

A Protégé Tutorial for ENGL 795

Yetian Wang

October 15, 2018



Active Ontology Tab

- IRI
- Annotation
- Import Existing Ontologies
- Prefix

The screenshot shows the Protege software interface with the 'Active Ontology' tab selected. The browser address bar shows the URL `https://cs.uwaterloo.ca/~y15wang/pokemon-tutorial`. The 'Ontology header' section contains the 'Ontology IRI' field, which is circled in red and points to the text 'Name your ontology'. Below it, the 'Ontology Version IRI' field is shown with a placeholder example. The 'Annotations' section displays a comment in English: 'A Protege Tutorial', dated 'October 15, 2018', created by 'Yetian Wang', for 'ENGL795 Fall 2018' by 'Prof. Randy Harris', and based on the 'Pokemon Ontology' by 'Katherine Tu'. The 'Ontology imports' tab is selected and circled in red. Below it, the 'Ontology prefixes' section shows a table of prefixes. The 'owl' and 'pk' prefixes are circled in red, with 'pk' being the active one. The table lists standard RDF prefixes and the custom 'pk' prefix.

Active Ontology × Entities × Individuals by class × DL Query ×

Ontology header:

Ontology IRI `https://cs.uwaterloo.ca/~y15wang/pokemon-tutorial` Name your ontology

Ontology Version IRI e.g. `https://cs.uwaterloo.ca/~y15wang/pokemon-tutorial/1.0.0`

Annotations +

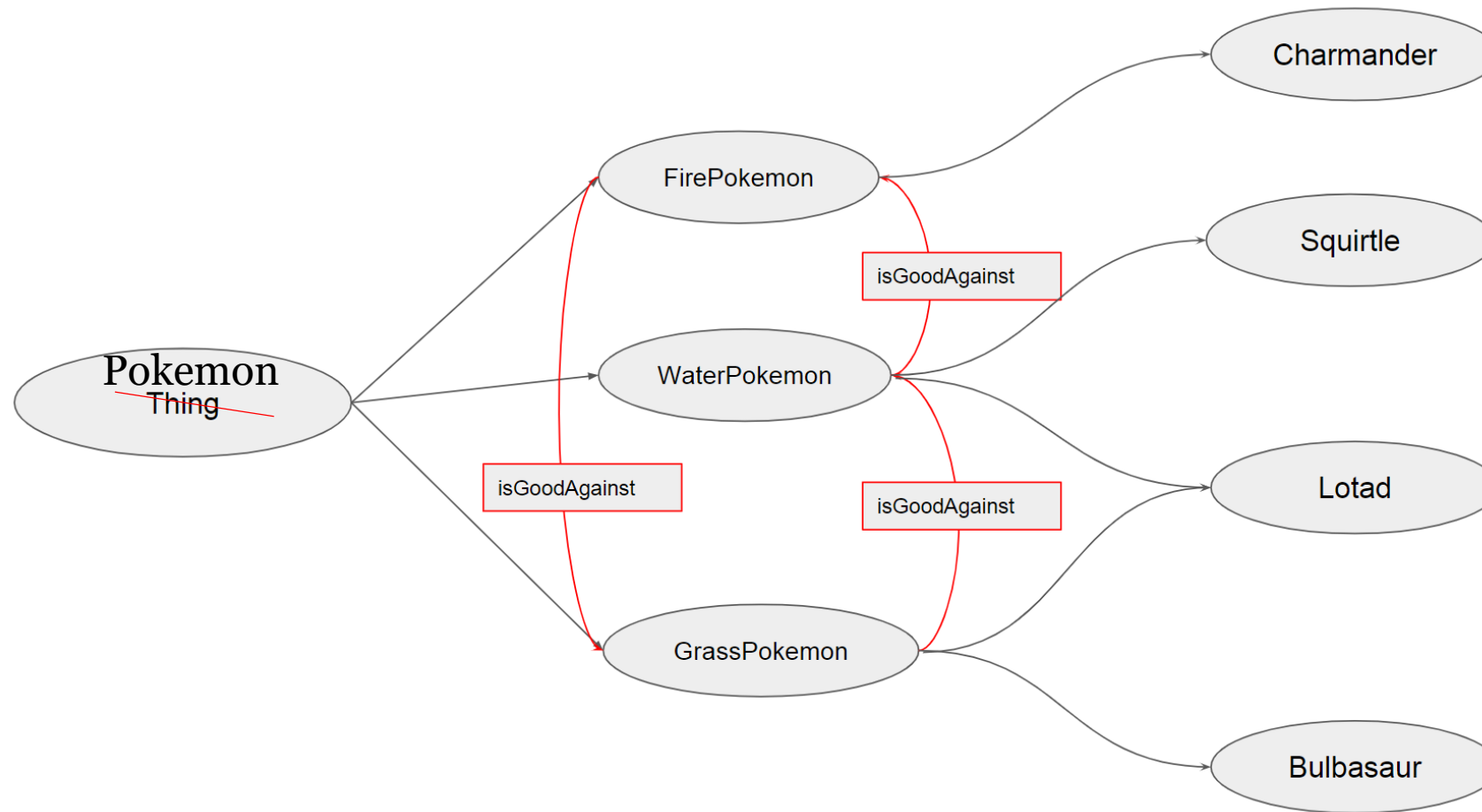
`rdfs:comment` [language: en]
A Protege Tutorial
October 15, 2018
Created by Yetian Wang
For
ENGL795 Fall 2018
Prof. Randy Harris
Based on
Pokemon Ontology
by Katherine Tu

Ontology imports × Ontology Prefixes × General class axioms

Ontology prefixes:

Prefix	
	<code>https://cs.uwaterloo.ca/~y15wang/pokemon-tutorial#</code>
owl	<code>http://www.w3.org/2002/07/owl#</code>
pk	<code>https://cs.uwaterloo.ca/~y15wang/pokemon-tutorial#</code>
rdf	<code>http://www.w3.org/1999/02/22-rdf-syntax-ns#</code>
rdfs	<code>http://www.w3.org/2000/01/rdf-schema#</code>
xsd	<code>http://www.w3.org/2001/XMLSchema#</code>

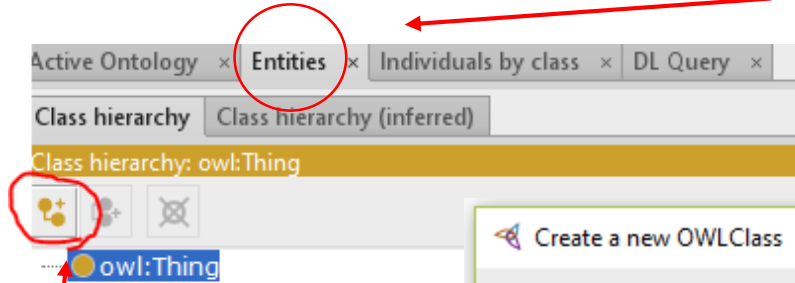
Pokemon Ontology



Creating a class

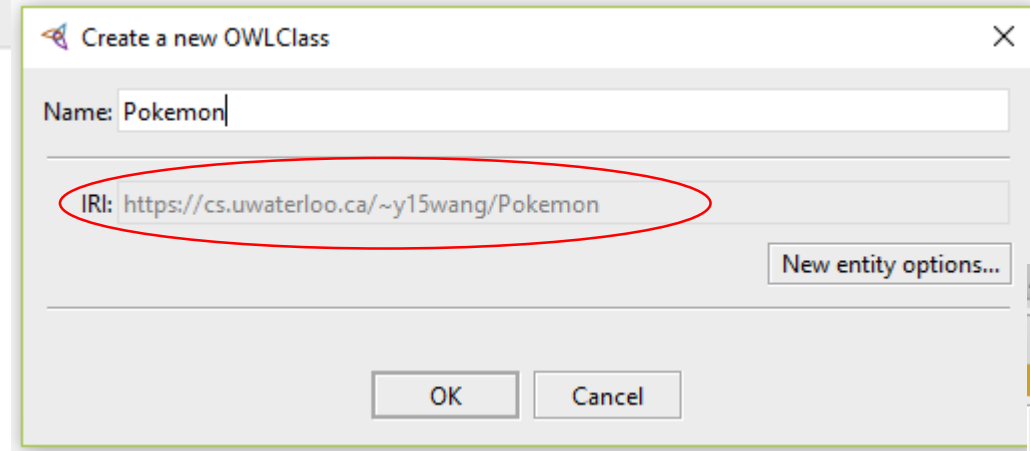
1

Go to Entities Tab



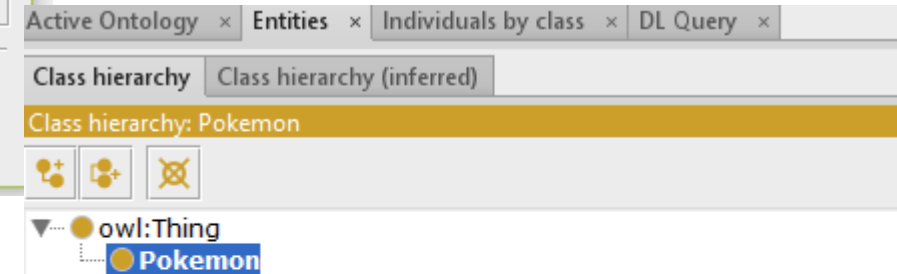
2

Click to add a class



3

Give a name. Note that the 'full name' of this class is the ontology IRI+classname



