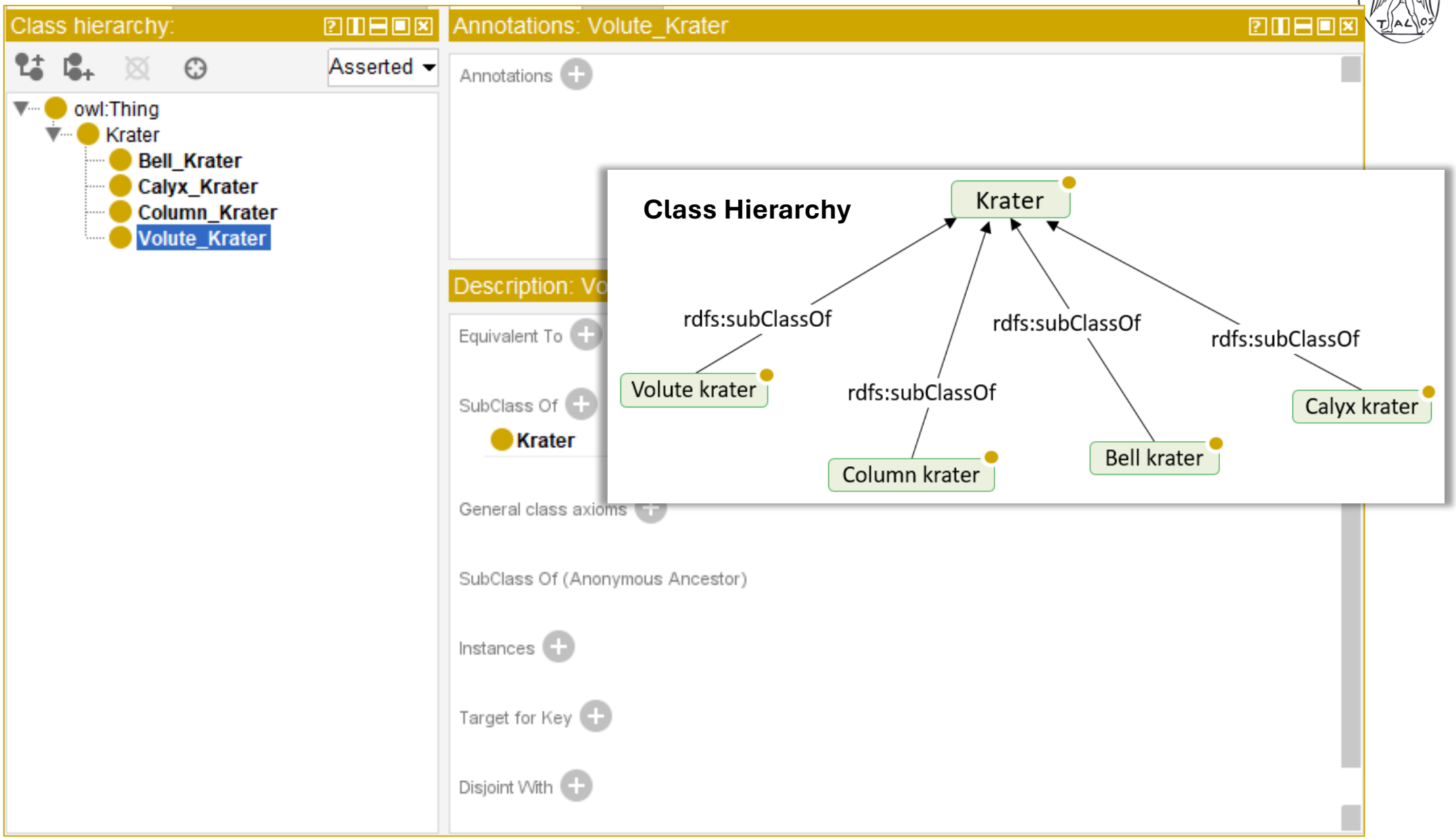
4. Ontology Building: Class Hierarchy



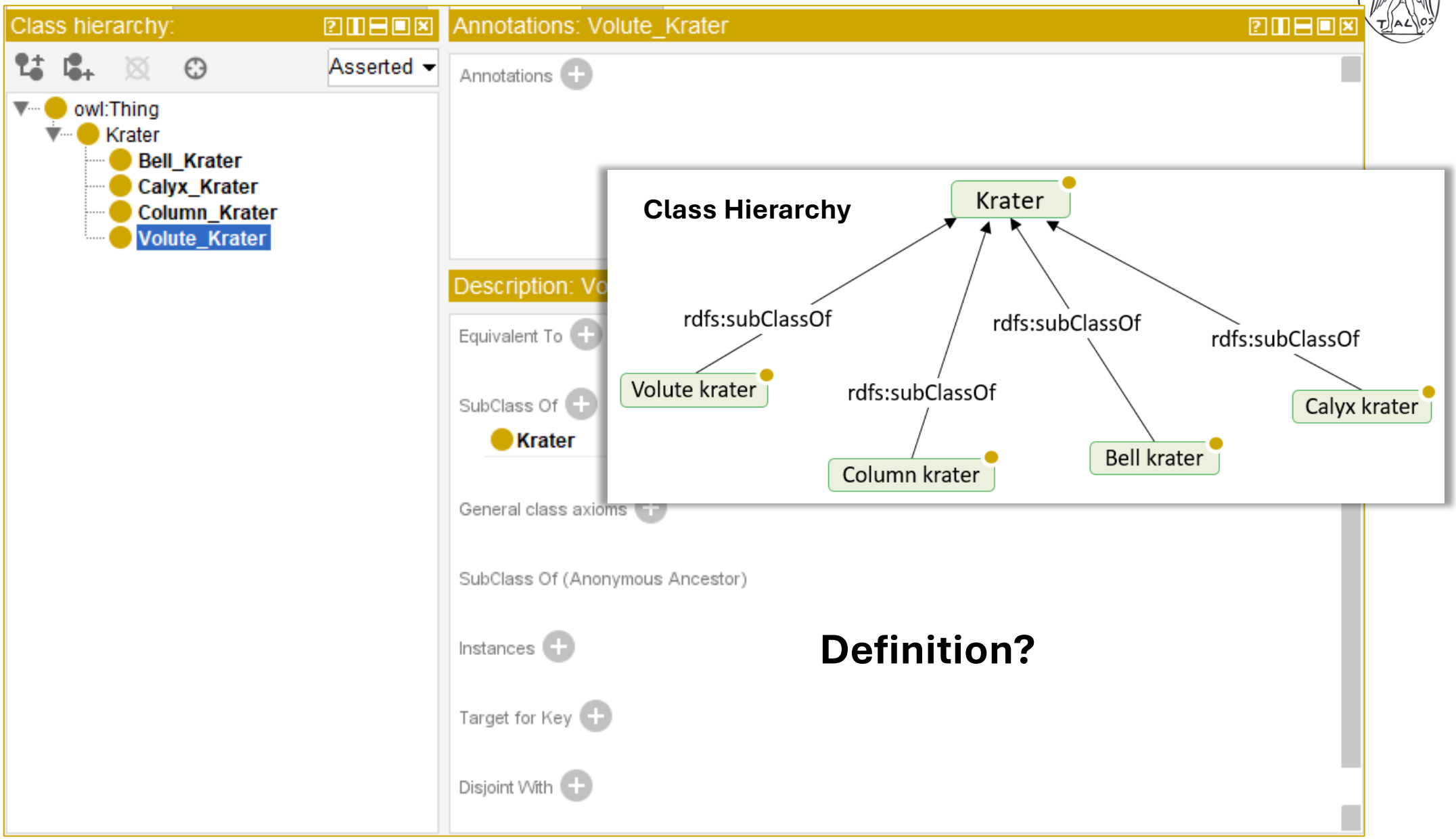




4. Ontology Building



4. Ontology Building



4. Ontology Building

Class hierarchy: Volute_Krater

Asserted

- owl:Thing
  - Krater
    - Bell_Krater
    - Calyx_Krater
    - Column_Krater
    - Volute_Krater

Annotations: Volute_Krater

Annotations

Description: Vo

SubClass Of

Krater

General class axio

SubClass Of (Anonymous Ancestor)

