Ontology101Tutorial Documentation Release 1.0

Nicole Vasilevsky

Jan 10, 2020

Contents:

1	Initial Preparation 1.1 Cloning the GitHub repository 1.2 Dowloading Protégé	1 1 1
2	Starting Protégé 2.1 The Protégé UI	3 6
3	The entities tab3.1Creating your first class3.2Renaming an entity3.3New entities3.4Adding annotations properties3.5Setting label rendering3.6Creating the class hierarchy	 11 12 15 16 18 20 21
4	EXERCISE: Basic Subclass Hierarchy 4.1 The Class description view	23 28
5	Protégé plugins5.1Annotation search plugin (for older versions of Protege)	31 32
6	Disjointness 6.1 Reasoning and inconsistency checking	35 36
7	Object properties 7.1 Create an object property	41 41
8	OWL class restrictions 8.1 Superclass restrictions	45 45
9	EXERCISE: Basic Restictions	49
10	DL query tab	53
11	EXERCISE: Basic DL Queries 11.1 Equivalent classes	59 65
12	Automatic classification	67

13	EXERCISE: Basic classification	69
14	EXERCISE: More basic classification	73
15	Object Properties	75
16	EXERCISE: Domains and Ranges	77

CHAPTER 1

Initial Preparation

You will need to clone this repository to download the exercise files in the BDK14_exercises folder.

Associated exercise files include:

- basic-classification
- basic-disjoint
- · basic-dl-query
- basic-restriction
- basic-subclass
- domain-range
- taxon-union

These exercises were tested under Protégé 5.1. Note: some screenshots may appear different if you are using a prior version of Protégé.

1.1 Cloning the GitHub repository

Instructions to clone a repository using the command line are available here.

You can also install a graphical user interface like Sourcetree or GitHub Desktop. Detailed instructions for downloading the repository in Sourcetree are here.

1.2 Dowloading Protégé

To download Protégé, go to: http://protege.stanford.edu/

CHAPTER 2

Starting Protégé

When you start Protégé you are presented with a default empty ontology.

To open Protege on the command line, navigate to your directory where your application is stored. See example below:

cd /Applications cd Protege-5.1.0/

Type the command open Protege.app to open the Protégé application.



We will begin by clicking into the Ontology IRI field and providing an IRI. The IRI will be used to identify our ontology on the Web. You can set the IRI to anything you want at this stage, for this tutorial we will use "http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl"

ao-workshop (http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl) : [http://www	.semanticweb.org/vasilevs/ontologies/2018/1/u	ntitled-ontology-176
> @ ao-workshop (http://purl.obolibrary.org/obo/owl-tutorial/ao-w	orkshop.owl)	Search
Active Ontology × Entities × Individuals by class × DL Query ×		
Ontology beader u	BBB Ontology metrics:	080
Ontology IRI http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl	Metrics	
Ontelogy Version IRI e.g. http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl/1.0.	0 Axiom	
	Declaration axioms count	
Annotations		
	Object property count	
	Data property count	
	Individual count	
	DL expressivity	AL
	Class axioms	
	SubClassOf	
	EquivalentClasses	
	DisjointClasses	
	GCL count	
	Hidden GCI Count	
	Object property axioms	
	SubObjectPropertyOf	
	EquivalentObjectProperties	
	InverseObjectProperties	
Ortelans impacts Ortelans Drafinas Conserval alago quianta		
Untology imports Untology Prefixes General class axioms		mer
Imported ontologies.		wat
Direct Imports 🕂		
Indirect Imports		
	To use the reasoner click Reasoner > Start reason	er 🔽 Show Inference

You will also want to save this ontology file to your computer. Under the File menu select Save. Use the next dialog box to specify the format of your ontology file.

	Select ar	n ontology for	mat	
Choose a fo	Choose a format to use when saving the 'ao-workshop' ontology.			
(If you are unsure as to what format to choose, we recommend that you use the standard RDF/XML format, or a widely supported format such as Turtle)		ecommend ely supported		
RDF/XML	Syntax			\$
		Ca	ncel	ОК

Protégé allows you to save your ontology in a variety of OWL formats, including the OBO 1.2 flat file format. We recommend that you save your ontology in RDF/XML, as this is the most stable format to work with in Protégé. You can always choose to export your file in one of the other formats later. Click OK to continue. Name your ontology, perhaps tutorial.owl. Choose a location on your computer to save your ontology.

2.1 The Protégé UI

The Protégé interface follows a basic paradigm of Tabs and Panels. By default, Protégé launches with the main tabs seen below. The layout of tabs and panels is configurable by the user. The Tab list will have slight differences from version to version, and depending on your configuration. It will also reflect your customizations.

To customize your view, go to the Window tab on the toolbar and select Views. Here you can customize which panels you see in each tab. In the tabs view, you can select which tabs you will see. You will commonly want to see the Entities tab and/or Classes tab and the Object Properties tab.

< 🔿 💿 - (http://pu	url.obolibrary.org/obo/owl-tutorial/ao-workshop.owl)	Q Search for entity	/
Active Ontology × Entities × In	ndividuals by class × DL Query ×		
Ontology header:			
Ontology IRI http://p	purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl		
Ontology Version IRI e.g. http	p://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl/1.0.0		
Annotations			
Ontology imports Ontology Prefi	xes General class axioms		
Ontology imports Ontology Prefi Imported ontologies:	ixes General class axioms		
Ontology imports Ontology Prefi Imported ontologies: Direct Imports +	ixes General class axioms		
Ontology imports Ontology Prefi Imported ontologies: Direct Imports + Indirect Imports	ixes General class axioms		
Ontology imports Ontology Prefi Imported ontologies: Direct Imports + Indirect Imports	xes General class axioms		
Ontology imports Ontology Prefi Imported ontologies: Direct Imports + Indirect Imports	Ixes General class axioms		
Ontology imports Ontology Prefi Imported ontologies: Direct Imports + Indirect Imports	txes General class axioms		
Ontology imports Ontology Prefi Imported ontologies: Direct Imports	fixes General class axioms		

The first tab you see is the Active Ontology tab. Here you will find some basic meta-data about the ontology you are viewing. At the very top you see the IRI and file name of the active ontology you are viewing. Protégé allows you to work with multiple ontologies at once, so *this top bar is very important as it lets you know which ontology you are viewing and editing*.

Note: if you open a new ontology while viewing your current ontology, Protégé will ask you if you'd like to open it in a new window. If you select no, it will open in the current window and you can then switch back and forth to it from the Active Ontology tab.

If you say yes, it will open in a new window. If you use a Mac, you can toggle back and forth between each window by using the hot keys Command \sim .

	<u>ش</u>
$\begin{array}{c} 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \\$	} \ } \ } \ return
shift Z X C V B N A > ? shift C V B N M <	shift A v
Active Ontology IRI http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl Ontology IRI http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl	
Ontology Version IRI e.g. http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl/1.0.0 Annotations +	
Ontology Imports Ontology Prefixes General class axioms	
Imported ontologies: Direct Imports Indirect Imports	0880

The panel in the center is the ontology annotations panel. You can use this panel to add basic meta-data to your ontology, such as the creation date, the authors and a short description. Using the annotation panel, create a simple comment on the ontology describing what it is about. First select the + button that is labelled Annotations.

ao- (http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl)	
Active Ontology × Entities × Individuals by class × DL Query ×	
Ontology header:	
Ontology IRI http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl	
Ontology Version IRI e.g. http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl/1.0.0	
Annotations 🕑	
Ontology imports Ontology Prefixes General class axioms	
Imported ontologies:	
Direct Imports	
Indirect Imports	

A dialog will open, select the comment annotation on the left, and type your text into the text box on the right-hand side. Click OK; to add the annotation.

	Create Annotation	
	Literal Entity IRI III Editor Property values	
 ow!:backwardCompatibleWith ow!:deprecated ow!:versionInfo ow!:versionInfo rdfs:isDefinedBy rdfs:label rdfs:seeAlso 	Value This is a tutorial for BDK14.]	
	Type Cancel OK	

The comment should appear in the ontology annotations where you have the option to either edit or delete it. Throughout the application, the grey-circle icons have related functionality: a + is used to add, x to delete, and o to edit.

ao-workshop (http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl) : [[/Users/vasilevs/Desktop/Untitled.owl]	
< > ao-workshop (http://purl.obolibrary.org/obo/owl-tutorial/ao-works	hop.owl)	Search
Active Ontology x Entities x Individuals by class x DL Query x		
Ontology header: DHB	Ontology metrics:	
Ontology IRI http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl	Metrics	
Ontology Version IRI e.g. http://purl.obolibrary.org/obo/owl-tutorial/ao-workshop.owl/1.0.0	Axiom	0
	Logical axiom count	0
Annotations 🗭	Declaration axioms count	0
rdfs:comment	Class count	0
This is a comment for BDK14.	Object property count	0
	Data property count	0
	Individual count	0
1	DL expressivity	AL
	Class axioms	
	SubClassOf	
	EquivalentClasses	
	DisjointClasses	
	GCI count	
	Hidden GCI Count	
	Object property axioms	
	SubObjectPropertyOf	
	EquivalentObjectProperties	
	InverseObjectProperties	

The active ontology tab contains additional information about the ontology that we will explore later. These include a panel for managing ontology imports.

CHAPTER 3

The entities tab

You will see along the top of the screen various tabs. Each tab provides a different perspective on the ontology. An entity is any class, property (object, data, or annotation), or individual. For example, the classes tab allows us to view and edit the classes in the ontology, and similarly the object properties tab focuses on the object properties in the ontology. The *primary tab* where you will spend most of your time is the Entities tab.