4

Class created

Creating a Subclass



1

Select Pokemon class then
create a FirePokemon class

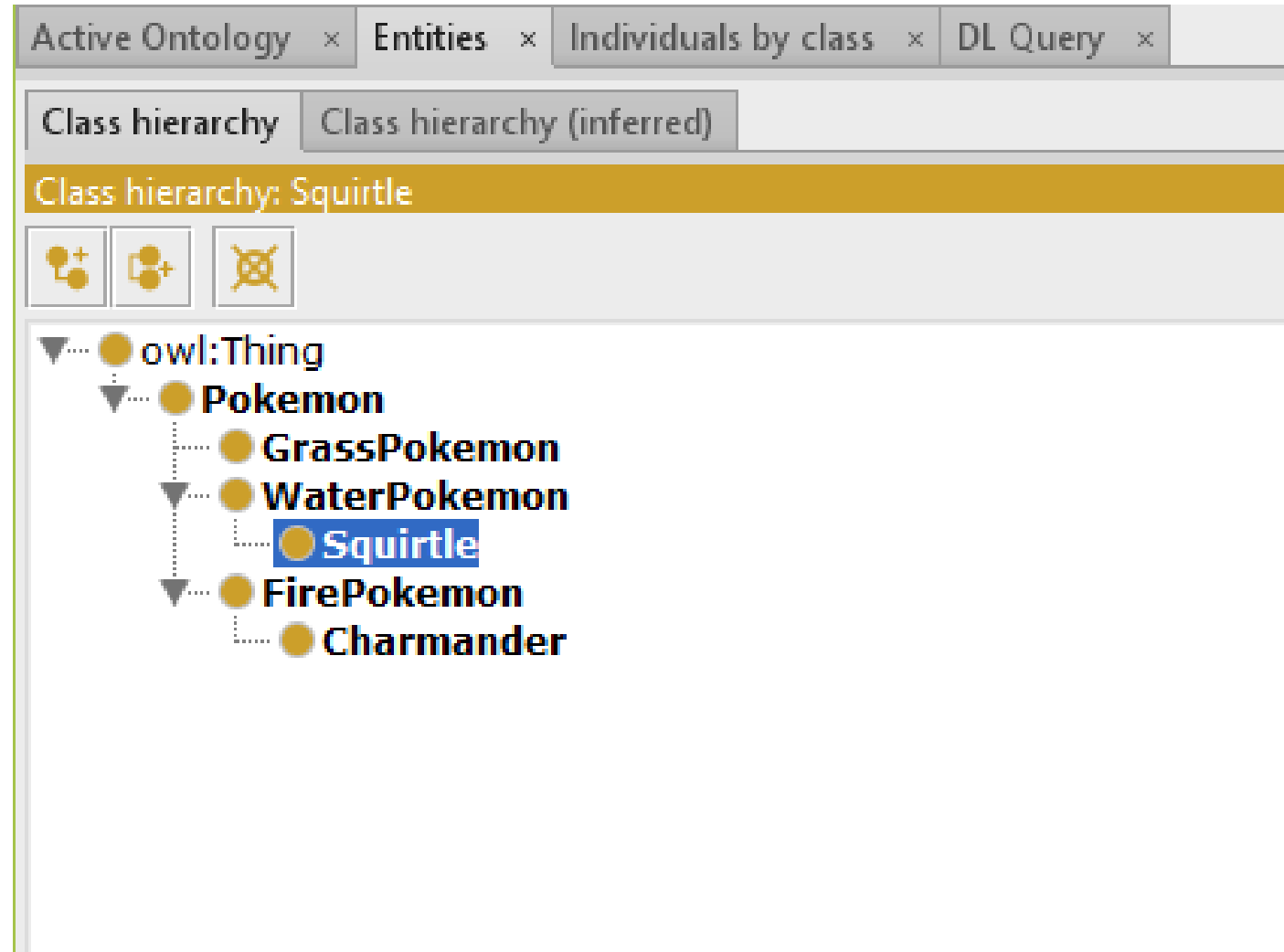
Description: FirePokemon

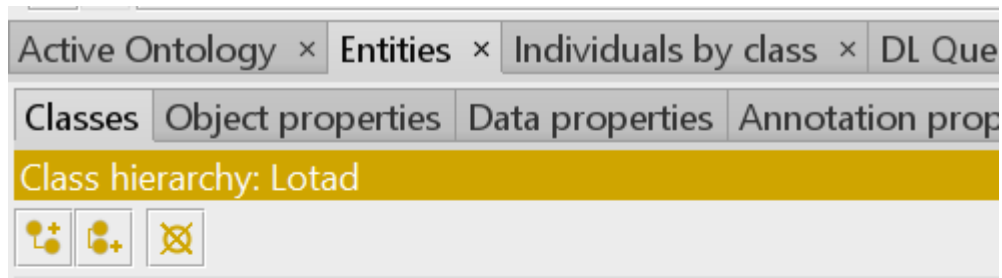
- Equivalent To +
- SubClass Of +
Pokemon
- General class axioms +
- SubClass Of (Anonymous Ancestor)
- Instances +
- Target for Key +
- Disjoint With +
- Disjoint Union Of +

2

You can modify class
hierarchy by double-click
on the Pokemon class, or
add more axioms by
clicking on the plus sign.