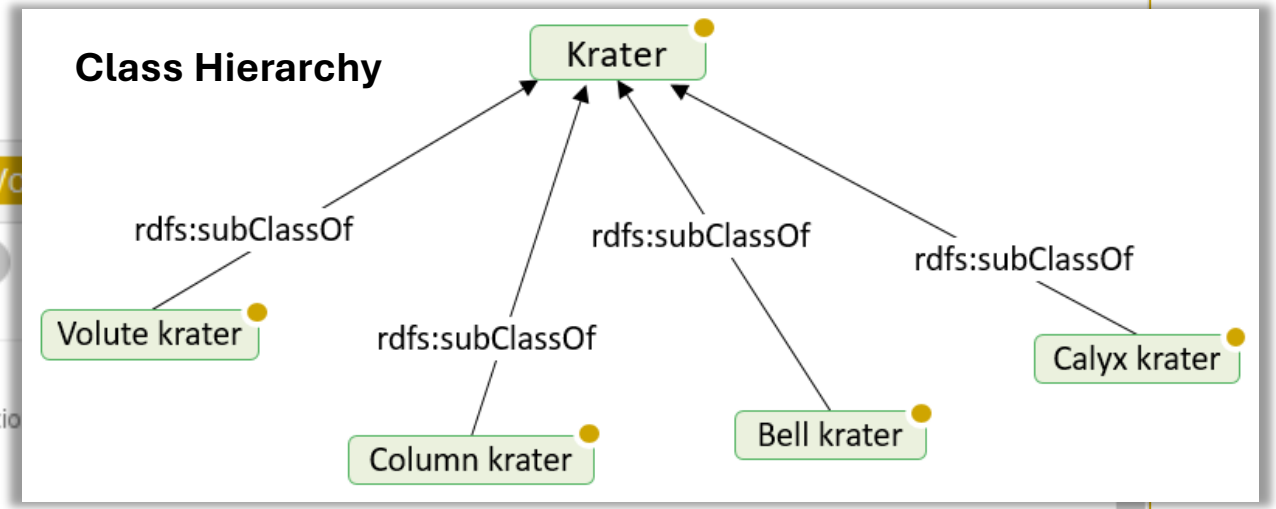
Instances

Target for Key

Disjoint With

Calyx_Krater, Column_Krater, Bell_Krater

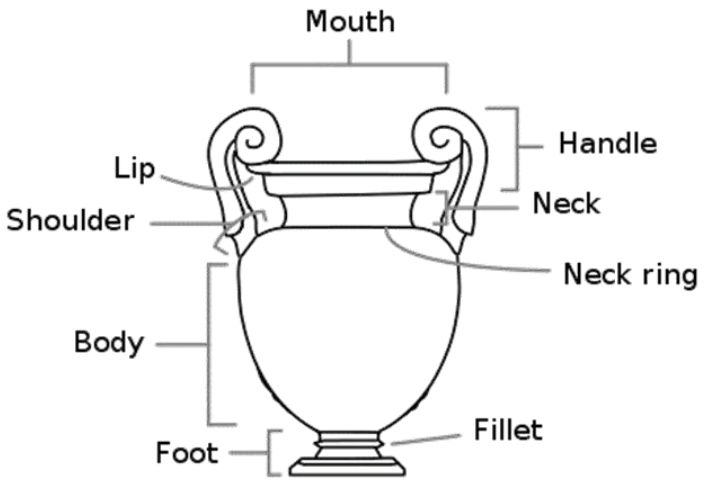
Disjoint Union Of

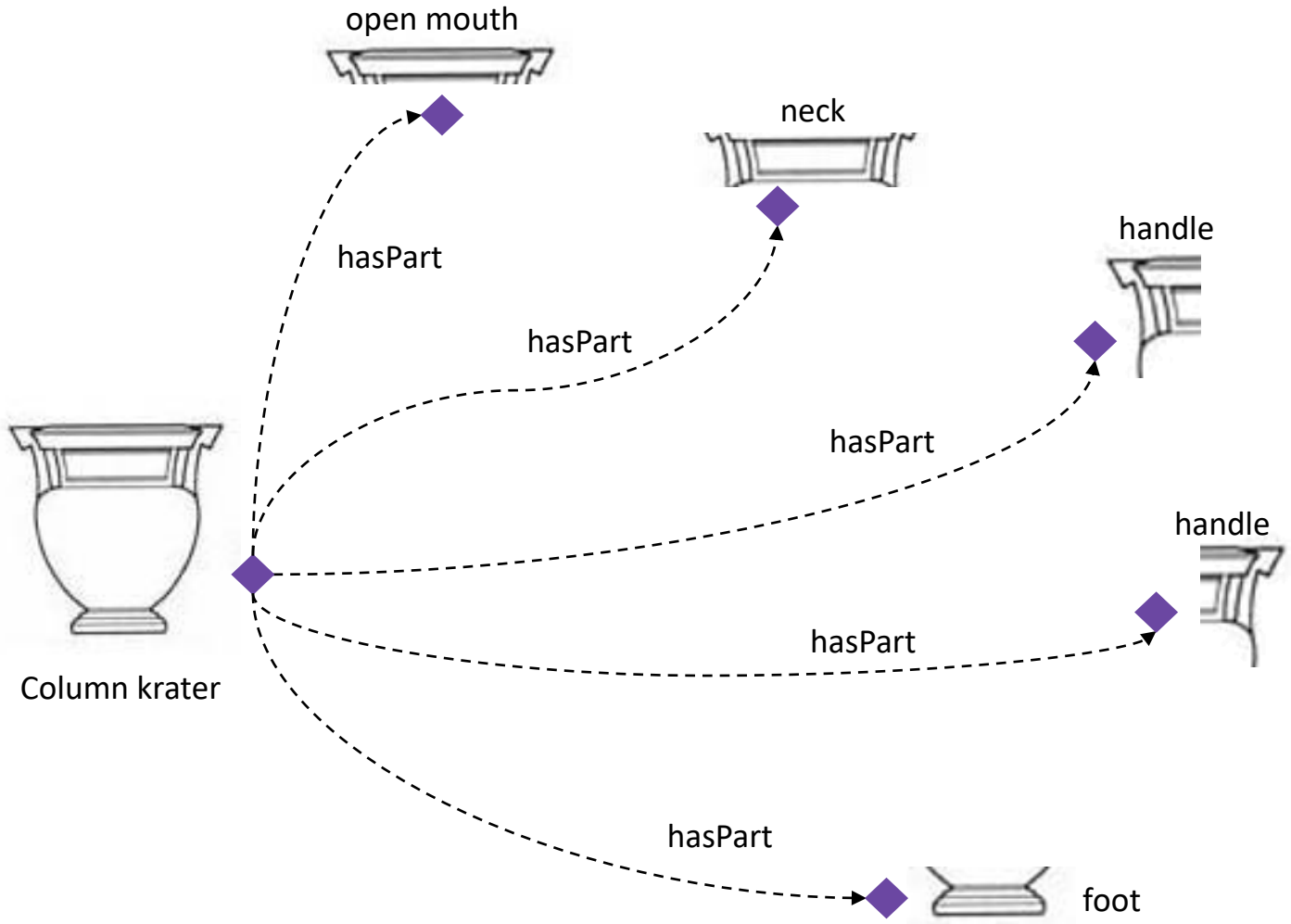


Definition?

Object Properties

Relationships between individuals





4. Ontology Building

Class hierarchy: Part

Asserted

- owl:Thing
  - Krater
    - Bell_Krater
    - Calyx_Krater
    - Column_Krater
    - Volute_Krater
  - Part
    - Foot
    - Handle
      - Column_like_Handle
      - Upward_Curling_Handle
      - Volute_like_Handle
    - Mouth
      - Close_Mouth
      - Open_Mouth
    - Neck

Annotations: Part

Annotations