(ao-	QSearch for entity	
Active Ontology × Entities × Individuals by class × DL Query ×		

Select the Entities tab and then select the Thing class. Thing is the root class for all OWL ontologies and it cannot be deleted in Protégé.

The Entities tab is split into two halves. The left-hand side provides a suite of panels for selecting various entities in your ontology. When a particular entity is selected the panels on the right-hand side display information about that entity. The entities panel is context specific, so if you have a class selected (like Thing) then the panels on the right are aimed at editing classes. The panels on the right are customizable. Based on prior use you may see new panes or alternate arrangements.

💠 🔿 ao-	Å.	Q Search for entity
Active Ontology × Entities × Individuals by c	lass × DL Query ×	
Class hierarchy Class hierarchy (inferred)	Class Annotations Class Usag	ge
Class hierarchy: Thing	Annotations: Thing	
	Annotations +	
Thing		
	Description: Thing	
	Equivalent To 🕂	
	Subclass of A	
	Subclass Of	
	General class axioms 🕂	
	SubClass Of (Anonymous Ancestor)	
	Instances +	
	Target for Key 🛨	
To use t	he reasoner click Reasoner 🚽	Start reasoner 🗹 Show Inferences

3.1 Creating your first class

By far the most common panel for working with your ontology is the Class hierarchy panel.



There are three button at the top of the class hierarchy view. These allow you to add a **subclass** (**L-shaped icon**), **add a sibling class** (**c-shaped icon**), or **delete a selected class** (**x'd circle**). Click the add subclass button to add a child class to OWL thing.

ao-workshop (http://purl.obolibrary.org/obo/owl-tuto
< > ao-workshop
Active Ontology \times Entities \times Individuals by class \times DL Quer
Class hierarchy: owl:Thing
ti tit X Asserted ♀
owl:Thing
add subclass

A dialog will popup. For now, simply name this class cellular_component. Click "OK" to add the class.

00	Create a new OWLClass
Name: <mark>cellular</mark> _	component
IRI: purl.obo	ibrary.org/obo/owl-tutorial/ao-workshop.owl#cellular_component
	New entity options
	Cancel

The class should have been created as follows. By default, Protégé will use the ontology IRI, followed by a #, followed by your specified name (replacing spaces with underscores) as the unique IRI for this entity. If you hover over this class with your mouse you will see the full IRI for this class. Important: If you have previously configured an IRI generation scheme you may see your IRI being generated in an alternate format (see below).

• • •	ao-workshop (http://purl.obolibr	ary.org/obo/owl-tut	torial/ao-workshop.owl) : [/U
<	> o ao-workshop (http:/	/purl.obolibrary.or	g/obo/owl-tutorial/ao-wor
Active Ontology \times	Entities × Object Properties ×	ndividuals by class	× DL Query ×
Class hierarchy: c	ellular_component		Annotations: cellular_comp
🐮 🕵 🕅		Asserted ᅌ	Annotations 🕂
 owl:Thing cellular_ 	_component		
(http://purl.obolibrary.org/obo/owl-tu	torial/ao-workshop.o	wl#cellular_component

3.2 Renaming an entity

We can change the IRI for a concept using the rename function in the Refactoring menu. Note that this function can also be accessed with a command U keystroke. Rename the **cellular_component** class to use its proper IRI from the Gene Ontology http://purl.obolibrary.org/obo/GO_0005575

Protege File Edit View Reasoner Tools	Refactor Window Help	🍓 🖯 😻 b 🌲 🖷 🖸 🛜 🕙
● ● ● ao- (http://purl.obolibrar	Rename entity	HU Change the IRI of the selected entity
♦ ao- (http://purl.obolibrary.org/obo/o	Rename multiple entities Change ontology IRI	r entity
Active Ontology × Entities × Individuals by class × DL Q	Convert entity IRIs to labels	
Class hierarchy Class hierarchy (inferred)	Convert property assertion on class/individual puns to annotations	s
Class hierarchy: cellular_component	Coerce data property values into property range	
 Thing cellular_component 	Split subclass axioms Amalgamate subclass axioms Split disjoint classes into pairwise disjoints Amalgamate disjoint classes into larger disjoint sets	
	Convert qualified min cardinality 1 to someValuesFrom	
	Copy/move/delete axioms	
	Merge ontologies	

Make sure to check the "Show full IRI" box so you can edit the full IRI.

00	Change entity UR	1
http://purl.ob	olibrary.org/obo/owl-tutorial/ao-worksh	op.owl#cellular_component
🗹 Change all	entities with this URI	Show full IRI
		Cancel OK

And then paste or type in the correct GO URI (http://purl.obolibrary.org/obo/GO_0005575).

00	Change entity U	RI
1		
http://purl.ob	blibrary.org/obo/GO_0005575	
🗹 Change all	entities with this URI	🗹 Show full IRI
		Cancel OK

Now the correct GO URI appears in the ontology; in the Class hierarchy panel, the class will appear as "GO_0005575" (to those who have used Protégé before you might see a different label). Luckily you don't have to rename every entity you create when building your own ontology; Protégé provides a "New Entities" preferences panel where you can specify how new IRI should be created, described in the next section.

3.3 New entities

Terms in the ontologies we use have separate names and IDs. The names are annotation values (labels) and the IDs are represented using IRIs. The OBO foundry has a policy on IRI (or ID) generation (http://www.obofoundry.org/principles/fp-003-uris.html). You can set an ID strategy using the "New Entities" tab under the Protégé preferences – on the top tool bar, click the "Protégé dropdown, then click Preferences.

Ű.	Protégé	File	Edit	View	Reasoner	Tools	Refactor	Window	Μ
•	About Pr	out Protégé ourl.obolibrary.org/obo/owl-tutorial/ao-wor) :
	Preferen	ces	ж,	lo-wo	rkshop				
Activ	Services		►	ndividu	als by class	× DL Qu	iery ×		
Clas L: V-(Hide Pro Hide Oth Show All	tégé iers	НЖ НЖ∵		Asser	ted ᅌ	Annotation Annotation rdfs:la	ons: GO_00 s + abel	05
	Quit Prot	égé	₩Q			dc:cre	ar_compone ator //orcid.org/0	nt <u>00(</u>	
							dc:dat 2018	te [type: >	(sd: 04:
							Descripti	on: CO_000)55
							Equivalent		,,,,,

Set your new entity preferences as in the following screenshot, then choose the New Entities tab):

			Prefe	rences			
New Ontologies Annotations	OWLViz Debugger	Ontop Re General	JDBC Drivers	Plugins	Reasoner lew Entities	New Entities M	User details Aetadata
Entity IRI							
Start with:	 Active ontolo Specified IRI: 	gy IRI http://purl.	.obolibrary.org/ol	bo/			
Followed by:	○ # ● / ○ :						
End with:	User supplied Auto-generat	l name red ID					
Custom la IRI http: Lang	abel //www.w3.org/200	00/01/rdf-sc	hema#label				
Auto-generated	1 ID						
O Numeric (○ Globally u	iterative) Inique Dig	Prefix: GO_ Suffix: it count: Start: End: V	7 0 0 0 1000: 0 Remember last II) between P	rotégé sessions		
Reset preference	es						

For ontologies other than GO, change the value of the prefix.

Note that all OBO library ontologies should use the "Specified URI" value: http://purl.obolibrary.org/obo

3.4 Adding annotations properties

Using Protégé you can add annotations such as labels, descriptions, cross references (xrefs) to any OWL entity. The panel on the right, named Annotations, is where these annotations are added. Use this panel to add a **cellu-lar_component** label to the class you created previously (notice how when changed the IRI, you also lost your label. This is because the label was previously part of the IRI, and Protégé was rendering the label based on the IRI. We'll fix that in a minute.) Click on the GO labelled class to select it.

Active Ontology × Entities × Object Properties ×	Individuals by class	× DL Query ×	
Class hierarchy: GO_0005575		Annotations: GO_0005575	
	Asserted ᅌ	Annotations 🛨	
• owl:Thing • GO_0005575			

Select the + button to add an annotation to the selected entity. Protégé has a set of built in annotation properties, such as label and comment – add rdfs:label "cellular_component" and click OK. You can also add a comment such as "created during BDK14 tutorial", by clicking the + sign again, choosing "rdfs:comment" on the left hand side bar, and typing your comment in the "Literal" box, then click OK.

•		cellular_component		
		Literal Entity IRI IRI Editor Property values		
	vevi:backwardCompatibleWith owl:backwardCompatibleWith owl:priorVersion owl:priorVersion owl:versionInfo rdfs:comment rdfs:label rdfs:label rdfs:seeAlso	Value cellular_component		
		Type 🔷 Lang 💌		
			Cancel	ОК

	cellular_component	
	Literal Entity IRI IRI Editor Property values	
 owl:backwardCompatibleWith owl:deprecated owl:priorVersion owl:versionInfo rdfs:isDefinedBy rdfs:label rdfs:seeAlso 	Value reated during BDK14 tutorial	
	Type Cancel OK	

Note that often you will start from an existing OWL file, then your ontology will include a pre-declared set of annotation properties such as 'has exact synonym' and 'definition'. *You may never need to create your own annotation properties.*

3.5 Setting label rendering

You can change how Protégé renders entities. It is common to want to view entities by their **label**, rather than **identifiers.** In fact, you can tell Protégé to render on any annotation property you choose. Experiment with the different options in the menu, and to conclude, set the rendering to use the class label (rdfs:label).