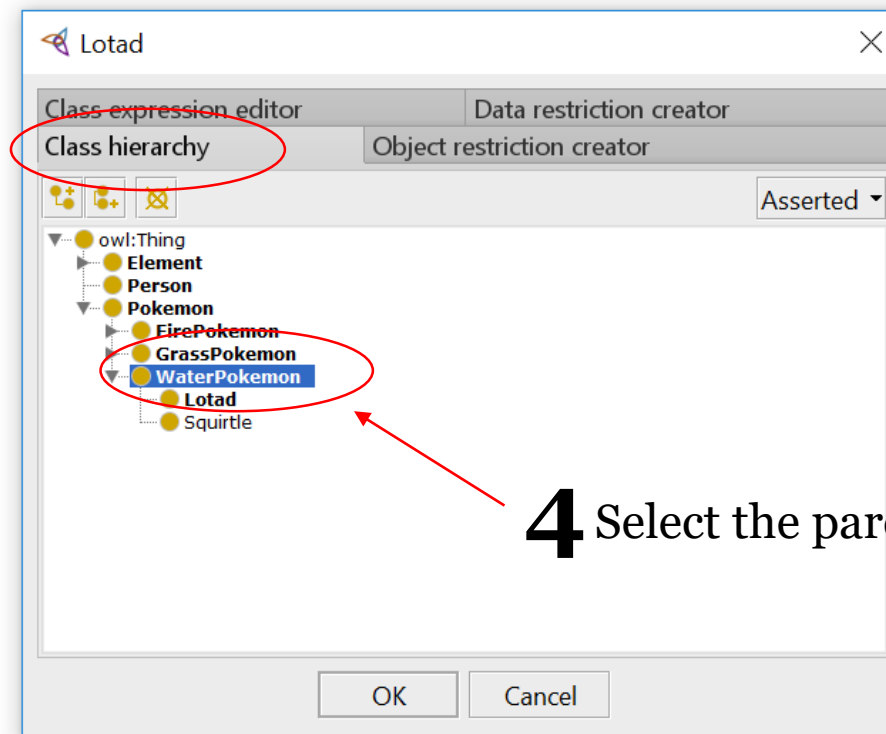
More Subclasses



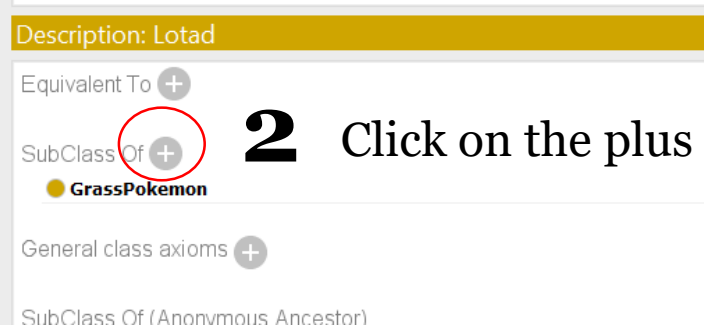


1 Make Lotad as a subclass of GrassPokemon. Select it.

3 Go to Class hierarchy tab



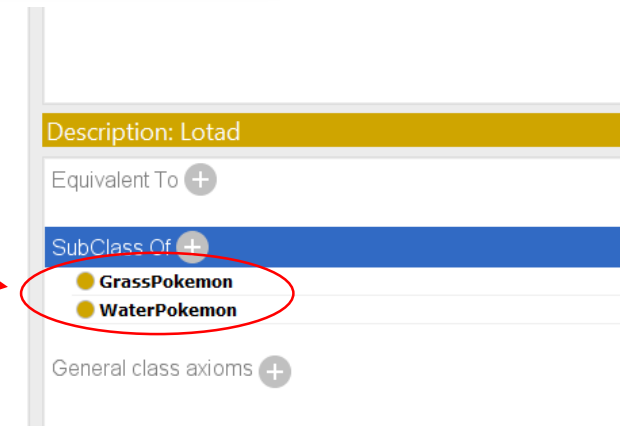
4 Select the parent class

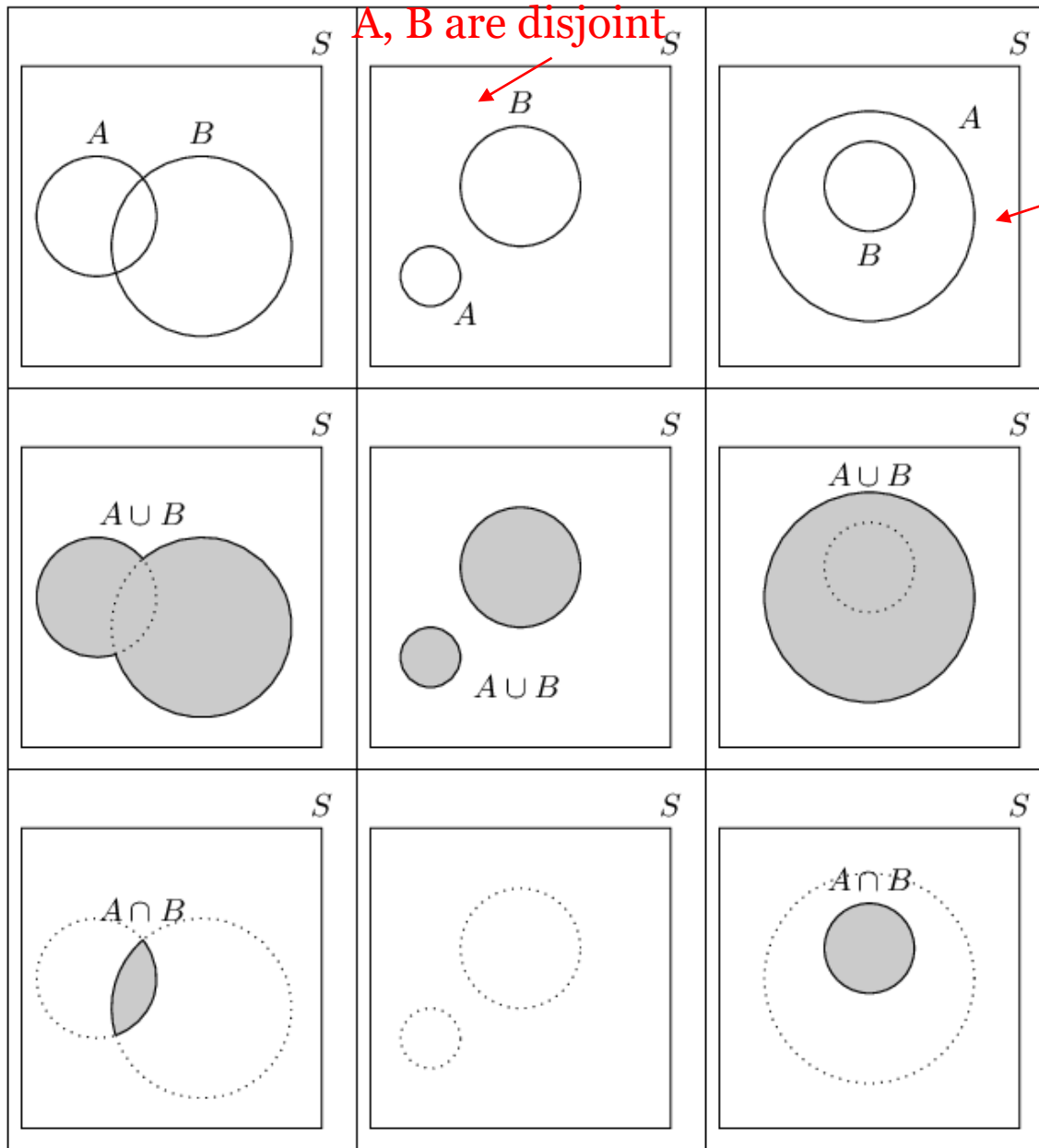


2 Click on the plus sign

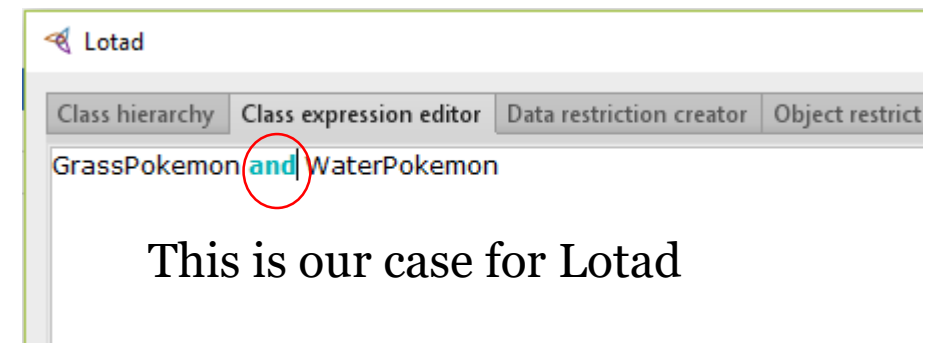
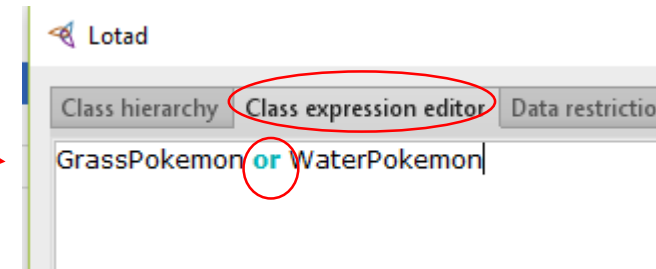
5 You should see this. This is equivalent to

‘GrassPokemon **and** WaterPokemon’