Description: Part

Equivalent To

SubClass Of

General class axioms

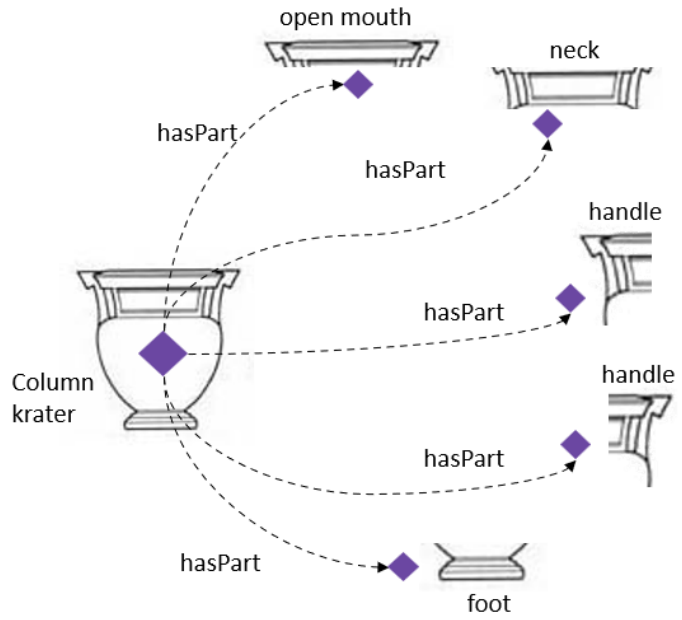
SubClass Of (Anonymous Ancestor)

Instances

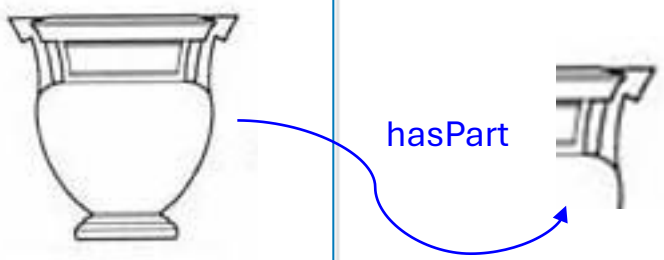
Target for Key

Disjoint With

Krater



4. Ontology Building



Object property hierarchy: hasPart

Asserted

owl:topObjectProperty

hasPart

Annotations | Usage

Annotations: hasPart

Annotations

Cha

Description: hasPart

- [ ] Functional
- [ ] Inverse functio
- [x] Transitive
- [ ] Symmetric
- [x] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

Equivalent To

SubProperty Of

Inverse Of

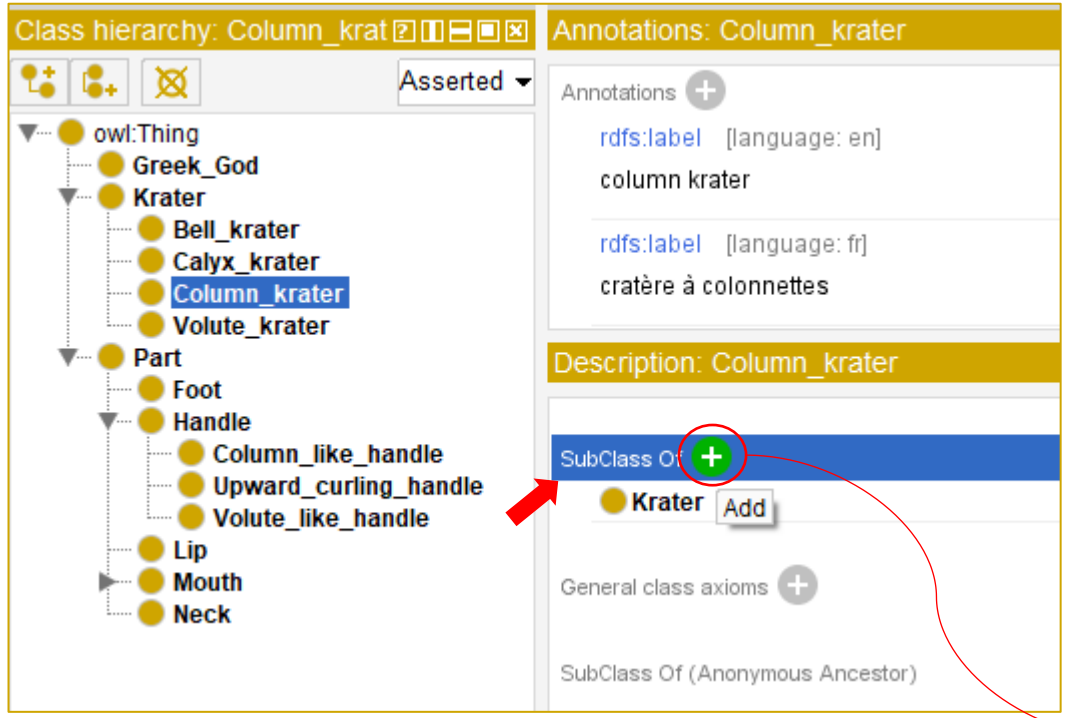
Domains (intersection)

