# protege-tutorial Documentation

Release 0.5

protege-tutorial

Sep 18, 2017

# Contents:

1	Initial Preparation	3
	1.1       Olifilitie Logini	3
	1.3    Install Protege 5 for Ontology Development	3
2	Starting Protégé	5
	2.1 The Protégé UI	7
3	The entities tab	11
	3.1 Creating your first class	13
	3.2 Renaming an entity	15
	3.4 Adding annotations properties	1/
	3.5 Setting label rendering	20
	3.6 Creating the class hierarchy	22
	3.7 EXERCISE: Basic Subclass Hierarchy	23
4	Class description view	25
5	Protégé plugins	31
	5.1 Annotation search plugin	32
6	Disjointness	33
	6.1 Reasoning and inconsistency checking	35
	6.2 EXERCISE: Basic Disjoint	39
7	Making Ontologies from Spreadsheets and templates (Optional)	41
8	Object properties	45
	8.1 Create an object property	45
9	OWL class restrictions	49
	9.1 Superclass restrictions	49
	9.2 Existential tree plugin	52
	9.3 EXERCISE: Basic Restrictions	53
10	DL query tab	55
	10.1 EXERCISE: Basic DL Queries	59

#### 11 Equivalent classes

12	Autor 12.1 12.2 12.3 12.4	matic classification         EXERCISE: Basic classification         EXERCISE: Classification using Unions (Optional)         Introduction to property axioms         Excercise:	65 65 65 65
13	<b>Impo</b> 13.1 13.2	prts         EXERCISE: response to stimulus         Ontology libraries	<b>6</b> 2 68 69
14	14.1	Catalog files	<b>7</b> 1 73

61

Protégé 5 tutorial for GO Editors. V2

# CHAPTER 1

#### **Initial Preparation**

#### **GitHub Login**

Obtain a login at GitHub (https://github.com/) and send your login name so you can be added to the GO Organization (https://github.com/geneontology). This is our repository for ontology files and issue trackers.

#### **Clone this repository**

Visit https://github.com/geneontology/protege-tutorial

click "Clone or Download"

This will give you a URL you can use

git clone https://github.com/geneontology/protege-tutorial.git

These files will be under protege-tutorial on your drive.

#### **Install Protege 5 for Ontology Development**

Download and install the latest version of Protege (5.1) for ontology development.

http://protege.stanford.edu/products.php#desktop-protege

The exercises assume that you have the tutorial directory checked out from GitHub. If for some reason you're unable to do this, you can download individual files from:

https://github.com/geneontology/protege-tutorial

# CHAPTER 2

### Starting Protégé

When you start Protégé a welcome dialog appears where you can choose to

- 1. Create a new OWL ontology
- 2. Open an existing OWL ontology
- 3. Open an ontology from a particular URI (e.g http://purl.obolibrary.org/obo/go.owl).

⊗ ⊖ ⊕

Welcome to Protégé

Create new OWL ontology
Open OWL ontology
Open OWL ontology from URI
Open from the TONES repository
r Open recent
http://purl.org/obo/owl/biological_process /Users/jupp/Documents/teaching/GO_tutorial/tutorial/basic-subclass/chromosome-parts.owl http://purl.obolibrary.org/obo/ro.owl /Users/jupp/dev/tomcats/apache-tomcat-6.0.29/webapps/kupkb/WEB-INF/classes/resources/ontologies/cto.owl /Users/jupp/Documents/e-lico/svn/trunk/Public/kupkb/kupkb.owl http://purl.obolibrary.org/obo/cl.owl /Users/jupp/Dropbox/JuppKlein/KUP/experiment_descriptions.owl /Users/jupp/Dropbox/JuppKlein/KUP/experiment_ontologies/schanstra_bachvarov.owl /Users/jupp/Dropbox/mouse GOA/ontology/mouse-phene.owl

More actions

We will begin by selecting "Create new Ontology". In the next dialog we will create an IRI that can 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/chromosome-parts.owl.

● ○ ●	Create ontology wizard
	Ontology ID
	Please specify the ontology IRI. The ontology IRI is used to identify the ontology in the context of the world wide web. It is recommended that you set the ontology IRI to be the URL where the latest version of the ontology
	will be published. If you use a version IRI, then it is recommended that you set the version IRI to be the URL where this version of the ontology will be published.
	http://purl.cholibran/.org/cho/ourl_tutorial/chromosome_parts.ourl
	http://pun.obonbrary.org/obo/owi-tutonar/chromosome-parts.owi
	Version IRI
<b>V</b>	http://www.semanticweb.org/ontologies/2012/0/Ontology1326971196330.owl
	Enable Version Iri
	Default base
	Go Back Continue Cancel

You will also want to save this ontology file to your hard disk. Use the next dialog to save the ontology file somewhere convenient. Finally you will want to choose a format for your ontology file. Protégé allows you to save your ontology in a variety of OWL formats, including the OBO 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.