B is a subclass of A



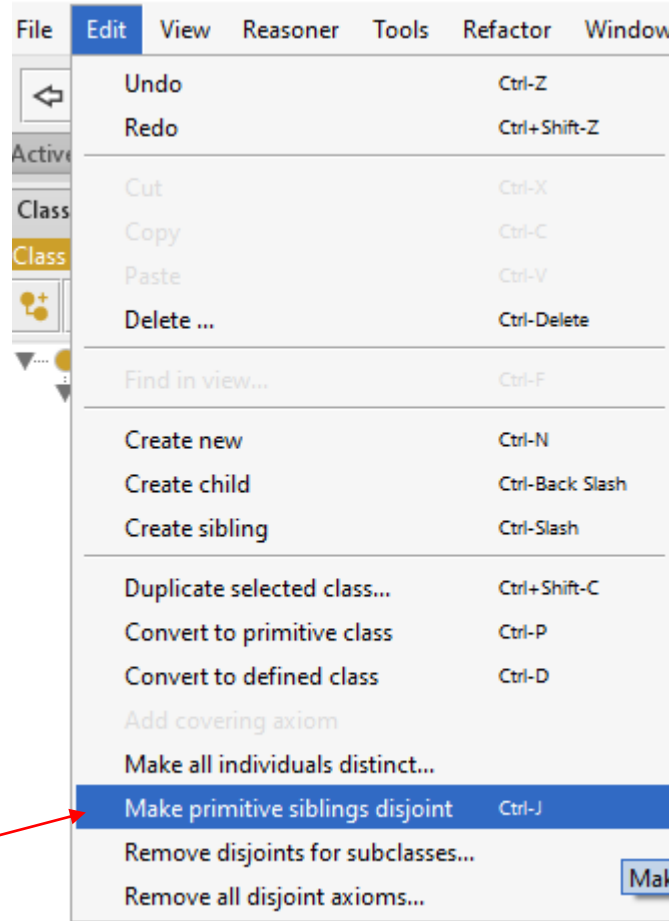
This is our case for Lotad

Disjoint Classes

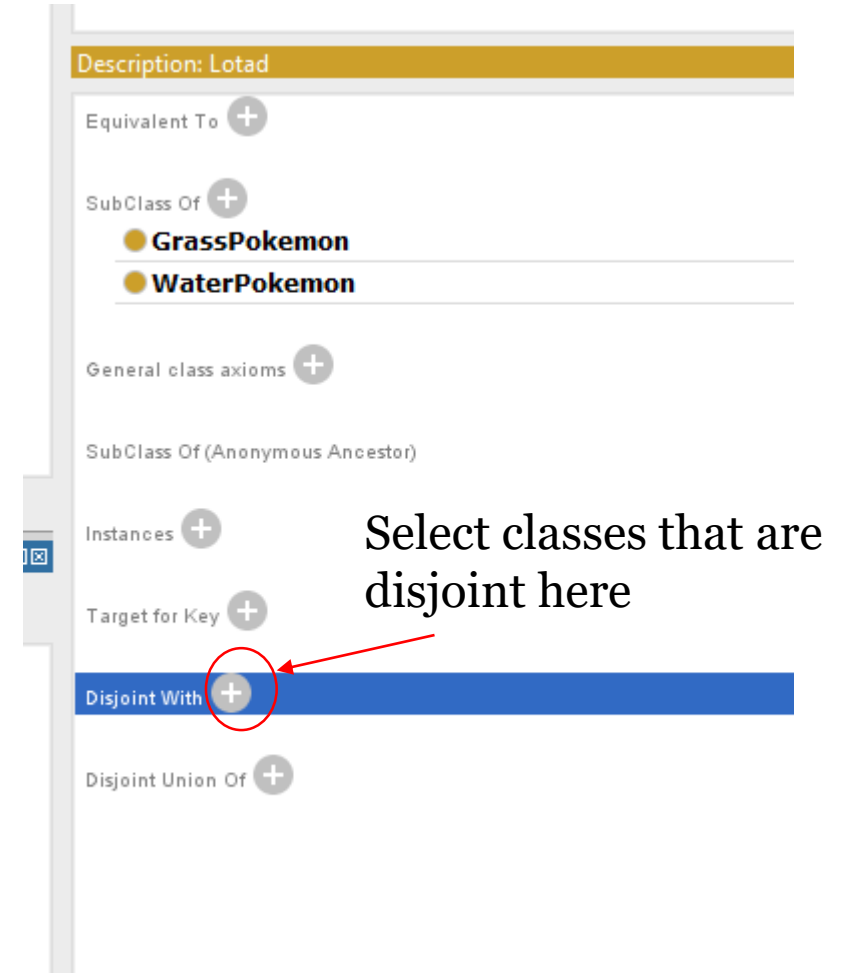
- Class names are different ≠ classes are different
- You need to specify this in your ontology
- But be careful with **inconsistencies**

Select a class. This option makes all sibling classes on that level disjoint

Option 1 – may not appear in newer versions



Option 2



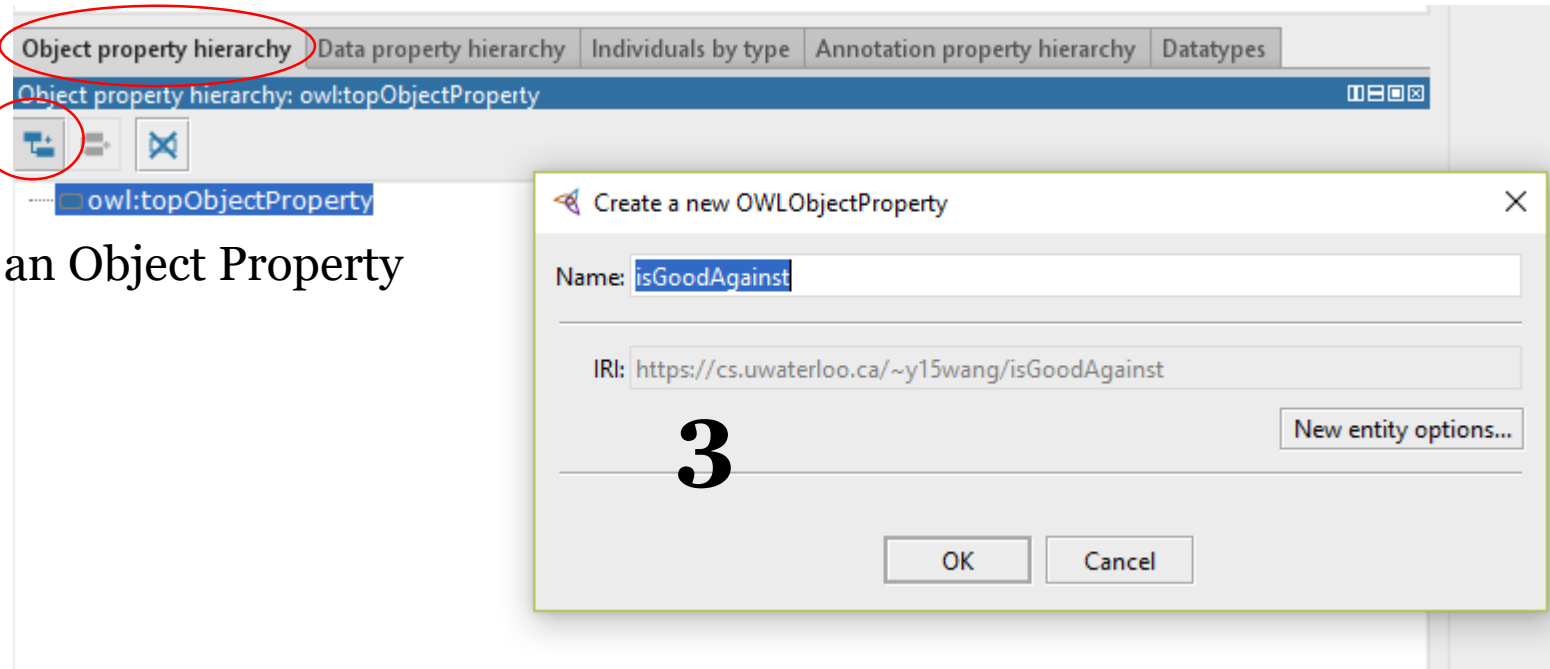
Properties

- Object Properties
 - Properties that relate instances of a **class** to instances another **class**
- Data Properties
 - Properties that relate instances of a **class** to **values** (e.g. string and numerical values)

Create a Property

1 Go to Object Properties Tab

2 Add an Object Property



Domain and Range

- Domain
 - the subjects of such property statements must belong to the class extension of the indicated class description [*]
- Range
 - that the objects/values of this property must belong to the class extension of the class description or to data values in the specified data range [*].
- Example
 - suppose we have a property '*takesCourse*' and an axiom (**Students** *takesCourse* **Course**)
 - The domain is the class **Student**
 - The range is the class **Course**