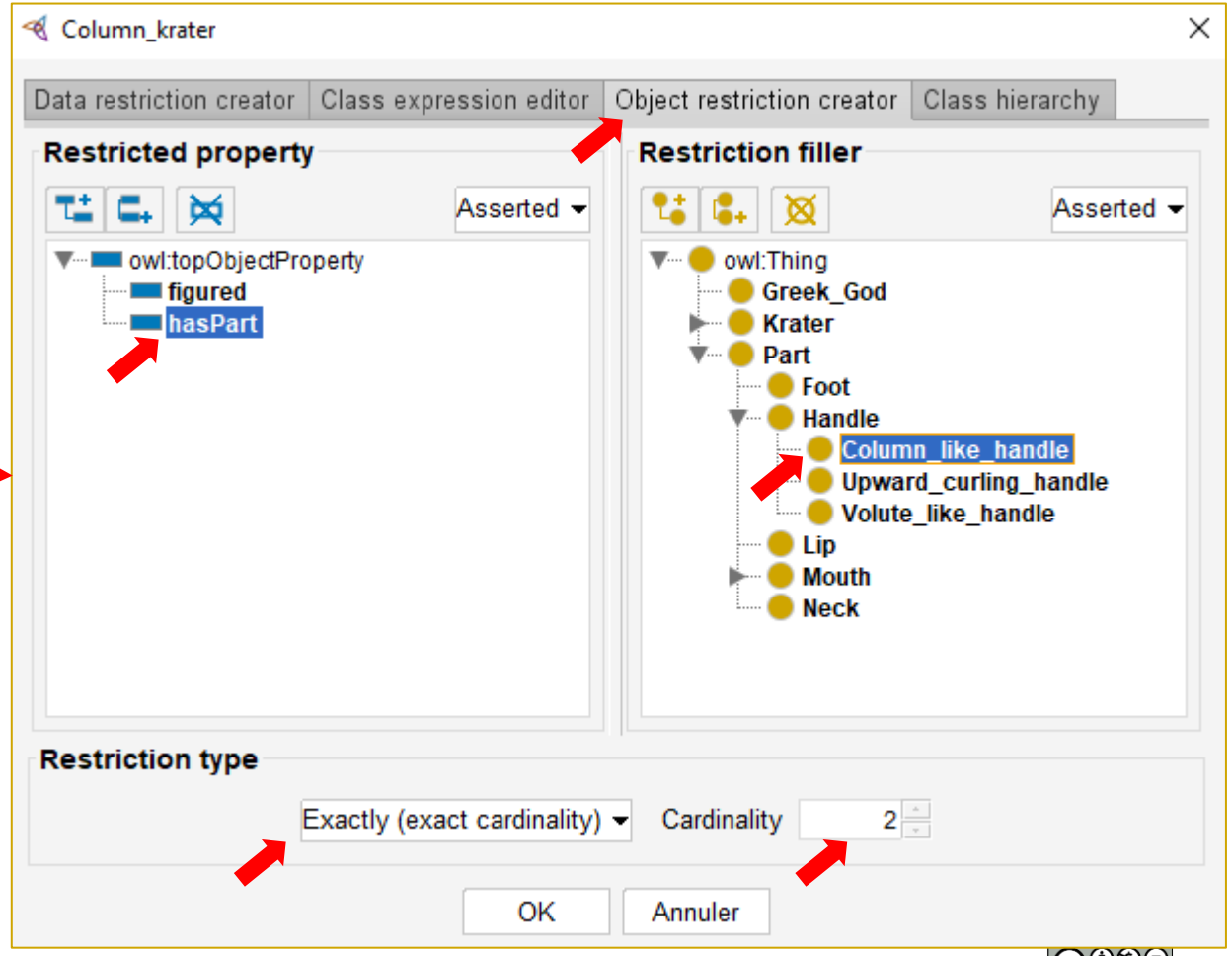
Krater

Ranges (intersection)