In the View menu choose "Render by label":

🗯 Pro	tégé	File	Edit	View	Reasoner	Tools	Refactor	Window	Mastro	Ontop	Help	42	_E WD	0
• • •	ao-wo	rkshop	o (http:	Rend	der by entity	IRI shor	t name (Id)		/User	s/vasilevs	/Desktop	/Untitlec	l.owl]	
<	tology	> × Enti	ities ×	Reno ✓ Reno Reno	der by prefix der by label (der by annot	ed name (rdfs:lab) ation pro	el) operty		•			©	Search	
Class hie	rarchy:	cellul	ar_con	Cust	om renderin	g			mpor	nent			081	
1	X			✓ Disp	lay axiom an	notation	is inline	100						
▼ ow	<pre>v liming cellular_compone</pre>			 ✓ Disp ✓ Disp ✓ Disp 	lay thumbna lay deprecat	ils for in ted (obs	nage URLs olete) entiti	es						
				✓ Show Show Show	w the import w all loaded w only the ac	s closure ontologi ctive ont	e of the act es ology	ive ontology	/ tutor	rial			@×0	
				Expa	and all									
							Descript	ion: cellular	_compon	ent			081	
							Equivalent	то 🕂						
							SubClass C	of 🕂						L
						General cl	ass axioms 🕇						l	
							SubClass C)f (Anonymous	Ancestor)					
Annotatio	n prope	rty hier	rarchy		Dataty	pes	Instances	Ð						
Data prop	perty hie	erarchy		Indiv	viduals by typ	e		-						
Object pr	roperty I	hierarcl	hy				😑 cellul	ar_compon	ent — GO	:0005575	— http:/	//purl.ol	bolibrary.o	rg/
Object p	roporty	hiorar	chy			meias	2	To use th	e reasoner cli	ck Reasoner :	Start reaso	ner 🔽	Show Inference	es

The **cellular_component** class will now render in the hierarchy view using the value of the label annotation property. Note that the ability to flip between different renderings can often be very useful in old versions of Protégé, as the Protégé search box in the upper right searches on whatever is rendered. If you are using an older version of Protégé, searching for a term by ID for example, it can be useful to render by ID and then flip back to render by label. In the 5.1 version, the search box will search for either labels or ID.

3.6 Creating the class hierarchy

We will now create a simple class hierarchy. In Protégé, 'class hierarchy' typically refers to a sub/superclass hierarchy (also known as an ' is_a ' hierarchy in OBO format). We will return to relations such as '*part_of*' later on in this tutorial. For now, we will take advantage of the fact that the GO cell component ontology allows us to bypass this for now by means of classes such as 'cell part' and 'nuclear part'.

Classes may be quickly added to an ontology with the **add subclass (vertical arrow below)** and **add sibling class** (**horizontal arrow below**) icons in the class hierarchy view.



Use the **add subclass (vertical arrow below)** and add **sibling class (horizontal arrow below**) buttons to create a hierarchy that looks like the following (your window will look slightly different than the view below which is from an earlier version of Protégé). Note that you can click and drag classes in the hierarchy to re-arrange them. When planning your own ontology take a good amount of time to standardize your label format, check other ontologies, and be consistent.



Don't bother to add textual definitions, synonyms, etc. at this stage, as you won't be using this ontology in latter exercises. *Note: the order of the classes in your Class hierarchy may not be the same as you see in the screenshot (e.g. 'cell part' may appear above cell). Don't worry about this. Just make sure that the subclass relationships are correct.*

CHAPTER 4

EXERCISE: Basic Subclass Hierarchy

Go to the directory basic-subclass in the BD2K14_exercises folder and open /basic-subclass/chromosome-parts.owl.

This example illustrates adding classes and class annotations into an existing Subclass hierarchy.

Note: This example does not make use of reasoning / automated classification or class expressions.

Constructs illustrated:

- · Adding subclasses
- · Adding annotations to classes
- Adding DatabaseCrossReferences (xref) classes

Instructions:

Add Subclass

- 1. Open chromosome-parts.owl and navigate to the Entities tab.
- 2. Add the class "replication fork" to the ontology as a subclass of 'chromosomal part'. (Don't worry about the ID.) *Note: Most ontologies use lowercase labels, except for proper names.*

a. To find the class 'chromosomal part', you can navigate through the hierarchy, or use the search function, see screenshot below. Make sure you select "Show all results". In the search results, double click on the highlighted class and it will open it up in the Class hierarchy pane.

chromosome-parts (ht)	ttp://purl.obolibrary.org/obo/test/ch	nromosome-parts.owl) : [/Users/vasilevs/git	/BDK14-Ontologies-101/BDK14 exercises/basi								
< > ¢ c	hromosome-parts (http://purl.c	bolibrary.org/obo/test/chromosome-pa	arts.owl) 🗘 Search								
Active Ontology × Entities × C	Dbject Protection	Search									
Class hierarchy: owl:Thing	chromosomal part										
🐮 🕵 · 🐹	Case sensitive	Whole words 🛛 Ignore white space	Regular expression Show all results								
▼	Search in IRIs	Search in IRIs Search in annotation values 🛛 Search in logical axioms									
cellular_component	Found in	Entity	Match								
	Display na	ame 😑 'chromosomal part'	chromosomal part								
	rdfs:1	abel 🤤 'chromosomal part'	rdfs:label [type: xsd:string]								
	EquivalentCla	sses 🏮 'chromosomal part'	' <mark>chromosomal part</mark> ' Equivalent'l								
	SubCla	ssof © 'chromosomal part' © 'chromasomal part' © chromatin © chromocenter © 'chromosome, telomeric region' © 'chromosome, centromeric regior © 'DNA replication termination regi	'chromosomal part' SubClassO 'chromosomal part' SubClassO chromatin SubClassOf 'chrom chromocenter SubClassOf 'chro 'chromosome, centromeric region 'chromosome, centromeric regio on' 'DNA replication termination re								
Individuals by type Annotatic	Copy selected entit	es SubClass Of (Anonymous Ancestor)									
Object property hierarchy	Data property hierarchy	To use the re	near a fick Bearanner - Clast menoner - Cl. Oh Information								

Moving around classes:

- 1. Add the term "intracellular non-membrane-bounded organelle" as a subclass of "Thing"
- 2. Move it, by dragging and dropping, to place it as a subclass of 'non-membrane-bounded organelle' (if you do not see the class 'non-membrane-bounded organelle';, search for it by clicking the 'search' radio button in the upper right.)

Your hierarchy should look something like this:



Annotations

The goal is to recreate the existing information from GO on the "replication fork" class.

- 1. Click on the 'replication fork' class you just created.
- 2. In the Annotations pane on the right, use the (+) next to 'Annotations' to add an annotation.

Annotations: 'replication fork'	
Annotations 🛨	
rdfs:label	
replication fork	
replication fork	

You will add the annotation values listed are below, detailed instructions follow. Make sure you click on the correct annotation on the left for each annotation.

id: GO_0005657

rdfs: label: replication fork

definition: The Y-shaped region of a replicating DNA molecule, resulting from the separation of the DNA strands and in which the synthesis of new strands takes place. Also includes associated protein complexes.

database_cross_reference: ISBN:0198547684

has_related_synonym: replication focus

xref: Wikipedia:Replication_fork

Detailed instructions:

Add the following (using the values above)

- 1. A text definition for the class
- Click on the "replication fork" class, then click (+) by Annotations
- By default, the window should be on the "Literal" tab
- Click (select) "definition" on the left



- Enter the definition in the window: The Y-shaped region of a replicating DNA molecule, resulting from the separation of the DNA strands and in which the synthesis of new strands takes place. Also includes associated protein complexes. *Note: Make sure there are not any extra spaces at the end of the sentence.*
- Click OK. The annotation should appear in the Annotations window.

Class Annotations Class Usage	
Annotations: 'replication fork'	
Annotations 🕂	
rdfs:label	30
replication fork	
definition	30
The Y-shaped region of a replicating DNA molecule, resulting from the separation of the DNA strands and in which the synthesis of new stran- takes place. Also includes associated protein complexes.	ds

2. dbxrefs to the text definition

• Click the (@) icon beside the definition annotation

Annotations 🕂	
rdfs:label	@×0
replication fork	
definition	
The Y-shaped region of a replicating DNA mo the DNA strands and in which the synthesis includes associated protein complexes.	olecule, resulting from the separation of of new strands takes place. Also
has_related_synonym	@×0
replication focus	
database_cross_reference	@ X 0
Wikipedia:Replication fork	

- A new box will open, and click the (+) beside Annotations
- Select "database_cross_reference" in the left pane
- Enter a value. This is often a PubMed ID (in the format PMID:xxxxx) or your ORCID ID or initials. For this exercise, enter the initials: mah.
- Repeat for the other cross reference: ISBN:0198547684
- Your annotations should look like the screenshot below.