[*] <https://www.w3.org/TR/owl-ref/#domain-def>

Specify Domain and Range

1 Select a property then click the plus sign

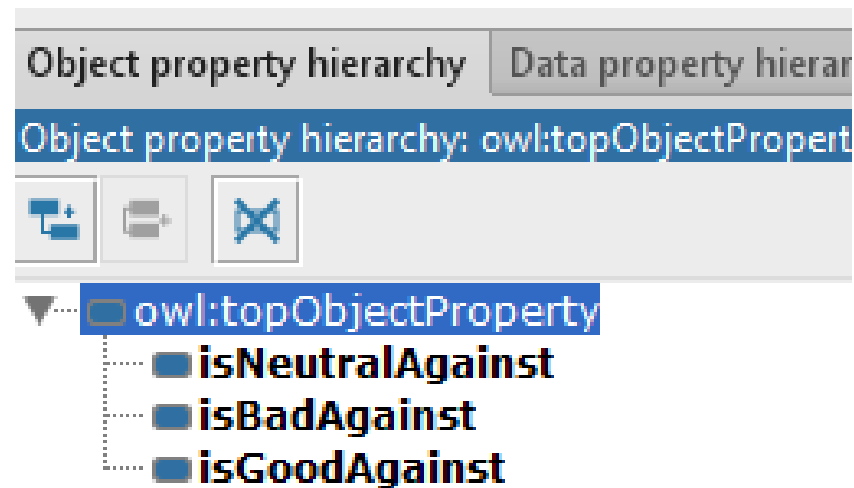
2 Go to Class hierarchy tab

3 Select the class for domain and range

4 Source and Target

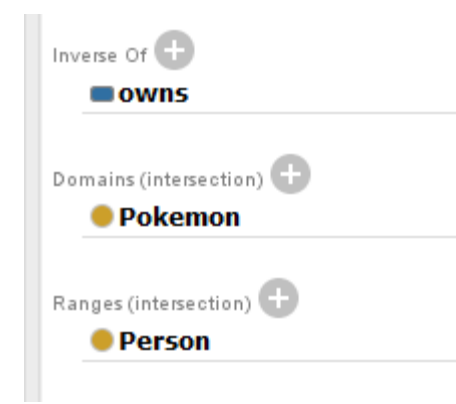
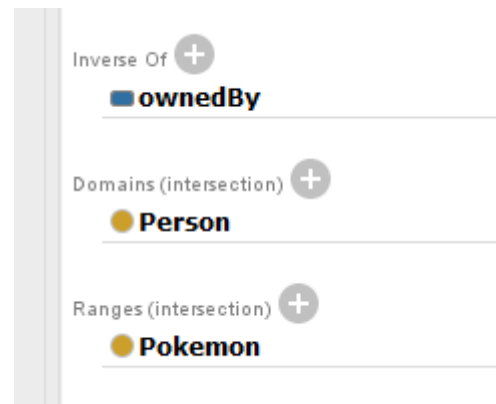
The screenshot shows the OWL editor interface for the property 'isGoodAgainst'. The left sidebar lists property types: Equivalent To, SubProperty Of, Inverse Of, Domains (intersection), Ranges (intersection), Disjoint With, and SuperProperty Of (Chain). The 'Domains (intersection)' and 'Ranges (intersection)' items are circled in red. The central window shows the 'Class hierarchy' tab with a tree structure: owl:Thing, Element, Person, Pokemon, FirePokemon, GrassPokemon, and WaterPokemon. The 'Pokemon' class is selected. The right sidebar shows the 'Description: isGoodAgainst' with the same property types. The 'Domains (intersection)' and 'Ranges (intersection)' items are circled in red, and red arrows point to them from the text 'Source and Target'.

Create the remaining properties and specify their domain and range



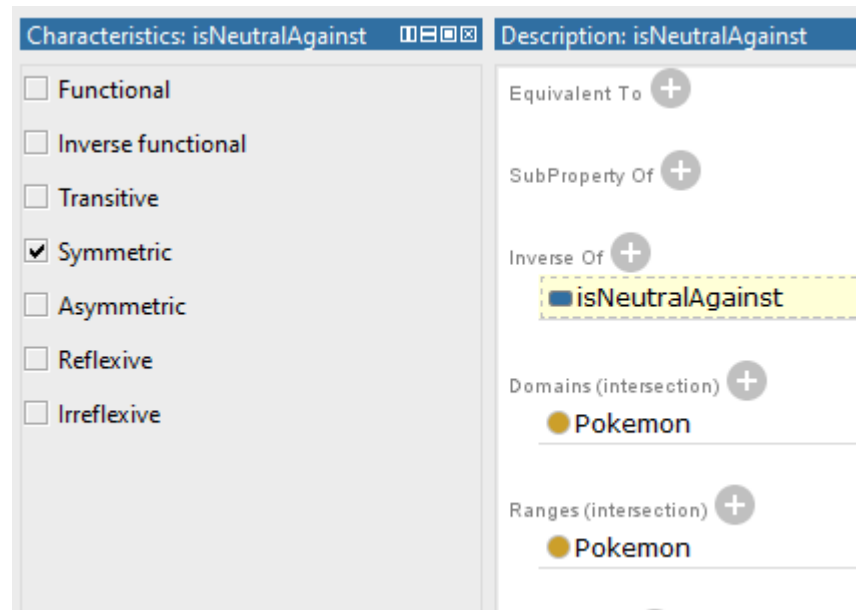
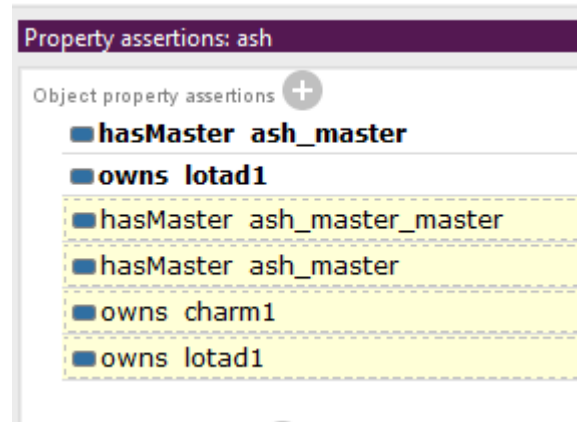
Sub-properties and Inverse Properties

- Sub-properties
 - hasMother and hasFather are sub-properties of hasParent
- Inverse
 - If (A, p, B) then (B, inverse of p, A)
 - E.g. hasParent, hasChild



Property Characteristics

- Functional
 - One individual only
 - E.g. hasHusband
- Transitive
 - If (A, p, B) and (B p C) then (A p C)
 - E.g. has Ancestor, hasMaster
- Symmetric
 - If (A, p, B) then (B, p, A)
 - E.g. marriedTo
- Reflexive
 - relates everything to itself



Specify Class Axioms

WaterPokenmon is good against FirePokemon

Necessary and Sufficient:

It's a WaterPokenmon iff it is good against FirePokemon'

e.g. a female who has at least 1 child is equivalent to a mother

Necessary:

WaterPokenmon is a subclass of 'things that are good against FirePokemon'

e.g. a female who has at least 1 child (i.e. mother) is a parent. But not all parents are females (e.g. fathers)