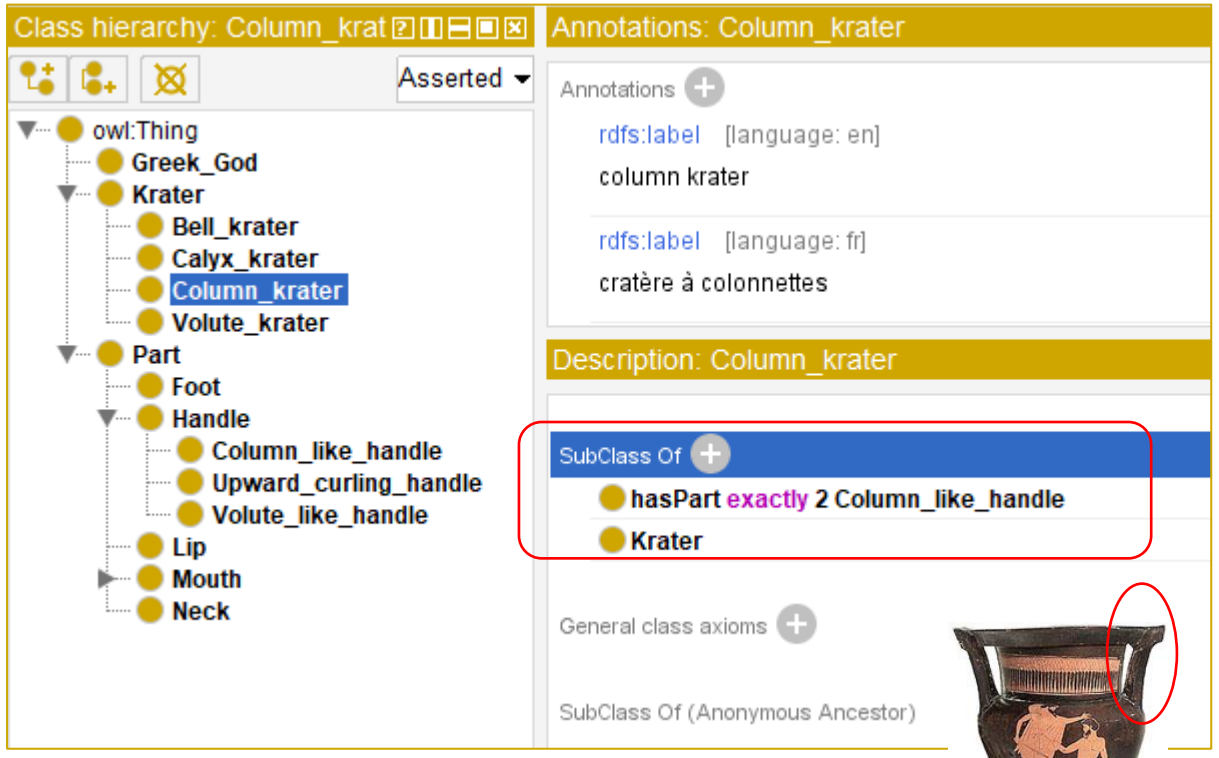
Part

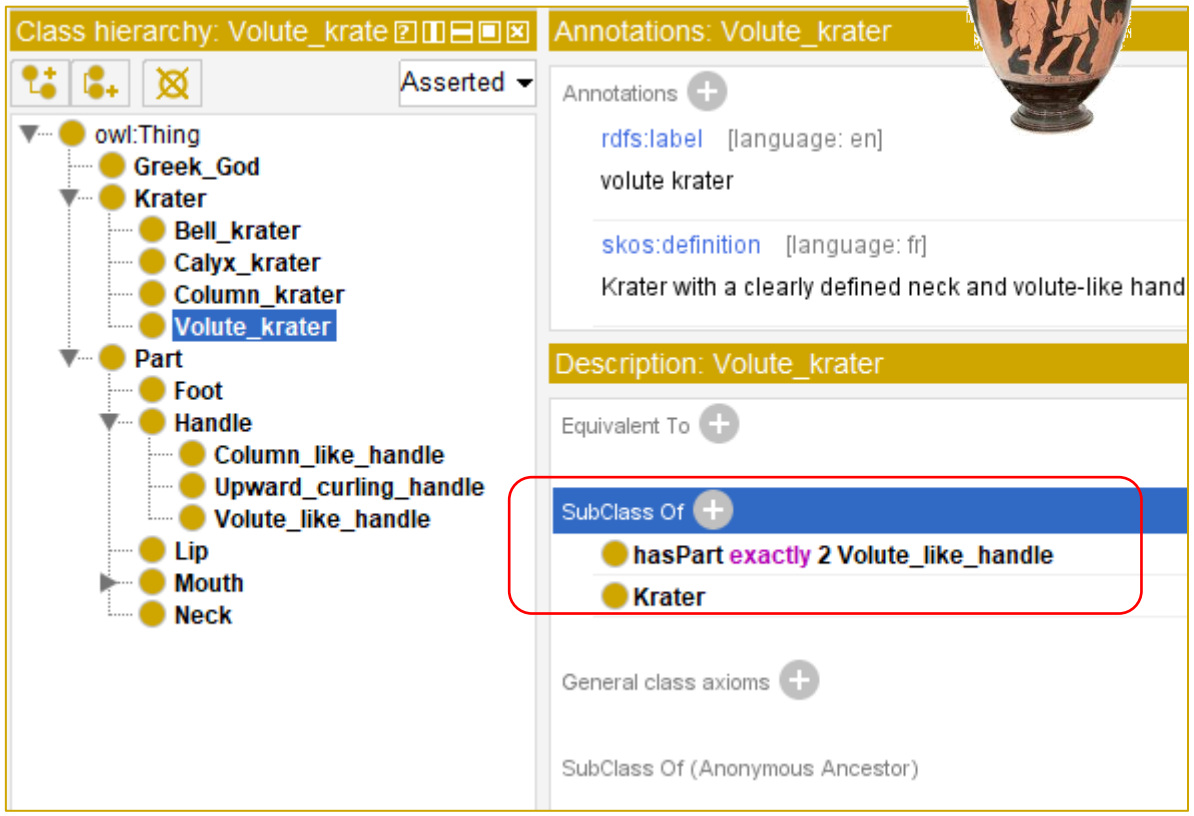
Definition of Krater Classes: Property Restrictions