Class Annotations Class	Usage
Annotations: 'replication	fork' 🛛 🗆 🖿 🖿 🗠
Annotations 🕂	
rdfs:label	@ × •
replication fork	
definition	<u> </u>
The Y-shaped region of separation of the DNA stakes place. Also includ	a replicating DNA molecule, resulting from the strands and in which the synthesis of new strands les associated protein complexes.
database_cross_	reference
GOC:mah	
database_cross_	reference
<u>ISBN:01985476</u>	<u>584</u>

A related synonym

- Add an annotation to "replication fork" with the (+)
- Choose "has_related_synonym" and enter the value: replication focus
- Add an xref to the class itself:
- click the (+) beside annotations
- Choose database_cross_reference
- Add xref (Wikipedia:Replication_fork)
- Click OK

Your annotation pane should look something like this:

Class Annotations Class Usage	
Annotations: 'replication fork'	
Annotations 🕂	
rdfs:label	
replication fork	
definition (@	×O
The Y-shaped region of a replicating DNA molecule, resulting from the separation of the DNA strands and in which the synthesis of new strant takes place. Also includes associated protein complexes.	nds
database_cross_reference	
ISBN:0198547684	
mah	
has_related_synonym @	×O
replication focus	
database_cross_reference	×o
Wikipedia:Replication_fork	

Synonym properties:

- 1. Add the subclass "site of double-strand break" to the ontology under "chromosomal part"
- 2. Add a synonym with a dbxref annotation. E.g. synonym: "site of DSB" has_exact_synonym [PMID:21035408]

Annotations: 'site of double-strand break'	
Annotations rdfs:label site of double-strand break	@×0
has_exact_synonym site of DSB database_cross_reference <u>PMID:21035408</u>	<u> </u>

Note that there are different synonym annotations:

- has_exact_synonym: a synonym that has the exact same meaning as the class name
- has_narrow_synonym: a more specific synonym
- has_related_synonym: a related term

4.1 The Class description view

Open the ontology chromosome-parts-interim.owl, found in the "basic-subclass" exercise folder. Save it to your local computer using Save-as.

We have seen how to add sub/superclasses and annotate the class hierarchy. Another way to do the same thing is via the Class description view (circled in the figure below). When an OWL class is selected in the entities view, the right-hand side of the tab shows the class description panel. If we select the **cell** class, we see in the class description view

that this class is a "SubClass Of" (= has a *SuperClass*) the **cellular_component** class. Using the (+) button beside "SubClass Of" we could add another superclass to the cell class.

< > \$ chro	Search	
Active Ontology × Entities × Indiv	riduals by class × DL Query ×	
Class hierarchy:	DED® Annotations:	
🐮 🕼 · 💢	Asserted 📀 Annotations 🕀	
owl:Thing		
	Description:	0862
	Equivalent To 🕀	
	Subclass of	
	General class axioms 🕂	
	SubClass Of (Anonymous Ancestor	or)
	Instances 🕀	
Individuals by type Annotation pro	perty hierarchy Datatype	
Object property nierarchy Di	ata property nierarchy	
Object property hierarchy:		

Select the **intracellular organelle part** class in your ontology. Notice it is a SubClass of **organelle part**. Using the SubClass Of (+) button, add the **organelle** class as a super class. There are various ways to assert a superclass. The simplest it to just type in the class expression editor. *Hint: Pressing Tab (or CTRL + SPACE on a Mac) allows you to autocomplete on a term.*



You can also use the class hierarchy tab here to search, browse and select the appropriate class.

	'intracellular organelle part'	
Class expression editor	Class hierarchy	
₽ ; ₽ , X		Asserted ᅌ
<pre>vwl:Thing v @ cellular_con v @ cell v @ cell v @ 'cell part' v @ organelle v @ 'organelle</pre>	nponent e part'	
	Cano	cel OK

The **intracellular organelle part** class will now have two parents asserted in the class hierarchy. You will also be able to see both parents in the class description view.

Class hierarchy: 'intracellular organelle part' DEBC Class hierarchy: 'intracellular organelle part' DEBC Asserted	Annotations: 'intracellular organelle part'	
😫 🕵 🕺 Asserted 🗘	Annotations 🛨	
	Annotations	
• owl:Thing • cellular_component	rdfs:label [type: xsd:string] intracellular organelle part	@×0
<pre>> Cell part' > organelle > 'intracellular organelle part'</pre>	id [type: xsd:string] GO:0044446	@×0
 intracellular organelle' organelle part organelle part 	has_obo_namespace [type: xsd:string] cellular_component	@ × O
intracellular organelle part'	definition [type: xsd:string]	
	A constituent part of an intracellular organelle, an or distinctive morphology and function, occurring within constituent parts of the nucleus, mitochondria, plass vesicles, ribosomes and the cvtoskeleton but exclude	rganized structure of n the cell. Includes tids, vacuoles, es the plasma
	Description: 'intracellular organelle part'	
	Equivalent To 🕂	
	organelle organelle	?@XO ?@XO

Save your interim ontology to your computer. Note: You will use this same file in the next section: Disjointness.

CHAPTER 5

Protégé plugins

Protégé is built on a plugin architecture. There is an active community of developers writing plugin extensions to Protégé. There is a plugin library in Protégé that allows you to pick and install plugins. You may also find plugins elsewhere on the web that must be installed manually. (*Note: Plugins are distributed as java archives (jars). To manually install a plugin you simply need to place the jar in the plugins folder inside the Protégé home/root directory.*)

You can find the plugin library in the Protégé preferences. Go to your preferences and then the Plugins tab.

Auto update	Automatically che	ck for plugin	updates at s	tart up				
Plugin registry	hubusercontent.com/protegeproject/autoupdate/master/update-info/5.0.0/plugins.repository							ory
	This is the location that I	Protégé will use t	o check which pl	ugins are availabl	le			
	Reset to de	fault registry	location					
Installed plugins	Name/ID			Version	Quali	fier		
, ,	Browser View (OWLDoc)			3.0.3				
	Cellfie Protege 5.0+ Plu	ıgin		2.1.0				
	Change Tracker			2.0.2				
	CSV Export Plugin			1.0.0				
	DL Query			4.0.1				
	ELK Reasoner Protege F	Plug-in		0.4.3				
	Existential Query			2.0.0				
	Explanation Workbench			3.0.0				
	Factplusplus Plug-in			1.6.5				
	HermiT			1.3.8	413			
	Individual Properties Co	ontextual Asser	tions	1.0.1				
Reset preferences								

In older versions of Protégé, you may want to install the **Annotation Search Views** and the **Outline/Existential Tree** plugins. Instructions for older versions are below. Note: this is not necessary for Protégé 5.1. If you are using Protégé 5.1, skip to the next section on Disjointness.

5.1 Annotation search plugin (for older versions of Protege)

Most plugins are either tabs, panels or menu items. The annotations search plugin provides a new panel that can be used to search through OWL annotations (such as labels and definitions). Tabs and panels can be found in the Window menu. Under Window -> Views -> Misc views -> Search Annotations. Once selected your pointer will become a circular icon. You can choose to drop this panel over any existing panel in Protégé by clicking. We recommend that you drop this panel to the right of the class hierarchy view, on top of the existing annotation view panel.
d Window Help		
Views > Tabs > Create new tab > Delete custom tabs Import tab Import tab * Store current tab * Store current tab * Reset selected tab to default state Increase font size Percease font size * Timestamp log / console Look & Feel Refresh User Interface	Annotation property views Class views Data property views Datatype views Individual views Misc views Object property views SKOSEd views Iabel 'part_of'	al/tutorial/basic-subclass/chromosome-parts-interim.owl] DL Query OPPL OPPL Patterns Axiom annotations Manchester syntax entity rendering Query Search Annotations Selected entity
	Characteristics: part dielse Desc Functional Inverse functional Transitive Symmetric Reflexive	rription: part_of UEECO

\varTheta 🖯 🖯 chr (http://purl.obolibrary.org/obo/test/chromos	some-parts-interm.owl):[/Users/vasilevs/Desktop/chromosome-parts-interm.owl]
chr (http://purl.obolibrary.org/obo/test/chr	romosome-parts-interm.owl)	
Active Ontology × Entities × Object Properties × Individuals by	class × DL Query ×	
Class hierarchy Class hierarchy (inferred) Class hierarchy: Thing Cl	Class Annotations Class Usage Annotations: Thing DEI	
Thing		
	Search Annotations:	+
	results (0)	
	Description: Thing III	
	SubClass Of T	
	To use the reasoner click Reasoner $ imes$ Start reasoner $\overline{\!$	

You can use the annotation search panel to search through all annotations, or restrict it to individual annotations, such as the label. The annotation view also supports *regular expression* queries.

Disjointness

In the chromosome-parts-interim.owl file (that you have now saved locally on your computer), at the top of our class hierarchy we have cell, cell part, chromosomal part, intracellular part, organelle and organelle part. By default, OWL assumes that these classes can overlap, i.e. there are individuals who can be instances of more than one of these classes. We want to create a restriction on our ontology that states these classes are different and that no individual can be a member of more than one of these classes. We can say this in OWL by creating a *disjoint classes* axiom.

If you do not already have it open, load your previous ontology that was derived from the 'interim file'. Note: you can open a recent file by going to File-> Open Recent

We want to assert that **organelle** and **organelle part** are disjoint. To do this first select the **organelle** class. In the class 'Description' view, scroll down and select the (+) button next to Disjoint With. You are presented with the now familiar window allowing you to select, or type, to choose a class. In the hierarchy panel, you can use CTRL to select multiple classes. Select 'organelle part' as disjoint with organelle.



Note that the directionality is irrelevant. Prove this to yourself by deleting the disjoint axiom, and adding it back from **organelle part**.

6.1 Reasoning and inconsistency checking

We have introduced a deliberate mistake into the ontology. We previously asserted that **intracellular organelle part** is a subclass of both **organelle part** and **organelle**. We have now added an axiom stating that **organelle** and **organelle part** are disjoint. We can use the reasoner to check the consistency of our ontology. The reasoner should detect our contradiction.