Select the class
WaterPokemon then
1 click on the plus sign

Select this tab

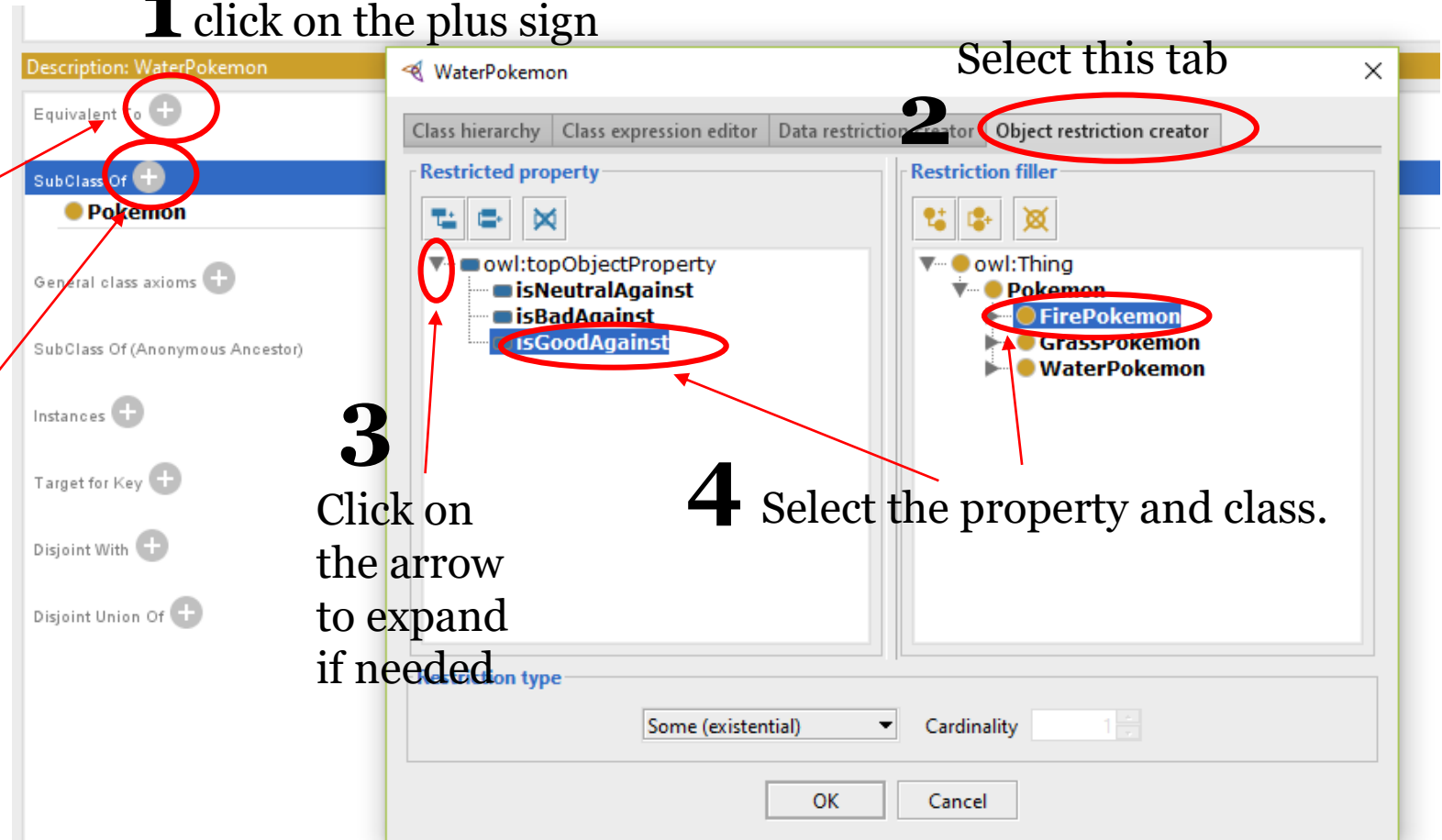
2

3

Click on
the arrow
to expand
if needed

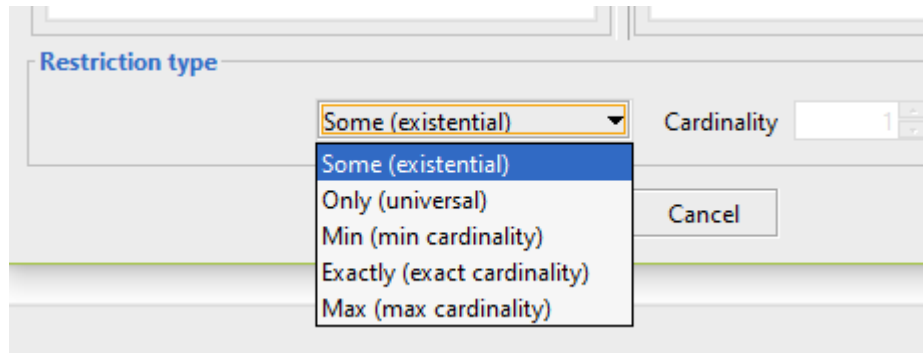
4

Select the property and class.



Restriction Type

- Some
 - There exists a FirePokemon that WaterPokemon isGoodAgainst
- Only
 - WaterPokemon is only good against FirePokemon
- Cardinality
 - There is $\langle \text{min, exactly, max} \rangle$ n FirePokemon that a WaterPokemon isGoodAgainst

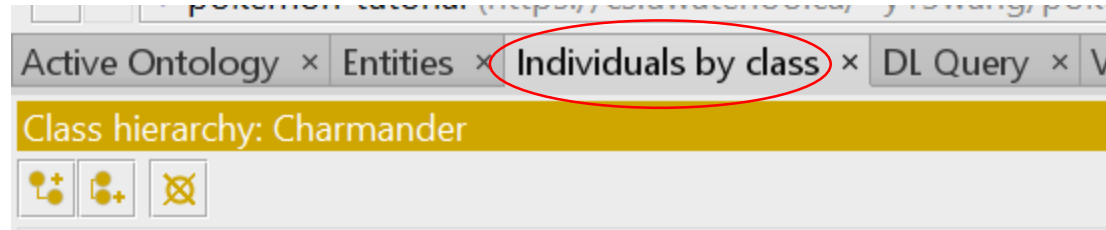


A screenshot of a software interface for defining a restriction type. The dialog box has a title bar and a main area with the text "Restriction type" in blue. Below this text is a dropdown menu currently showing "Some (existential)". To the right of the dropdown is a "Cardinality" field with a value of "1". Below the dropdown menu is a list of options: "Some (existential)", "Only (universal)", "Min (min cardinality)", "Exactly (exact cardinality)", and "Max (max cardinality)". A "Cancel" button is located to the right of the list.

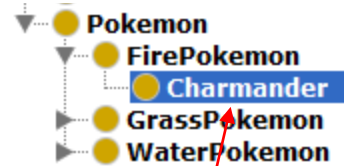
Individual

- Instances of classes.
- The Data
- E.g.
bobTheCharmander
is an instance of the
class **Charmander**
- Then create
emilyTheSquirtle as
an instance of
Squirtle