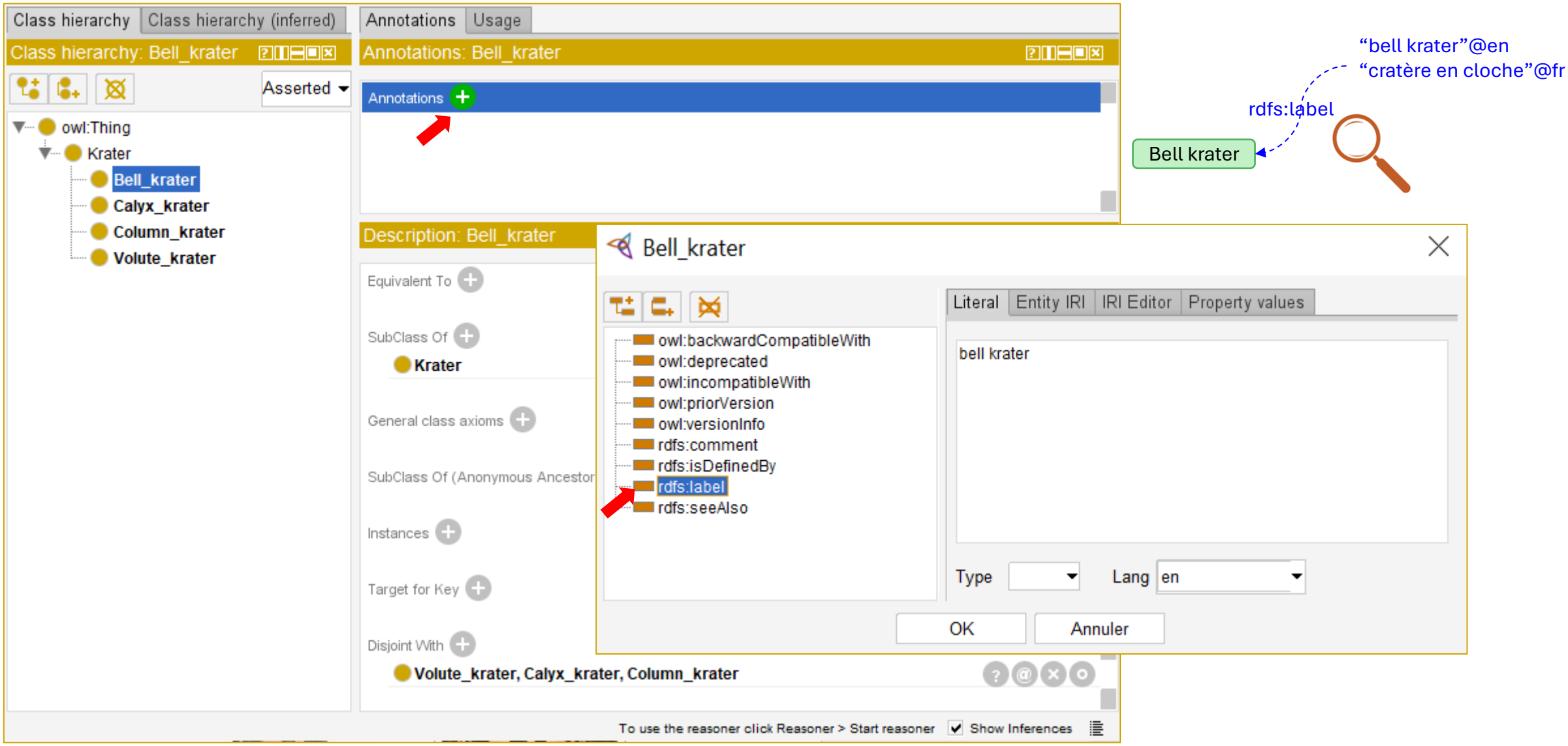


Definition of Krater Classes: Property Restrictions

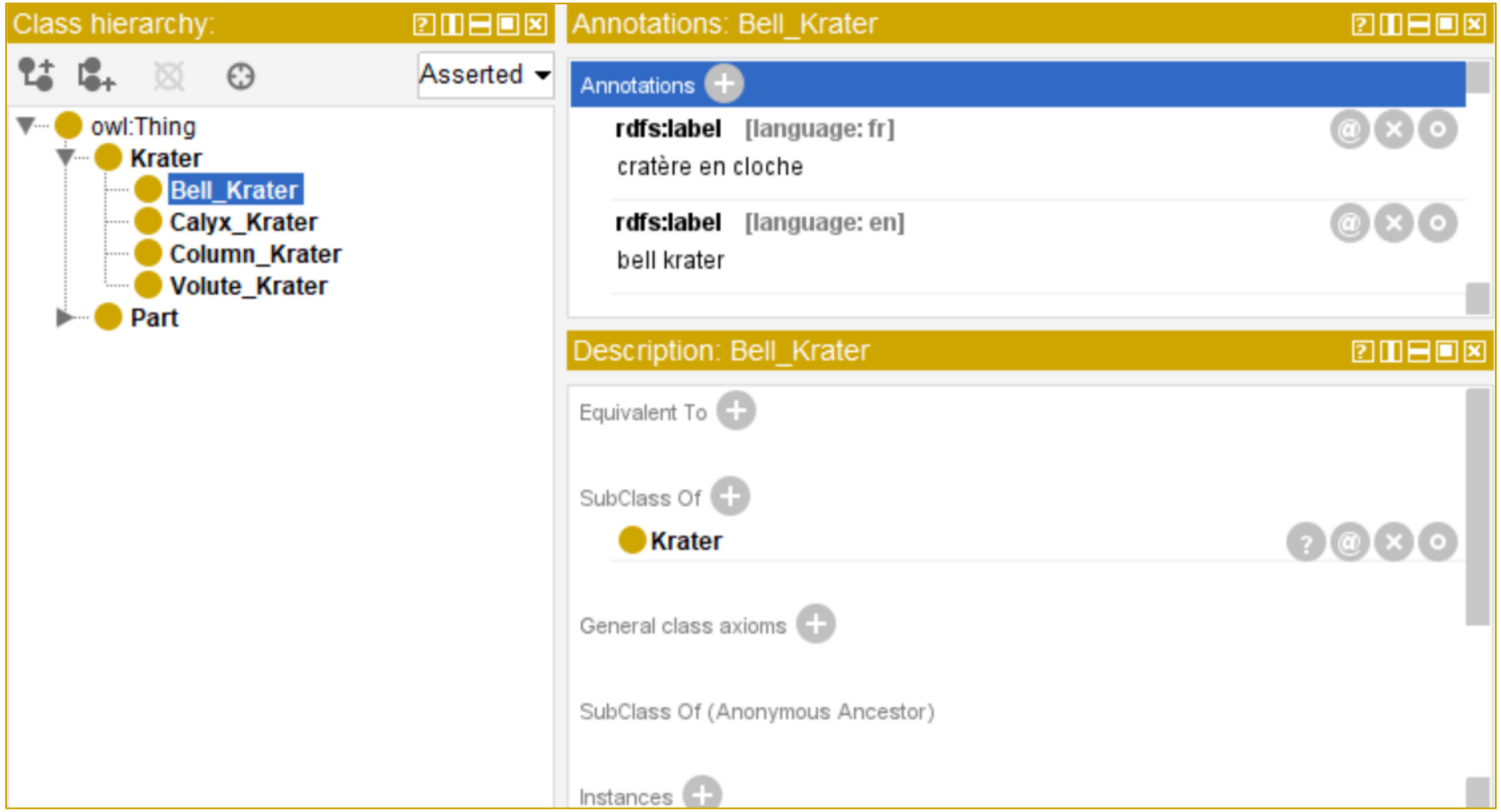


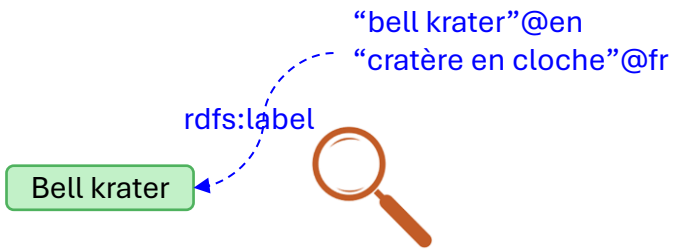


4. Ontology Building: Annotating

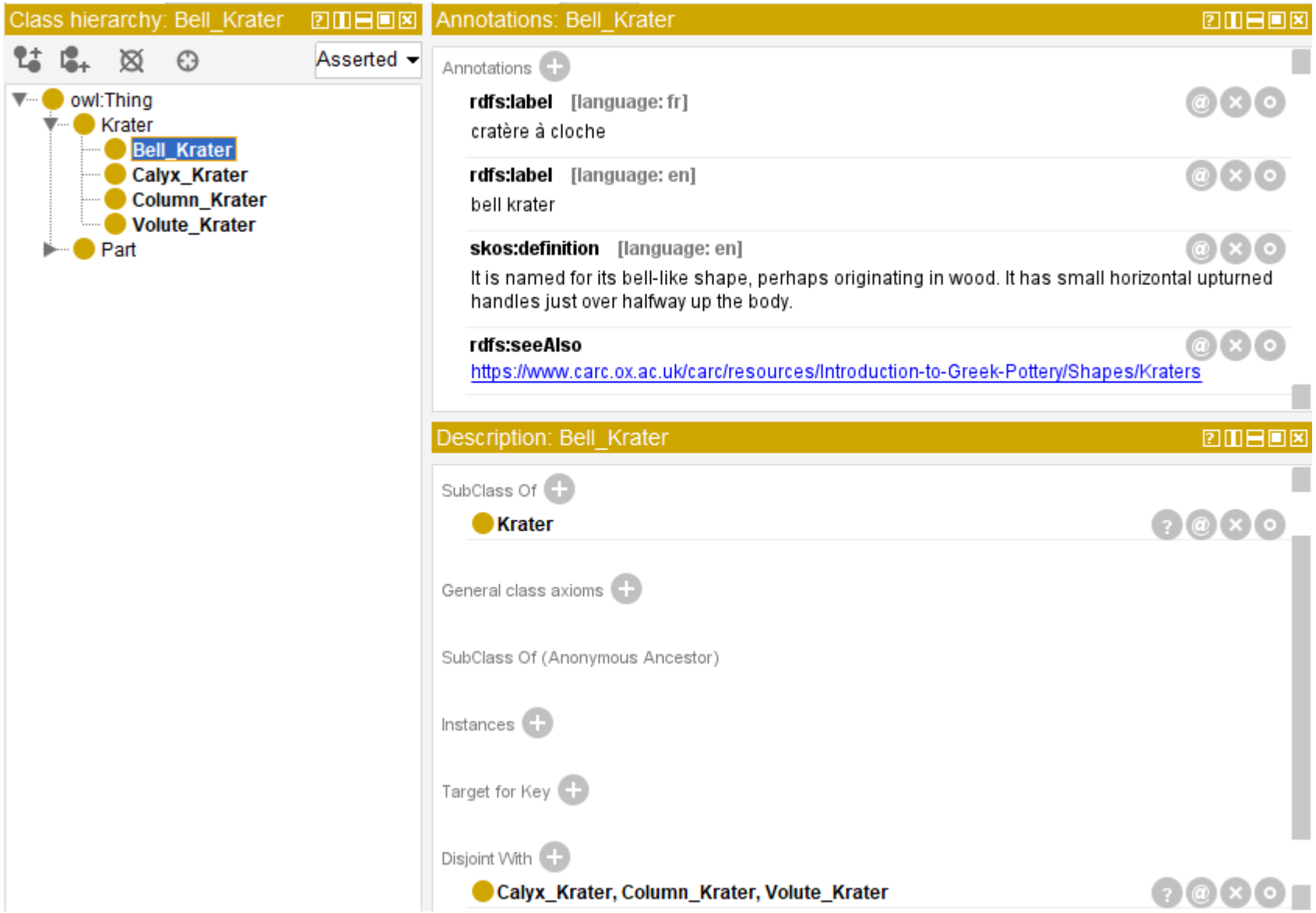


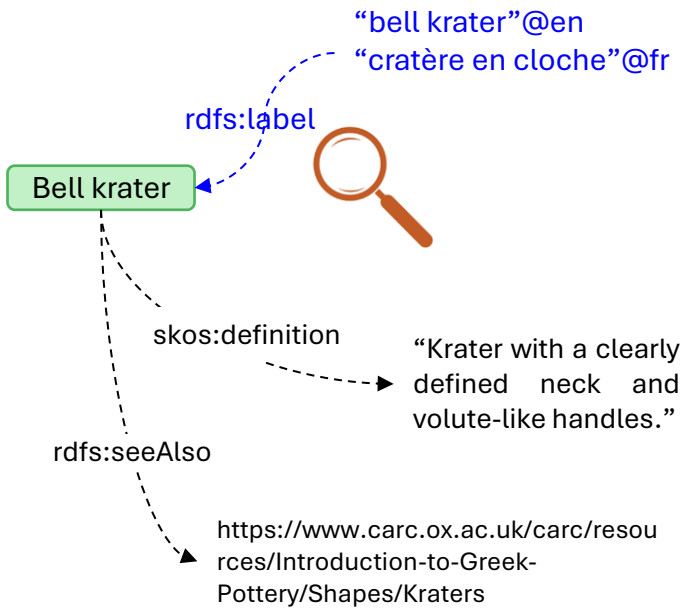
4. Ontology Building: Annotating



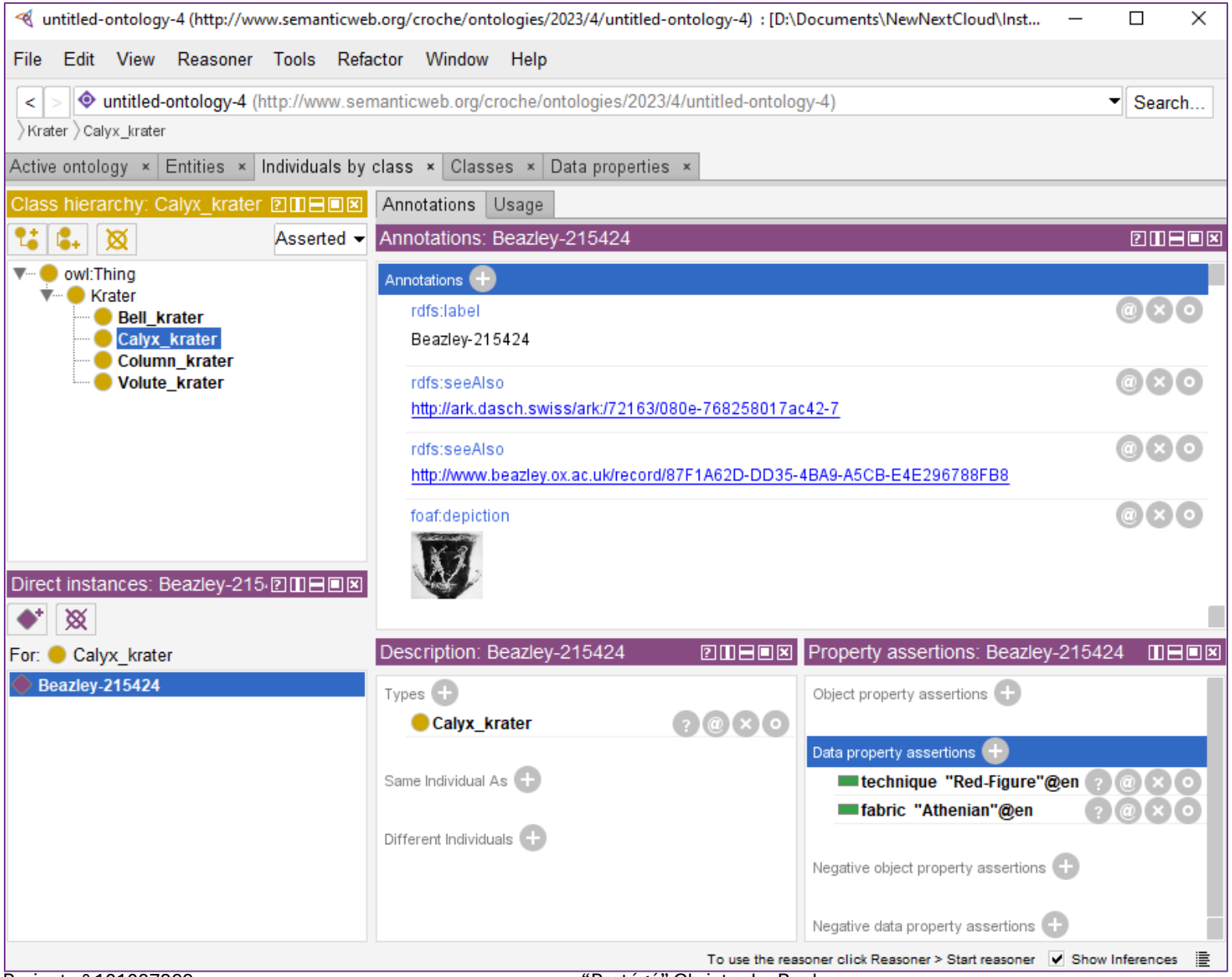


4. Ontology Building: Annotating





4. Ontology Building: Populating