Ontology Format         Please select the format in which the ontology will be saved (by default).         Note that the Manchester OWL Syntax does not support all OWL constructs (e.g. GCI's and anotations of undeclared entities) and the Latex format cannot be reloaded         Image: Construct of the Manchester OWL Syntax does not support all OWL constructs (e.g. GCI's and anotations of undeclared entities) and the Latex format cannot be reloaded         Image: Construct of the Manchester OWL Syntax does not support all OWL constructs (e.g. GCI's and anotations of undeclared entities) and the Latex format cannot be reloaded         Image: Construct of the Manchester OWL Syntax does not support all OWL Syntax does not suport	$\bigcirc \bigcirc \bigcirc$	Create ontology wizard
Please select the format in which the ontology will be saved (by default). Note that the Manchester OWL Syntax does not support all OWL constructs (e.g. GCI's and annotations of undeclared entities) and the Latex format cannot be reloaded           Image: Comparison of Comparison of Comparison of Undeclared entities         Image: Comparison of Comparison	C	Ontology Format
Note that the Manchester OWL Syntax does not support all OWL constructs (e.g. GCI's and annotations of undeclared entities) and the Latex format cannot be reloaded V IDF/XML       OWL/XML         OWL /XML       OWL Functional Syntax         OBO 1.2 flat file       KRSS2 Syntax         Latex       Turtle		Please select the format in which the ontology will be saved (by default).
V RDF/XML OWL_XML WML Functional Syntax Manchester OWL Syntax OBO 1.2 flat file KRSS2 Syntax Latex Turtle		Note that the Manchester OWL Syntax does not support all OWL constructs (e.g. GCI's and annotations of undeclared entities) and the Latex format cannot be reloaded
		V RDF/XML OWL/XML OWL Functional Syntax Manchester OWL Syntax OBO 1.2 flat file KRSS2 Syntax Latex Turtle
Go Back Finish Cancel	L	Go Back Finish Cancel

# The Protégé UI

After a few seconds Protégé will launch into the main user interface. The protégé interface follows a basic paradigm of Tabs and Panels. The layout of tabs and panels is configurable by the user. By default protégé launches with 5 main tabs.

Chromosome-parts (http://purl.obolibran	.org/obo/owl-tutorial/chromosome-parts.owl) - [/Users/jupp/Documents/teaching/GO_tutorial/ontology_files/chromo	osome-parts.owl]
Control Con	arts.owl)	
Activ	e Ontology Entities Classes Object Properties Data Properties Individuals DL Query	
Ontology appotations:		
	DL metrics Ontology metrics	
	DL metrics:	
	A C	
	$\mathcal{AL}$	
	Symbol key	
	Attributive language. This is the base language which allows:	
	• Atomic negation (negation of concepts that do not appear on the left hand side of axioms) • Concept intersection	
	• Universal restrictions	
	<ul> <li>Limited existential quatification (restrictions that only have milers or 1 hing)</li> </ul>	
	${\cal F}{\cal C}^-$ A sub-langauge of AL, which is obtained by disallowing atomic negation	
	5 2 7 0	
	${\cal FL}_o$ A sub-language of FL", which is obtained by disallowing limited existential quantification	
	${\cal C}$ Complex concept negation	
	${\cal S}$ An abbreviation for AL and C with transitive properties	
	${\cal H}$ Role hierarchy (subproperties - rdfs:subPropertyOf)	
	Nominals. (Enumerated classes or object value restrictions – owl:oneOf, owl:hasValue)	
	$\overline{\mathcal{I}}$ Inverse properties	
	Cardinality restrictions (owl:Cardinality, owl:minCardianlity, owl:maxCardinality)	
	Ortology imports OntoGraf Import View Ontology Prefixes General class axioms RDF/XML rendering	
Imported ontologies:		080
Durect imports		
Indirect Imports		
		the maconer click Bearconer - Start maconer - dist
	10 u	se the reasoner click keasoner->start reasoner 🗹 Show Inferen

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 (See later), so this top bar is very important as it lets you know which ontology you are viewing and editing.