Protégé comes with several reasoners, and more can be installed via the plugins mechanism (see plugins chapter). Select a reasoner from the Reasoner menu (Elk, HermiT, Pellet, or Fact++ will work). Once a reasoner is highlighted, select 'Start reasoner' from the menu. *Note: you may get several pop-boxes/warnings, ignore those.*

The intracellular organelle part class will have changed to red indicating that the class is now unsatisfiable.



You can also see unsatisfiable classes by switching to the inferred view.



Here you will a special class called **Nothing**. When we previously said that all OWL classes are subclasses of OWL Thing. OWL **Nothing** is a leaf class or bottom class of your ontology. Any classes that are deemed unsatisfiable by the reasoner are shown as subclasses or equivalent to OWL Nothing. The inferred view will show you all subclasses of Nothing.



Once the ontology is classified, inferred statements or axioms are shown in the various panels with a light-yellow shading. The class description for **intracellular organelle part** should look something like the screen shot below. You will see that the class has been asserted equivalent to the **Nothing** class. Inside this statement, a small question mark icon appears, clicking this will get an explanation from the reasoner for this inconsistency.

Description: 'intracellular organelle part'	
Equivalent To 🕂	
owl:Nothing	?@×0
SubClass Of + organelle part'	?@×0
🛑 organelle	?@×0
General class axioms	

Select the (?) icon to get an explanation for this inconsistency. The explanation shows the axioms involved. We see the disjoint class axiom alongside the two subclass axioms are causing the inconsistency. We can simply repair this ontology by removing the **intracellular organelle part** subClassOf **organelle** axiom.

 Show regular jus Show laconic jus 	tifications stifications	 All justifications Limit justifications to 2 	
Explanation 1	Display laconic	explanation	
Explanation for mitra			0
'intracellular	organelle part	Subclassor organelle	
'intracellular 'intracellular	organelle part organelle part	Subclassof organelle part'	0
'intracellular 'intracellular organelle Di g	organelle part organelle part sjointWith 'or	subclassof organelle part' rganelle part'	0
'intracellular 'intracellular organelle Dis	organelle part organelle part sjointWith 'or	subclassor organelle part' rganelle part'	0
'intracellular 'intracellular organelle Dig	organelle part organelle part sjointWith 'or	subclassor organelle part' rganelle part'	0

Remove the Disjoint with axiom (click the (x) beside **organelle** in the Description pane for **intracellular organelle part**), and resynchronise the reasoner from the reasoner menu. Save your ontology, you'll return to it after this following exercise.

Object properties

Re-open your previously saved ontology, which you may have named something like "chromosome-parts-interim.owl".

We will now create an object property and use this to add some restrictions onto classes. In OWL, object properties are used to assert relationships between individuals (or instances). Object properties in OWL can have characteristics such as being *transitive* or *symmetric*. We can assert additional information about properties such their *domain* and *range*, along with defining *inverse* properties.

7.1 Create an object property

We will use the object property view in the Object Properties tab to create a part_of property. In OWL, all properties are a sub-property of topObjectProperty.

• • chromosome-parts (http://purl.obolibrary.org/obo/test/chromosome-parts.owl) : [/Users/vasilevs/Dropbox/ODG_Shared_Folder/				
< > > chromosome part	(http://purl.obolibrary.org/obo/test/chromosome-parts.owl) 🗘	Search		
Active Ontology × Entities × Object Properties ×	Individuals by class × DL Query ×			
Object property hierarchy:	Annotations Usage			
Ti Ci 🔀 Asserted 🥃	Annotations:			
owl:topObjectProperty	Annotations 🕂			
	Charact @888 Description:			
	Function Equivalent To			
	Inverse f			
	Transitiv			
	Symmetr			
	Asymme Domains (intersection) +			
	Reflexive Ranges (intersection)			
	To use the reasoner click Reasoner > Start reasoner	Show Inferences		

Select owl:topObjectProperty, then the "Add sub property button" circled below and name the property part_of.

chromosome-parts (http://purl.obolibrary.c	rg/obo/test/chrom	osome-parts.owl) : [/Users/vasilevs/Dropbox/ODG_	Shared_Folder/
< > < chromosome-parts	(http://purl.oboli	brary.org/obo/test/chromosome-parts.owl) 🗘	Search
Active Ontology × Entities × Object Properties ×	Individuals by clas	s × DL Query ×	
Object property hierarchy: owl:topObjectF III III	Annotations Usa	ge	
T Asserted 🗘	Annotations: owl	:topObjectProperty	
	Annotations +		
	Create a new OWL	ObjectProperty	
Name: Short name or full	IRI or Prefix-Nam	e	
IRI: IRI (auto-generate	:)		
		New entity options	
		Cancel OK	
	Transitiv	SubProperty Of	
	Symmetr	Inverse Of 🛨	
	Asymme	Domains (intersection) 🕂	
	Reflexive	Ranges (intersection)	
		To use the reasoner click Reasoner > Start reasoner	Show Inferences

We can use the property description view shown below to make assertions about this property. We want to state that the **part_of** property has the characteristic of being transitive. If a property is transitive, and the property relates individual a to individual b, and also individual b to individual c, then we can infer that individual a is related to individual c via property P. A good example of a transitive property is the geneological 'ancestor of' relationship. We can make a property transitive in Protégé by simply selecting the transitive check box.

💿 😑 Chromosome-parts (http://purl.obolibrary.	org/obo/test/chromosome-pa	rts.owl) : [/Users/vasilevs/Dropbox/ODG_S	hared_Folder/
< > < chromosome-parts	(http://purl.obolibrary.org/	/obo/test/chromosome-parts.owl) ᅌ	Search
Active Ontology × Entities × Object Properties ×	Individuals by class × DL 0	Query ×	
Object property hierarchy: part_of	Annotations Usage		
T C Asserted 😋	Annotations: part_of		0800
<pre>owl:topObjectProperty part_of</pre>	Annotations () rdfs:label part_of		@ ×0
	Characteristics: part_ WHU	Description: part_of	
	Inverse functional	SubProperty Of 🕀	- 1
	Symmetric	Inverse Of 🛨	- 1
	Asymmetric	Domains (intersection) (+) Ranges (intersection) (+)	
		To use the reasoner click Reasoner > Start reasoner	Show Inferences

OWL class restrictions

Keep the previously opened interim ontology open.

As previously stated, in OWL we use object property to describe binary relationships between two individuals (or instances). We can also use the properties to describe new classes (or sets of individuals) using *restrictions*. A restriction describes a class of individuals based on the relationships that members of the class participate in. In other words, a restriction is a kind of class, in the same way that a named class is a kind of class.

For example, we can use a named class to capture all the individuals that are chromosome parts. But we could also describe the class of chromosome parts as all the instances that are '*part of*' a chromosome.

In OWL, there are three main types of restrictions that can be placed on classes. These are **quantifier restriction**, **cardinality restrictions** and **hasValue** restriction. In this tutorial will initially focus on quantifier restrictions.

Quantifier restrictions are further categorized into two types, the existential and the universal restriction.

- Existential restrictions describe classes of individuals that participate in at least one relationship along a specified property to individuals that are members of a specified class. For example, the class of individuals that have at least one (**some**) 'part of' relationship to members of the 'Chromosome clas. In Protégé, the keyword ' **some'** is used to denote existential restrictions.
- Universal restrictions describe classes of individuals that for a given property only have relationships along this property to individuals that are members of a specified class. For example, we can say a cellular component is capable of many functions using the existential quantifier, however, OWL semantics assume that there could be more. We can use the universal quantifier to add closureto the existential. That is, we can assert that a cellular component is capable of these function, and is only capable of those functions and no other. Another example is that the process of hair growth is found **only** in instances of the class Mammalia. In Protégé the keyword '**only**' is used.

In this tutorial, we will deal exclusively with the existential (some) quantifier.

8.1 Superclass restrictions

Strictly speaking in OWL, you don't make relationships between classes, however, using OWL restrictions we essentially achieve the same thing.

We want to capture the knowledge that the named class '**organelle part**' is part of an **organelle**. In OWL speak, we want to say that every instance of an '**organelle part**' is also an instance of the class of things that have at least one 'part of' relationship to an '**organelle'**. In OWL, we do this by creating an existential restriction on the '**organelle part**' class.

In the Entities tab, select '**organelle part**' in the class hierarchy and look at its current class description in the bottom right box. At the top of this view there are two slots for defining **equivalent classes** and superclasses (as denoted by the SubClass Of list). '**organelle part**' already has one superclass named **cellular_component**.