Vase Number: 215424

Fabric: ATHENIAN

Technique: RED-FIGURE

Shape Name: KRATER, CALYX

Date: -450 to -400

5. Open Questions

Amphora

for storing and transport

Definition: { essential characteristics }

without neck

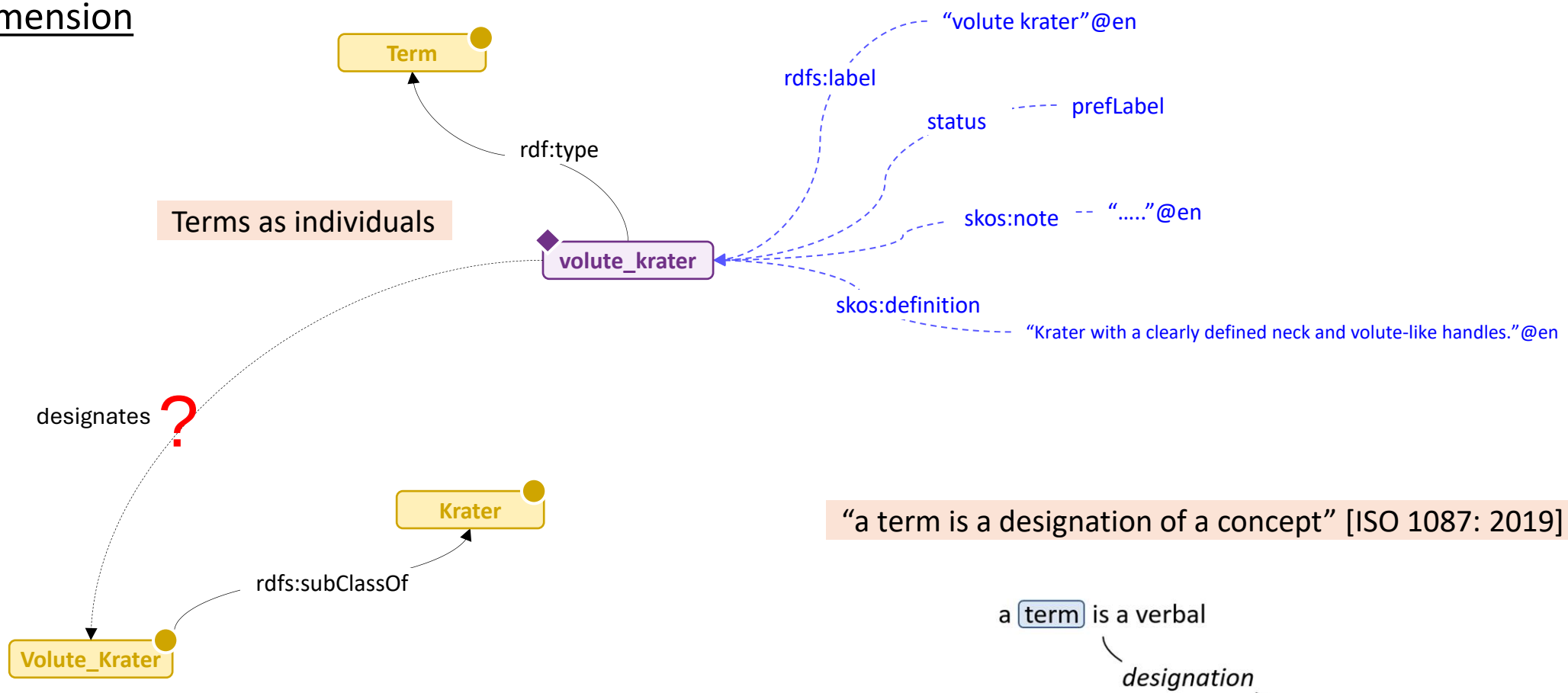
Krater

for mixing wine and water

Bell_Krater ::= { *for_mixing_wine_and_water , without_neck ,*
with_foot , with_open_mouth ,
with_upward_curling_handles }

5. Open Questions

Linguistic Dimension



"a term is a designation of a concept" [ISO 1087: 2019]

a term is a verbal *designation* of a concept

6. Conclusion

- ✓ Free environment
- ✓ Large Community
- ✓ Definition based on relations between objects (Class)
- ✓ Description Logic Reasoners
- ✓ W3C Standards compliant

- How to represent essential characteristics (Concept)?
- How to represent the linguistic dimension?
- How to take into account the way of thinking of Experts?