•	0 0 chromosome-parts (http://purl.obolibrary.c	ng/obo/owi=tutorfal/chromosome=parts.owl) = [/Uscro/jupp/Documents/teaching/GO_tutorial/ontology_files/chromosome-parts.owl]	
	chromosome-parts (http://purl.obolibrary.org/obo/owl-tutorial/chromosome-parts)	rts.owl)	(A) (Q.
	Active	Ontology Entities Classes Object Properties Data Properties Individuals DL Overy	
	Antology apparent/seco		
	Anastations:	DL metrics Ontology metrics	
1		DL metrics:	
		A C	
		AL	
		Symbol key	
		Attributive language. This is the base language which allows:	
		• Atomic negation (negation of concepts that do not appear on the left hand side of axioms) • Concept intersection	
		Universal restrictions     Universal restrictions     Limited existential qualification (restrictions that only have fillers of Thing)	
5		· · · · · · · · · · · · · · · · · · ·	
		${\cal FL}^-$ A sub-langauge of AL, which is obtained by disallowing atomic negation	
		$\mathcal{FL}_o$ A sub-language of FL", which is obtained by disallowing limited existential quantification	
		${\mathcal C}$ Complex concept negation	
5		${\cal S}$ An abbreviation for AL and C with transitive properties	
		${\cal H}$ Role hierarchy (subproperties - rdfs:subPropertyOf)	
		O Nominals. (Enumerated classes or object value restrictions - owl:oneOf, owl:hasValue)	
		$\mathcal{I}$ Inverse properties	
		Cardinality restrictions (owl:Cardinality, owl:minCardianlity, owl:maxCardinality)	
1			
1		Ontology Imports OntoGraf Import View Ontology Prefixes General class axioms RDF/XML rendering	
	Imported ontologies:		
1			
	Indirect Imports		
		To use the reasoner cli	ck Reasoner->Start reasoner 🗹 Show Inferences
-			_

The panel on the left 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.

# 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. For example, the classes tab allows us to view and edit the classes in the ontology, and similarly the properties tab focuses on the properties in the ontology. The primary tab where you will spend most of your time is the entities tab.

	Active Ontology	Entities	Classes	Object Properties	Data Properties	Individuals	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 displays 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.

Chromosome-parts (http://purl.obolibrary.org/obo/owl-tutorial/chromosome-parts (http://purl.obolibrary.org/obo/owl-tutori	mosome-parts.owl) - [/Users/jupp/Documents/teaching/GO_tutorial/ontology_files/chromosome-parts.owl]	
chromosome-parts (http://purl.obolibrary.org/obo/owl-tutorial/chromosome-parts.owl)	4	88 Q
Active Ontology Entities C	asses Object Properties Data Properties Individuals DL Query	
Class hierarchy Class hierarchy (inferred) Search Annotations	Flass dependations Class lisane Manthester surfax rendering	
Class hierarchy: Thing 0800	Annotations: Thing	DESC
😰 🕸 - 👿	Annotations	
Thing		
a		
	Description: Thing	Desc
	Equivalent classes 🕥	
	Superclasses	
Object property interactive Data property interactive individuals by type Annotation property interactive Datatypes	Inherited anonymous classes	
	Members	
topObjectProperty		
	Keys 💿	
	Disjoint classes @	
	Disjoint union of 🕞	
	To use the reasoner clic	k Reasoner->Start reasoner 🗹 Show Inferences

If you select the topObjectProperty property in the panel at the bottom, the right hand side panels will change context.

Chromosome-parts (http://purl.obolibrary.org/obo/owl-tutorial/chromosome-parts (http://purl.obolibrary.org/obo/owl-tutori	omosome-parts.owl) - [/Users/jup	p/Documents/teaching/GO_tutorial/ontology_files/chromosome-parts.owl]
Comparison (http://purl.obolibrary.org/obo/owl-tutorial/chromosome-parts.owl)		÷) 🕷 (Q,
Active Ontology Entities C	lasses Object Properties Dat	a Properties Individuals DL Ouery
(netre ontoing)		arreperous mannadas se query
Class hierarchy (inferred) Search Annotations		Annotations Object Property Usage
Class hierarchy: Thing OBEO	Annotations: topObjectPrope	rty 00880
	Annotations 🕒	
	Characteristics: topOb 00000	Description: topObjectProperty DBB
	Functional	Domains (intersection)
	Inverse functional	
		Ranges (Intersection)
		Equivalent object properties 💿
	Symmetric	
	Asymmetric	Super properties 🕥
	Reflexive	Inverse properties 🕣
	Irreflexive	
Alter and the second difference in the second		Disjoint properties 💿
Object projectly metarchy bata property metarchy individuals by type Annotation property metarchy batatypes		Property chains 🚯
topObjectProperty		
		To use the reasoner click Peasoner - Start reasoner - Start