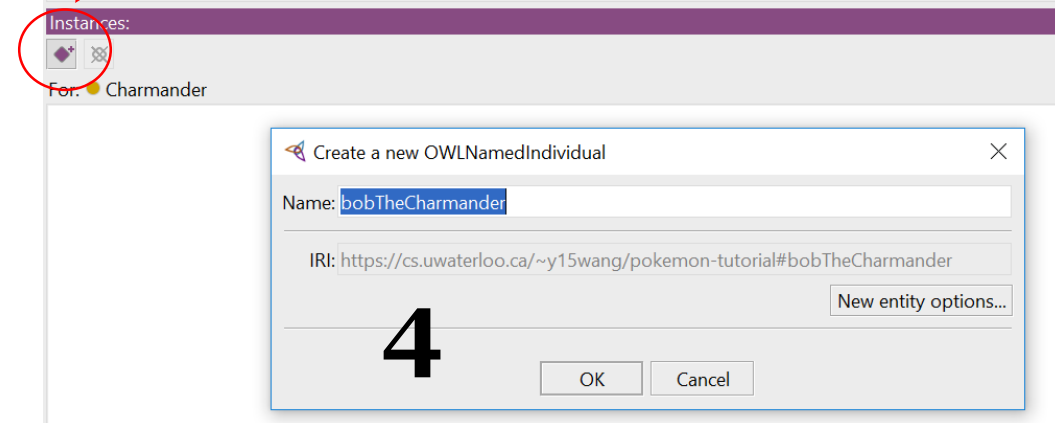
1 Go the Individual Tab



3 Create an instance

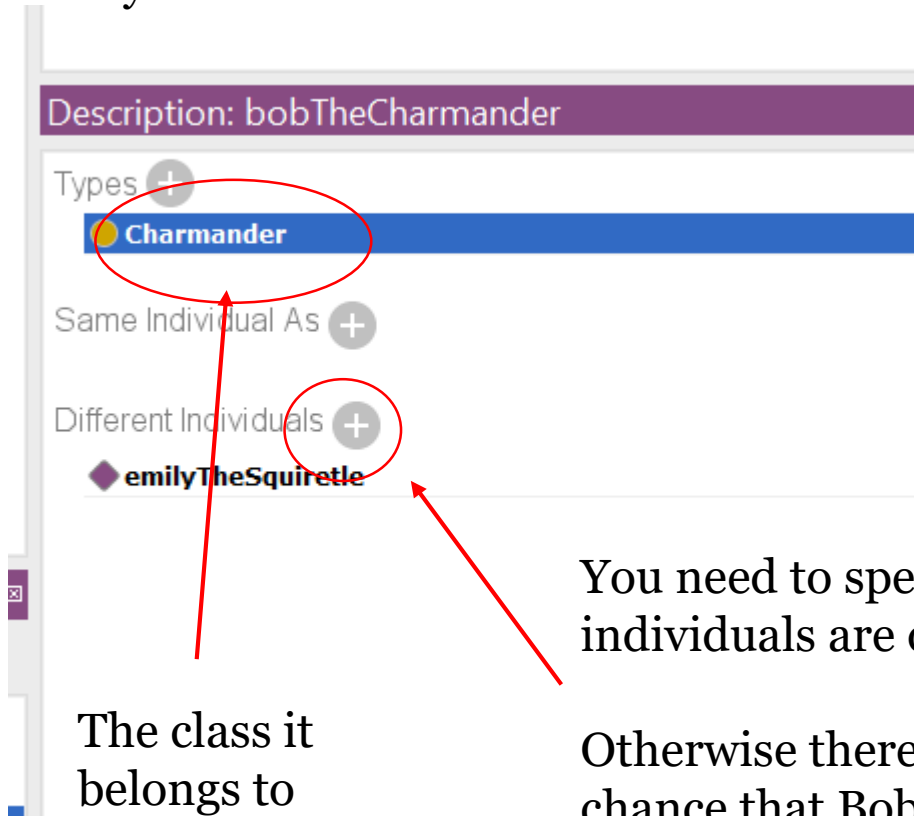


2 Select the
Charmander class



Specify Individual Description

1 Select the instance *bobTheCharmander* then you should see this

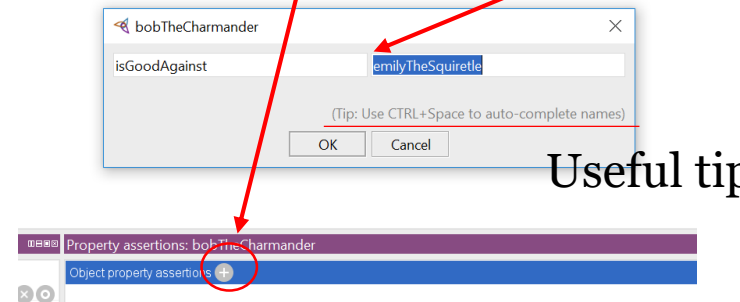


The class it belongs to

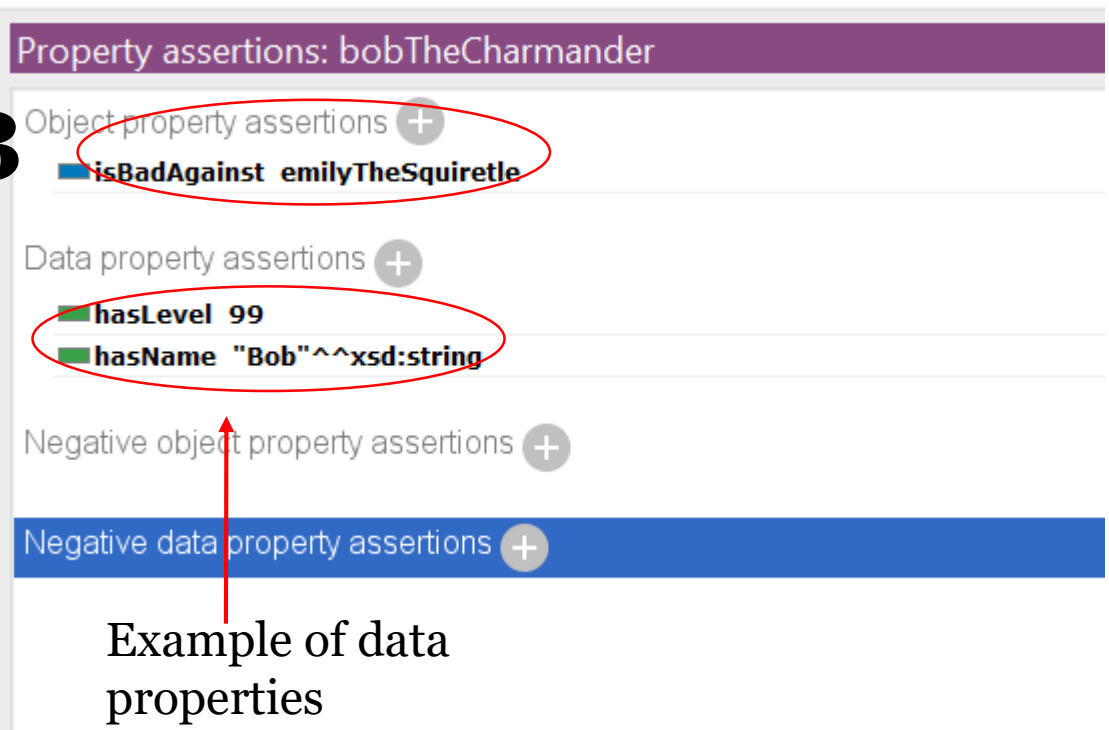
You need to specify if individuals are different.