	Description: 'organelle part'	
	Equivalent To 🕂	
\rightarrow	SubClass Of + cellular_component	7@XO
	General class axioms 🕂	
	SubClass Of (Anonymous Ancestor)	
	Instances 🕂	- 1
	Target for Key 🛨	
	Disjoint With	

We will create a restriction on '**organelle part**' stating '**organelle part**' has a '*part of*' relationship to some '**organelle**'. Select the (+) icon next to the SubClass Of slot. Select the Class expression editor pane. We will define this *anonymous superclass* in Manchester OWL syntax as 'part_of some organelle'.

chromosome-parts (http://purl.ob	olibrary.org/obo/test/	chromosome-parts.	owl) : [/Users/vasilevs/Dropbox/ODG_S	Shared_Folder/O
< > < chromosom	ne-parts (http://pur	l.obolibrary.org/ob	o/test/chromosome-parts.owl) 🗘	Search
Active Ontology × Entities × Object Prope	erties × Individuals	by class × DL Que	ry ×	
Class hierarchy: 'organelle part'		😑 'organelle part'	— GO:0044422 — http://purl.oboli	ibrary.org/obo/G
*	Asserted 🗘	Class Annotations	Class Usage	
▼- ● owl:Thing		Annotations: 'org	anelle part'	
 cellular_component cell cell part' organelle 		Annotations rdfs:label [ty organelle part	pe: xsd:string]	@×0
► <mark>O</mark> 'organelle part'		id [type: xsd GO:0044422	:string]	@×0
		has_obo_name	espace [type: xsd:string]	@×0
		Condidi Combe	'organelle part'	
	Class expre	ssion editor	Data restriction creator	
	Class hierar	chy	Object restriction creator	
Individuals by type Annotation property hie Object property hierarchy Data prop	erarch		Cancel	01
				OK

The class restriction will be shown in the SubClass of slot as follows.

Selected entity Description	
Description: 'organelle part'	
Equivalent To 🕂	
SubClass Of 🛨	
cellular_component	?@×0
part_of some organelle	? @ × O
General class axioms +	
SubClass Of (Anonymous Ancestor)	
Instances +	

Using Protégé create your own part_of restrictions for the 'cell part', 'intracellular part' and 'chromosomal part' classes. *Note: you must use single quotes around text strings that are separated by a space, e.g. 'intracellular organelle part'.*

NOTE: After each edit to the ontology you might want to synchronize the reasoner to make sure you didn't introduce any inconsistencies into your ontology. The edit, reason, edit, reason iteration becomes particularly important as your ontologies grow more complex.

EXERCISE: Basic Restictions

This example illustrates how to use object properties to make existential restrictions.

In OWL, it helps to think in terms of the set of entities represented by each class. To say: 'every finger is part of a hand' we say:

finger SubClassOf part_of some hand

The anonymous class expression 'part_of some hand'; represents the set of all instances that have a part_of relationship to a hand. Every member of the set of all fingers is a member of the set of all things that are part of a hand.

Instructions:

- 1. Open er-sec-complex.owl from the basic-restiction folder
- 2. Navigate to the class 'protein complex' using the search box
- 3. Add a class 'endoplasmic reticulum Sec complex' as a subclass of 'protein complex'
- 4. Say that every 'endoplasmic reticulum Sec complex' is part of a 'rough endoplasmic reticulum membrane'

💿 😑 🔵 go (http://purl.obolibrary.org/obo/go.owl) : [/Users/vasilevs	/git/BDK14-Ontologies-101/BDK14_exercises/basic-restriction/e	r-sec-complex
< > of the second secon	(go.owl)	Search
Active Ontology × Entities × Object Properties × Individuals by c	lass × DL Query ×	
Class hierarchy: 'endoplasmic reticulum Sec complex' 🛛 🕮 🖽 🖼	😑 'endoplasmic reticulum Sec complex' — GO:6000000 —	http://purl.obol
🐮 🕵 🔀 Asserted 📀	Class Annotations Class Usage	
<pre> • owl:Thing • cellular_component • cell • cell part' • component • cell part' • complasmic part' • o 'endoplasmic reticulum part' • 'macromolecular complex' • o 'protein complex' • o 'endoplasmic reticulum Sec complex' • o 'translocon complex' • o 'rganelle • o 'organelle part' </pre>	Annotations: 'endoplasmic reticulum Sec complex' Annotations rdfs:label [language: en] endoplasmic reticulum Sec complex	
	Description: 'endoplasmic reticulum Sec complex'	
	Equivalent To 🕂 SubClass Of + o 'protein complex' o part_of some 'rough endoplasmic reticulum membrane' General class axioms 🕂	7@20 7@20

1. Say that a 'endoplasmic reticulum Sec complex' has a 'Sec61 translocon complex' as part

Description: 'endoplasmic reticulum Sec complex'	
Equivalent To 🕂	
SubClass Of 🛨	
'protein complex'	?@×0
has_part some 'Sec61 translocon complex'	?@×0
part_of some 'rough endoplasmic reticulum membrane'	?@×0

Navigating over the resulting ontology:

- 1. Synchronize the reasoner
- 2. Navigate to 'rough endoplasmic reticulum membrane'.
- 3. Find the parts of the rough ER membrane. To do this, go to the DL query tab and write the query as depicted below. Your results should look something like the screenshot below.

Query (class expression) part_of some 'rough endoplasmic reticulum membrane'	
part_of some 'rough endoplasmic reticulum membrane'	
Execute Add to ontology	
Query results	
Direct subclasses (2 of 2) Query for	
endoplasmic reticulum Sec complex' Direct superclasses	
'translocon complex' Superclasses	
Subclasses (4 of 4)	
Sec61 translocon complex' ⑦ ☑ Direct subclasses	
endoplasmic reticulum Sec complex' Subclasses	
Itranslocon complex'	
owl:Nothing	

Aside

If you like, you can look up the current GO file (open go.owl from http://purl.obolibrary.org/obo/go.owl directly in Protégé -> File-> 'open from URL' to examine how the part of restrictions in the actual ontology were created.

DL query tab

The DL query tab shown below provides an interface for querying and searching an ontology. The ontology must be classified by a reasoner before it can be queried in the DL query tab.

Go to the "basic-dl-query" folder and open "cc.owl". Run the reasoner. Navigate to the DL Query tab.

🛑 😑 🔵 go (http://purl.obolibrary.org/obc	/go.owl) : [/Users/vasilevs/git/BDK14-Ontologies-101/BDK14_exer	cises/basic-dl-query/	(cc.owl]
< > o go (http://p	purl.obolibrary.org/obo/go.owl)		Search
Active Ontology × Entities × Individuals b	v class × DL Query ×		
Class hierarchy:			Meng
	Query (class expression)		
Asserted	Execute Add to ontology		
 bacterial-type hagehum cell cell cortex part' cell junction' cell projection part' cell septum part' cell wall part' cell wall part' cell wall part' cell wall part' contractile fiber part' cytoplasmic vesicle part cytoplasmic reticulum part' cytoskeletal part' cytoskeletal part' cytoskeletal part' cytoskeletal part' cendosomal part' centractile fiber part' cytoskeletal part' cytoskeletal part' cytoskeletal part' centractile reticulum part' cytoskeletal part' cytoskeletal part' centractile fiber part' centractil	Query results	Query for Direct supercl Superclasses Equivalent cla Direct subclasses Subclasses Instances Result filters Name contains Display owl:Th (in superclass resul	asses sses ing ts)
		Reasoner active	Show Inferences

go (http://purl.obolibrary.org/obo/go.owl) : [/Users/vasilevs/git/BDK14-Ontologies-101/BDK14_exercises/basic-dl-query/cc.owl] go (http://purl.obolibrary.org/obo/go.owl) < Search... Active Ontology × Entities × Individuals by class × DL Query × 📫 📪 🛛 🔯 Asserted Query (class expression) organelle owl:Thing cellular_component Execute Add to ontology |axon part' 😑 'axoneme part' -😑 'bacterial-type flagellum Query results 🛑 cell Query for 'cell cortex part' Direct subclasses (5 of 5) 'cell division site part' 'extracellular organelle' Direct superclasses 'cell junction' 'intracellular organelle' 7 😑 'cell part' Superclasses Imembrane-bounded organelle 'cell projection part' Equivalent classes 😑 'cell septum part' Inon-membrane-bounded organelle 2 'cell wall part' vesicle Direct subclasses 'chloroplast part' 'chromosomal part' Subclasses 😑 'cilium part' Subclasses (282 of 282) Instances 'collagen and cuticulin-b 0 'Barr body' 'contractile fiber part' 2 'COPI-coated vesicle' 'cytoplasmic part' ? 'CVT vesicle' 😑 'cytoplasmic vesicle part 'cytoskeletal part' ? **Result filters** 'ER bodv' 'cytosolic part' 2 🛑 'ER to Golgi transport vesicle' Name contains 😑 'endoplasmic reticulum r 2 🛑 'Golgi apparatus' 😑 'endosomal part' 'external encapsulating s 🛑 'Golgi to ER transport vesicle' 😑 'extracellular matrix par 🛑 'Golgi to plasma membrane transport vesicle' 🛜 Display owl:Thing 'extracellular region' in superclass resu 😑 'extracellular region part Golgi-associated vesicle 2 🔽 Displav owl:Nothing Reasoner active 🗸 Show Inferences

Type "organelle" into the box, and make sure "subclasses" and "direct subclasses" are ticked.

You can type any valid OWL class expression into the DL query tab. For example, to find all classes whose members are part_of a membrane, type "part_of some membrane" and click 'execute'. Note the linking underscore for this relation in this ontology. Some ontologies do not use underscores for relations, whereby you'd need single quotes (i.e. 'part of').