# Creating your first class

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

	Class hierarchy Class hierarchy (inferred) Search Annotations	
Class hierarchy: Thing		
🐮 🕼 · 🕱		
Thing		

There are three buttons at the top of the class hierarchy view. These allow you to add a subclass, add a sibling or delete a selected class. We will use the 'add subclass' button to add a child class to OWL thing. For now, simply name this class cellular\_component.

	Class hierarchy Class hierarchy (inferred) Search Annotations	
Class hierarchy: Thing		
😫 🔹 · 🕱		
Thing		
	Please enter a class name	
	cellular_component	
	Ignore entity creation preferences	
	http://purl.oboilbrary.org/obo/owi-tutoriai/chromosome-parts.owi#ceilular_componen	
	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 the unique IRI for this entity. If you hover over this class with your mouse you will see the full IRI for this class

Class hierarchy Class hierarchy (inferred) Search Annotations	
Class hierarchy: cellular component	
Thing cellular_component	

### **Renaming an entity**

We can change the IRI for a concept using the rename function in the refactoring menu. Rename the cellular\_component class to use its proper IRI from the Gene Ontology (http://purl.obolibrary.org/obo/GO\_0005575)

🗯 org.protege.osgi.framework.Launcher	File Edit View Rea	soner Tools	Refactor SKOSEd Window Help	-
• • •       • • • <t< td=""><td>chromosome-parts (htt library.org/obo/owl-tutor</td><td>rial/chromosome</td><td>Rename entity     #U       Rename multiple entities     Change ontology URI       Convert entity URIs to labels     Convert entity URIs to labels</td><td>Change th</td></t<>	chromosome-parts (htt library.org/obo/owl-tutor	rial/chromosome	Rename entity     #U       Rename multiple entities     Change ontology URI       Convert entity URIs to labels     Convert entity URIs to labels	Change th
Class hierarchy	Class hierarchy (inferred) Sear	ch Annotations	Convert property assertion on class/individual puns to annotations Coerce data property values into property range	Class
Class hierarchy: cellular_component Class hierarchy: cellular_component Class hierarchy: cellular_component Class hierarchy: 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 the check the "Show full IRI" box so you can edit the full IRI.

	Active Ontology Entities Classes Object Properties Data
Class hierarchy       Class hierarchy (inferred)       Search Annot         Class hierarchy:       cellular_component       Image: Class hierarchy (inferred)       Search Annot         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Search Annot         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)         Image: Class hierarchy (inferred)       Image: Class hierarchy (inferred)       Image: Class	Annotations: cellular_compon
G Chang	e entity UKI
http://purl.obolibrary.org/obo/owl-tutorial/	chromosome-parts.owl#cellular_component
Change all entities with this URI	Show full IRI
	Cancel
	Description: cellular_componen
	Superclasses 🕣
Object property hierarchy Data property hierarchy Individuals by type Annotatic Object property hierarchy: topObjectProperty	In property hierarchy Datatypes Inherited anonymous classes

And then paste or type in the correct GO URI.

		Active ontology	Entertes	ciusses	objecthopenies
Class hierarchy: cellular_com	Class hierarchy Class hierarchy (inferred) ponent	Search Annotations			Annotations: cellular_con
Thing cellular_componer	nt				Annotations 🛞
	000	Change entity UPI			
		Change entity OK			
	http://purl.obolibrary.org/obo/G	O_0005575			
	Change all entities with this U	RI		Show fi	III IRI
				511011 10	
		(	Cancel	0	ĸ
				"	
					scription: cellular_comp
				Eq	uivalent classes 🕂 perclasses 📀

Now the correct GO URI appears in the ontology. Obviously you don't want to have to rename every entity you create when building your own ontology. Luckily Protégé provides a "New Entities" preferences panel where you can specify how new IRI should be created.

#### **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. You can set an id strategy using the "New Entities" tab under the Protégé preferences.