Otherwise there is a chance that Bob and Emily are the same Pokemon

2 Add an object property assertion then type the property and individual names

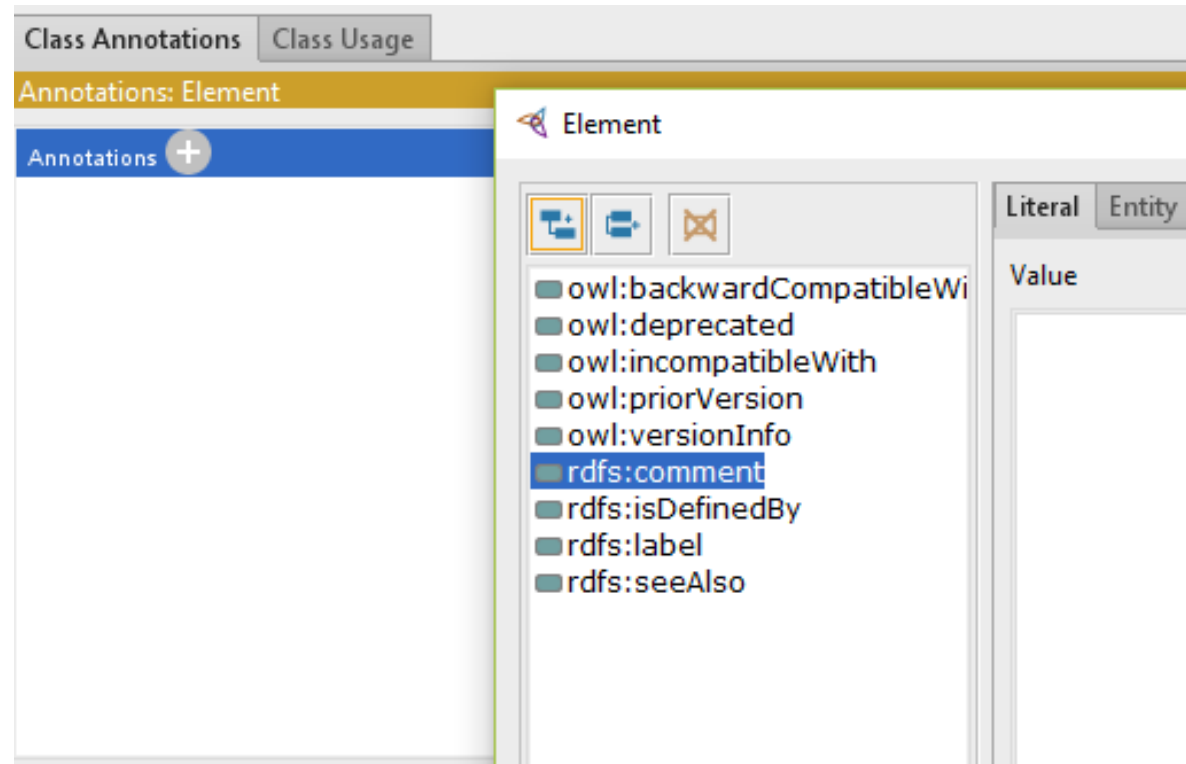


3 Object property assertions +



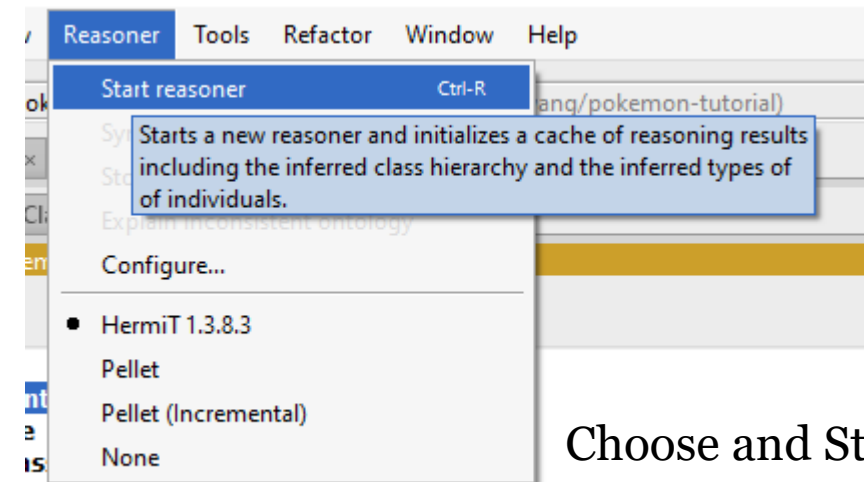
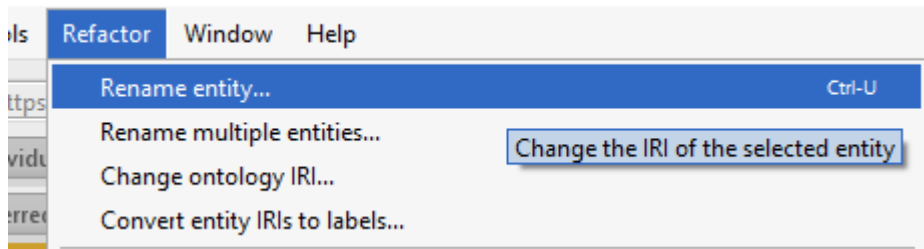
Annotation

It is important to describe what you are trying to do with each class, property, and individual in your ontology.



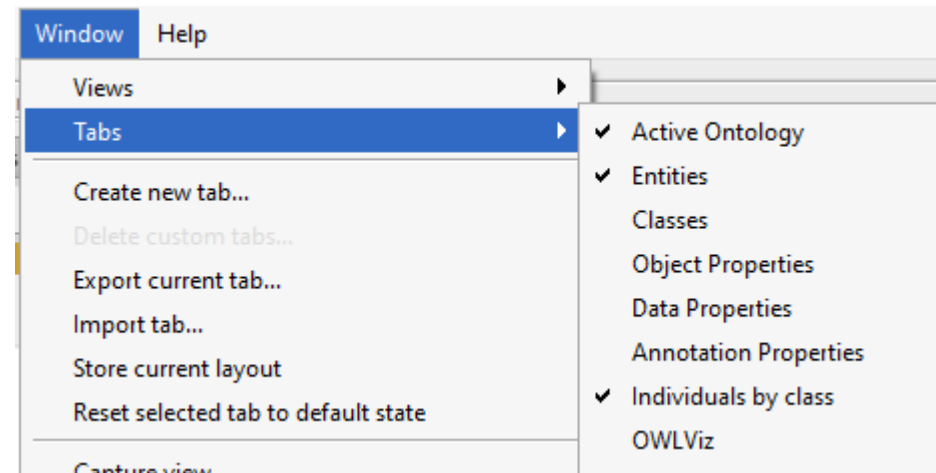
Other Features

Rename and change IRI

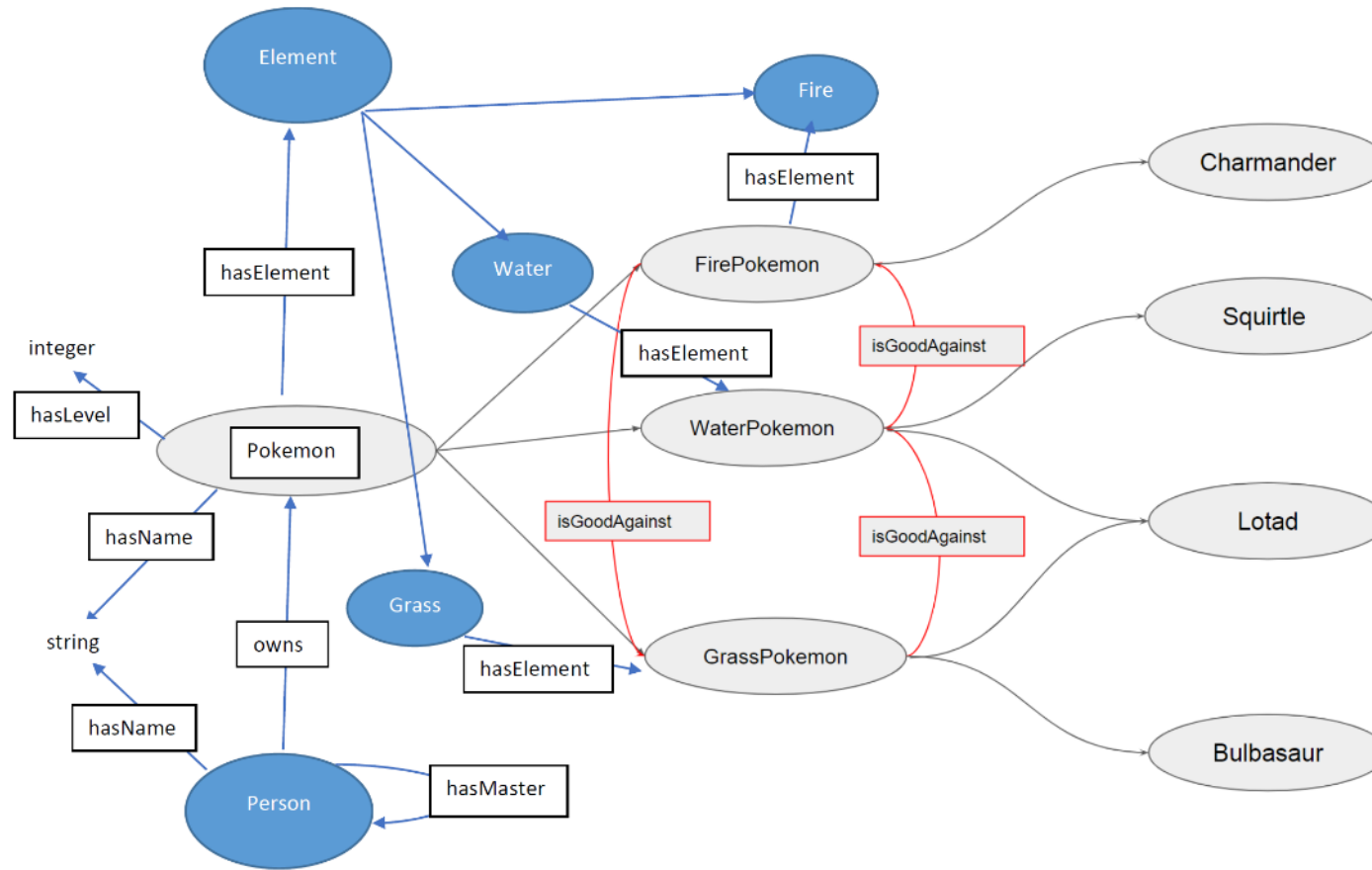


Choose and Start a Reasoner

Customize View and Tabs



Extended Pokemon Ontology



Thank You

- A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools
 - http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf
- Pizza Tutorial
 - https://cwi.unik.no/images/e/ef/UNIK4710-Protege_Presentation.pdf
- yetian.wang@uwaterloo.ca