The OWL keyword "and" can be used to make a class expression that is the intersection of two class expressions. For example, to find the classes in the red area below, we want to find subclasses of the intersection of the class 'organelle' and the class 'endoplasmic reticulum part'



Note that we do not need to use the "part" grouping classes in the gene ontology (GO). The same results can be obtained by querying for the intersection of the class "organelle" and the restriction "part_of some ER" – try this and

😑 😑 🔵 go (http://purl.obolibrary.org/obc	/go.owl) : [/Users/vasilevs/git/BDK14-Ontologies-101/BDK14	4_exercises/basic-dl-query/cc.owl]
< > ø go (http://	purl.obolibrary.org/obo/go.owl)	Search
Active Ontology × Entities × Individuals b	oy class × DL Query ×	
Class hierarchy: □⊟■⊠	DL query:	
🐮 🖏 🐹 Asserted 📀	Query (class expression)	
▼ ● owl:Thing	organelle and part_of some 'endoplasmic reticulum'	
 cellular_component axon part' axoneme part' axoneme part' 	Execute Add to ontology	
cell	Query results	
Gell cortex part'	Direct subclasses (1 of 1)	Query for
'cell division site part' 'cell junction'	plasmodesmatal endoplasmic reticulum	Oirect superclasses
cell part'	Subclasses (2 of 2)	Superclasses
 Gen projection part Cell septum part' 	'plasmodesmatal endoplasmic reticulum'	👩 🗌 Equivalent classes
Cell wall part'	owl:Nothing	7 Direct subclasses
 Chromosomal part' 		Subclasses
► 🤤 'cilium part'		

We can also ask for superclasses by ticking the boxes as below:

see.

🛑 😑 🔵 go (http://purl.obolibrary.org/obc	o/go.owl) :[/Users/vasilevs/git/BDK14-Ontologies-101/BD	K14_exer	cises/basic-dl-query/cc.owl]
< > ø go (http://	purl.obolibrary.org/obo/go.owl)		Search
Active Ontology × Entities × Individuals b	ov class × DL Query ×		
Class hierarchy:			
	Query (class expression)		
✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	organelle and 'endoplasmic reticulum part' Execute Add to ontology Ouenv results		
 cell 'cell cortex part' 'cell division site part' 'cell junction' 'cell part' 'cell projection part' 'cell septum part' 'cell wall part' 'chloroplast part' 'chloroplast part' 'collagen and cuticulin-b 'collagen and cuticulin-b 'cytoplasmic part' 'cytoplasmic vesicle part 'cytoplasmic vesicle part 	Superclasses (9 of 9) Cutery results Superclasses (9 of 9) Cutery results Superclasses (9 of 9) Cutery results Cutery results Superclasses (9 of 9) Su		Query for
 cytoskeletal part' cytosolic part' cytosolic part' endoplasmic reticulum p endosomal part' e 'entosomal part' e 'external encapsulating part' e 'extracellular matrix part e 'extracellular region' e 'extracellular region part' 	Direct superclasses (2 of 2) reticulum part' organelle Direct subclasses (1 of 1) reticulum'	0	Result filters Name contains Display owl:Thing (in superclass results) Display owl:Nothing
			Reasoner active 🛛 Show Inferences

The 'or' keyword is to used to create a class expression that is the union of two class expressions. For example:



This is illustrated by the red area in the following Venn diagram:



EXERCISE: Basic DL Queries

Go to the 'basic-dl-query' folder in the Exercises directory and open the file cc.owl.

This illustrates basic DL (description logic) queries.

A DL query is a class expression that is constructed using constructs such as 'and' (corresponding to set intersection) and 'some' (see previous example).

Note that the example that follows this one also revisits DL queries.

Constructs used: 'and', 'some'

- 1. open cc.owl
- 2. Turn on the ELK reasoner
- 3. Go to DL query tab
- 4. Find all subtypes of chromosome

Query results	
Direct subclasses (6 of 6)	
🛑 'condensed chromosome'	?
😑 'cytoplasmic chromosome'	?
😑 'nuclear chromosome'	?
🛑 'polytene chromosome'	?
🛑 'sex chromosome'	?
🛑 autosome	?
Subclasses (17 of 17)	0
😑 'W chromosome'	?
🛑 'X chromosome'	2
'XY body'	?
🛑 'Y chromosome'	?
🛑 'Z chromosome'	?
😑 'chloroplast chromosome'	?
🛑 'condensed chromosome'	?
A 'condensed nuclear chromosome'	6

- 1. Experiment with the tab what do the different checkboxes give you?
- 2. Find all parts of a cytoplasm

DL query:	
Query (class expression)	
part_of some cytoplasm	
Add to ontology	
•	
Query results	
Direct subclasses (1 of 1)	Query for
😑 'cytoplasmic part'	Direct superclasses
	Superclasses
Subclasses (958 of 958)	
1-phosphatidylinositol-4-phosphate	Equivalent classes
3-kinase, class IA complex	🗹 Direct subclasses
 '1-phosphatidylinositol-4-phosphate 3-kinase, class IB complex' 	Subclasses
😑 '3-isopropylmalate dehydratase complex' 📀	Instances
'3-methyl-2-oxobutanoate dehydrogenase (lipoamide) complex'	
'3-methylcrotonyl-CoA carboxylase complex, mitochondrial'	Result filters
'6-phosphofructo-2-kinase/fructose-2,6-biph 1 complex'	Name contains
'6-phosphofructokinase complex'	
😑 'A band' 📀	Z Display owl Thing
ADPG pyrophosphorylase complex'	(in superclass results)
'AP-1 adaptor complex'	🔽 Displav owl:Nothing
	Reasoner active 🗹 Show Inferences

1. Find all chromosomes that are part of a cytoplasm

DL query:	
Query (class expression)	
chromosome and part_of some cytoplasm	
Execute Add to ontology	
Query results	
Direct subclasses (2 of 2)	Query for
😑 'mitochondrial chromosome'	Direct superclasses
🤤 'plastid chromosome'	Superclasses
Subclasses (4 of 4)	Equivalent classes
Chloroplast chromosome'	Direct subclasses
'mitochondrial chromosome'	🗹 Subclasses
plastid chromosome'	Instances
owl:Nothing (7)	
	Result filters Name contains Display owl:Thing (in superclass results)
	🔽 Displav owl:Nothing
	Reasoner active 🗹 Show Inferences

NEXT:

1. Find all classes whose instances have a snRNP ('small nuclear ribonucleoprotein complex') as part

DL query:			
Query (class expression)			
has_part some 'small nuclear ribonucleoprotein complex	x'		
Execute Add to ontology			
Query results			
Direct subclasses (15 of 15)		Query for	
'U12-type precatalytic spliceosome'	?	Direct superclasses	
U12-type prespliceosome'	?	Superclasses	
'U2-type precatalytic spliceosome'	?		
'U2-type prespliceosome'	?	Equivalent classes	
● 'U4/U6 snRNP'	?	🗹 Direct subclasses	
'U4/U6 x U5 tri-snRNP complex'	? I	🗹 Subclasses	
😑 'U4atac/U6atac snRNP'	?	V Instances	
'U4atac/U6atac x U5 tri-snRNP complex'	?	Mistances	
😑 'catalytic step 1 spliceosome'	?		
😑 'catalytic step 2 spliceosome'	?		
'commitment complex'	?	Result filters	
'penta-snRNP complex'	?	Name contains	
'post-mRNA release spliceosomal complex'	?		
'post-spliceosomal complex'	?		
'trans spliceosomal complex'	?	Display owl:Thing (in superclass results)	
Subdaccos (24 of 24)		🔽 Displav owl:Nothing	
		Reasoner active 🛛 Show I	nferences

1. Find all classes whose instances have both a U11 snRNP AND a U12 snRNP as parts

DL query:	
Query (class expression)	
has_part some 'U11 snRNP' and 'U12 snRNP'	
Execute Add to ontology	
Query results	
Direct subclasses (1 of 1) Qu	iery for
owl:Nothing ?	Direct superclasses
	Superclasses
Subclasses (1 of 1)	Equivalent classes
	Direct subclasses
Instances (0 of 0)	Subclasses
	Instances
Re	sult filters
Na	me contains
	Display owl:Thing
	(in superclass results)

1. Find all classes whose instances have both a U11 snRNP OR U12 snRNP as parts

DL query:	
Query (class expression)	
has_part some 'U11 snRNP' or 'U12 snRNP'	
Execute Add to ontology	
Query results	
Direct subclasses (2 of 2)	Query for
🧶 'U12 snRNP' 💦 👔	Direct superclasses
'U12-type precatalytic spliceosome'	Superclasses
	Equivalent classes
Subclasses (4 of 4)	
	Direct subclasses
complex'	✓ Subclasses
'U12-type precatalytic spliceosome'	✓ Instances
owl:Nothing	
Instances (0 of 0)	Result filters Name contains
	 Display owl:Thing (in superclass results) Display owl:Nething
	✓ Disblav owi:Nothing
	Reasoner active 🗸 Show Inferences

1. Create a class from this DL query by clicking the 'Add to ontology' button.

11.1 Equivalent classes

The previous example showed the creation of a class restriction. These restrictions were asserted as superclass restrictions, and are sometimes known as *necessary conditions*. That is, if an individual is a member of the 'cell part' then it is necessary for it to also be related to a 'cell' along the 'part of' property.

Things that are cells part of some cell Named cell parts part_of

cell_part subClassOf part_of some cell

Necessary conditions alone mean that individuals can exist that are part of a cell, but are not a type of 'cell part'. In OWL, we can make an even stronger statement and define the 'cell part' class as being equivalent to 'part of' some cell. This is known as a necessary and sufficient condition.

In Protégé we can create an equivalent class restriction inside the 'Equivalent To' slot of the class description view.



Automatic classification
EXERCISE: Basic classification

The file you need is in the basic-classification folder. Follow the instructions below.

This example introduces 'defined classes' and automatic classification. The example involves classification of different ubiquitin ligase complexes. It is based on a subset of the gene ontology (GO) http://purl.obolibrary.org/obo/go.owl with some classes removed for teaching purposes.

Constructs:

- and (intersection)
- equivalence (logical definitions)
- existential restrictions (e.g. part_of some)

Background knowledge for non-GO people:

GO includes pre-composed grouping classes such as 'chromosomal part' and 'nuclear part'.

PART 1: Adding classes and automatically classifying them

- 1. Open basic-classification/ubiq-ligase-complex.owl
- 2. Navigate to 'ubiquitin ligase complex'
- 3. Add a subclass of 'ubiquitin ligase complex' called 'cytoplasmic ubiquitin ligase complex'
- 4. NOTE: do this *directly* under the 'ubiquitin ligase complex' class, don't move things around!
- 5. NOTE: this class already exists in the main GO, but it has been removed for this tutorial example
- 6. Give it a logical definition (equivalence axiom)

	'cytoplasmic ubiquitin ligase complex'		
Class expression editor			
'ubiquitin ligase complex' and	d (part_of some cytoplasm)		
	Cancel OK		

- 1. Synchronize the reasoner
- 2. Find the class you made under 'Class hierarchy (inferred)'
- 3. You should see 'cytoplasmic part' is now inferred to be a parent class of 'cytoplasmic ubiquitin ligase complex'.

Description: 'cytoplasmic ubiquitin ligase complex'	
Equivalent To 🕂	
'ubiquitin ligase complex' and (part_of some	? @ X O
SubClass Of +	
'ubiquitin ligase complex'	?@×0
(cytoplasmic part')	?@
General class axioms + SubClass Of (Anonymous Ancestor)	
cellular_component and (part_of some intracellular)	?@×0
<pre>ellular_component and (part_of some cytoplasm)</pre>	?@
Instances 🛨	
Reasoner activ	e Show Inferences

PART 2: Another example

- 1. Do the same for 'nuclear ubiquitin ligase complex' create the class and add the equivalence axiom.
- 2. Synchronize the reasoner.
- 3. You should see 'nuclear ubiquitin ligase complex' is inferred to be a child of 'nuclear part'.

BONUS:

- 1. Remove the classes you have created.
- 2. Find all 'ubiquitin ligase complex' classes whose instances are in a nucleus in the DL query tab
- 3. Make the class directly from here



EXERCISE: More basic classification

Go to the taxon-union folder and follow the instructions below. This introduces classification using 'or' and 'not'.

This example extends the previous one introducing 'or' (UnionOf) and 'not' (complementOf)

Instructions:

- 1. Open taxslim-with-union.owl
- 2. Briefly check the asserted hierarchy this is a subset of the NCBI taxonomy
- 3. Examine the 'union' classes at the top of the class hierarchy. In particular:
- 4. 'Nematoda or Protostomia' (note that NCBI classifies nematodes as pseudocoelomata)
- 5. 'Viridiplantae or Bacteria'
- 6. Select ELK reasoner and start reasoner (ignore the pop-up windows).
- 7. Navigate to the inferred hierarchy. How have the union terms we examined before been placed? * Note that you'll need to know a little taxonomy here to translate the common names into scientific names (Wikipedia is your friend).
- 8. Create your own grouping classes and classify them. Some examples:
- 9. Mouse or Human
- 10. Mouse or Primate
- 11. Pescetarian dietary component (plant or fish)
- 12. Pescetarian dietary component, more relaxed variant (plant or fish or fungi or mollusc or arthropod)
- 13. Note: don't manually place these in the hierarchy, let the reasoner do this. How can you quickly tell where this class will be placed using the DL query pane?
- 14. Tip: use the DL query tab to test your class expression first
- 15. Create a non-sensical 'transgenic hybrid' class, such as a fly-human which is both a Drosophila and a human. What happens to this when you classify?

- 16. Try using the explanations feature (the '?'). Hint: the explanation is more compact if you choose two sibling taxa e.g. Deuterostome and Protostome
- 17. Remove the hybrid class before moving on.
- 18. Use the DL query tab to find all mammals that are not humans
- 19. Try creating one or more of the following paraphyletic classes. This will involve the 'not' construct
- 20. Nonhuman primate (How can you quickly find the label used for 'Primates' if you know that human = 'Homo sapiens'?)
- 21. Invertebrate
- 22. Invertebrate chordate
- 23. Reptilia, as traditionally defined: (amniote minus aves and mammals)
- 24. A Land mammal
- 25. Classify your classes. What does superclass of the classified class represent? Discuss with your instructorsdoes this reasoned classification reflect an evolutionary history?!
- 26. See HINTS.txt or talk to an instructor if you get stuck on how to classify things.

Object Properties

EXERCISE: Domains and Ranges

Got to the domain-range folder and follow the instructions below. This introduces the concepts of 'domains' and 'ranges' on object properties.

In this exercise, we will illustrate how object properties can be used to subclassify classes that are restricted by them.

- 1. Create a new ontology.
- 2. Add the following class hierarchy:

depressor mandibulae muscle leveator quadrati muscle nerve facial nerve trigeminal nerve

Class hierarchy: owl:Thing		
	Asserted	\Diamond
• owl:Thing nerve 'trigeminal nerve' 'facial nerve' muscle 'leveator quadrati' 'depressor mandibulae muscle' 		

- 1. Add the following:
- An object property named 'is_innervated_by'

💿 😑 🜑 untitled-ontology-188 (http://www.semanticweb.org/vasilevs/ontologies/2018/1/untit				
< > 🔷 untitled-ontology-188				
Active Ontology × Entities × Individ	duals by class \times	DL Query \times	Object Properties	×
Object property hierarchy: owl:topC	DbjectPro¦ Ш⊟∎⊠	Annotations	Usage	
	Asserted ᅌ	Annotations	: owl:topObjectPro	operty
• owl:topObjectProperty		Annotations	Ð	

• Add a domain on 'is_innervated_by' - 'muscle'

💿 😑 untitled-ontology-188 (http://www.semanticv	veb.org/vasilevs/on	tologies/2018/1/untitled-ontology-188) : [http://www.se	manticweb.org/v
< > 🔷 untitled-ontology-18	8	C	Search 🔺
Active Ontology × Entities × Individuals by class ×	DL Query × Obje	Act Properties ×	
Object property hierarchy: is_innervated_by III=II	Annotations Usa	ge	
T 💶 🙀 Asserted ᅌ	Annotations: is_i	nnervated_by	
owl:topObjectProperty Is_innervated_by	Annotations rdfs:label is_innervated	_by	© × 0
	Characte III III	Description: is_innervated_by	
	Functiona	Equivalent To 🛨	
	🗌 Inverse fu		
	Transitive	SubProperty Of	
	Symmetri	Inverse Of 🕀	
	Asymmet	Domains (intersection) 🛨	
	Reflexive	e muscle	?@×0
	- Institution	Ranges (intersection) 🛨	
		Reasoner state out of sync with active ontology	Show Inferences

• To 'depressor mandibulae muscle' a subclass restriction 'is innervated_by some 'facial nerve"

• • • untitled-ontology-188 (http://www.semanticweb.org/vasilevs/ontologies/2018/1/untitled-ontology-188) : [http://www.semanticweb.org/v				
< > (a) untitled-ontology-188 (c) Search				
Active Ontology × Entities × Individuals by class	s × DL Query ×	Object Properties ×		
Class hierarchy: 'depressor mandibulae muscle'		😑 'depressor mandibulae muscle' — HP:0410115 — htt	p://purl.obolibrary	
	Asserted ᅌ	Class Annotations Class Usage		
ewl:Thing		Annotations: 'depressor mandibulae muscle'		
• erve		Annotations 🛨		
 'facial nerve' 		rdfs:label	$\textcircled{0} \times \textcircled{0}$	
e muscle		depressor mandibulae muscle		
Idepressor mandibulae muscle'		dc:creator		
		http://orcid.org/0000-0001-5208-3432		
		dc:date [type: xsd:dateTime]		
		2018-02-06T01:06:37Z		
		Description: 'depressor mandibulae muscle'		
		Equivalent To 🛨		
		SubClass Of		
		Is_innervated_by some 'facial nerve'		

• To 'leveator quadrati' a subclass restriction 'is innervated_by some 'trigeminal nerve'

📀 😑 untitled-ontology-188 (http://www.semanticweb.org/vasilevs/ontologies/2018/1/untitled-ontology-188) : [http://www.semanticweb.org/v				
< > • untitled-ontology-188				
Active Ontology × Entities × Individuals by class	s × DL Query ×	Object Properties ×		
Class hierarchy: 'leveator quadrati'		😑 'leveator quadrati' — HP:0410116 — http://purl.obo	library.org/obo/HP	
	Asserted ᅌ	Class Annotations Class Usage		
▼ ● owl:Thing		Annotations: 'leveator quadrati'		
• erve		Annotations 🛨		
Ingeninal nerve		rdfs:label	@ × 0	
muscle		leveator quadrati		
'depressor mandibulae muscle'		dc:creator	$\odot \times \odot$	
		http://orcid.org/0000-0001-5208-3432		
		dc:date [type: xsd:dateTime]	$\odot \times \odot$	
		2018-02-06T01:06:43Z		
		Description: 'leveator quadrati'		
		Equivalent To 🛨		
		SubClass Of		
		is_innervated_by some 'trigeminal nerve'	?@X0	

1. Now run the reasoner and inspect the inferred class hierarchy. You should see a new classification under the 'muscle' class.