Set your new entity preferences as follows:

General Ne	w Entities New Ontologies OWLViz Plugins Reasoner Renderer Save Tree preference
Entity URI	
Start with:	O Active ontology URI
	Specified URI: http://purl.cholibrary.org/obc
	Specifica on http://pun.obombiary.org/obo
Followed by:	○ #
	$\odot$ /
	0:
End with:	O User supplied name
	Auto-generated ID
Entity Label	
Create lab	pel from user supplied name
Create Jak	pel from auto-generated ID
	in nom auto-generated ib
Same as la	abel renderer
Custom la	ahel
Customia	
LIDI	
URI http:	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang	<pre>://www.w3.org/2000/01/rdf-schema#label</pre>
URI http Lang	://www.w3.org/2000/01/rdf-schema#label
URI http Lang	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric (	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric ( Unique an	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric ( Unique an	://www.w3.org/2000/01/rdf-schema#label  d ID (pseudo random) Prefix: GO_ (iterative) Suffix: Digit count: 7 \$
URI http: Lang Auto-generated Numeric ( Numeric ( Unique an	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric ( Unique an	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric ( Unique an	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric ( Unique an	://www.w3.org/2000/01/rdf-schema#label
URI http: Lang Auto-generated Numeric ( O Numeric ( Unique an	<pre>c//www.w3.org/2000/01/rdf-schema#label  d ID (pseudo random) Prefix: GO (iterative) Suffix:  Ind meaningless Digit count: 7 \$ Start: 6,000,000 \$ End: -1 \$ Remember last ID between Protégé sessions</pre>
URI http: Lang Auto-generated Numeric ( ONumeric ( Unique an	<pre>c//www.w3.org/2000/01/rdf-schema#label  d ID (pseudo random) Prefix: GO</pre>

For ontologies other than GO, change the value of the prefix. Note that all OBO library ontologies should use the obolibrary.org Specified URI value.

#### Adding annotations properties

You can add annotations (such as labels, descriptions, xrefs, etc.) to any OWL entity using Protégé. The panel on the right, named Annotations is where these annotations are added. Use this panel to add a "cellular\_component" label to the class you created previously.

000	chromosome-parts (http:/	//purl.obolibrary.org/obo/owl-tutorial/chrom	osome-parts.owl) - [/Users/jupp/Documents/teaching/GO_tutorial/ontology_files/chromosome-parts.owl]	
🗢 🔿 (\$ chromosom	e-parts (http://purl.obolibrary.org/obo/owl-tutorial	/chromosome-parts.owl)	;	88 Q.
		Active Ontology Entities Clas	sses   Object Properties   Data Properties   Individuals   DL Query	
	Class hierarchy Class hierarchy (inferred) Search A	Annotations	Class Annotations Class Usage Manchester syntax rendering	
Class hierarchy: GO_000	5575	0840	Annotations: GO_0005575	DEXO
📽 🔹 🕱			Annotations	
Thing				
GO_0005575				
			Description: GO_0005575	180
			Equivalent classes 📀	
			Superclasses 🕥	
Object property hi	Prarchy Data property hierarchy Individuals by type Annu	otation property hierarchy Datatypes	Inherited anonymous classes	

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 a comment such as "created during Berkeley tutorial". You can also create your own annotation properties.

Note that for practical purposes you will start from an existing OWL file that 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.

00	CO_0005575
	Constant Entity IRI IRI Editor Property values
backwardCompatibleWith comment	Value
<pre>—deprecated —incompatibleWith</pre>	cellular_component
label	
seeAlso	
versioninio	
	Type 主 Lang 🔻
	Cancel

### Setting label rendering

You can change how Protégé renders entities. It is common to want to view entities by the their label, rather than identifiers. You can tell Protégé to render on any annotation property you choose. Lets render all entities by their class label.

000	chromosome-parts (http://purl.obolibrary.org/obo/owl-tutorial/chromo	osome-parts.owl) - [/Users/jupp/Documents/teaching/GO_tutorial/ontology_files/chromosome-parts.owl]	
\$	Chromosome-parts (http://purl.obolibrary.org/obo/owl-tutorial/chromosome-parts.owl)	;) (0, (0,	
	Active Ontology Entities Clas	ises Object Properties Data Properties Individuals DL Query	
	Class hierarchy Class hierarchy (inferred) Search Annotations	Class Apportations Class Usage Manchester syntax rendering	
	nierarchy: CO_0005575 WBB/2	Annotations: CO_0005575	0880
•	Thing 60 0005575	Accention and a second se	000
		Description: GO_0005575	0880
		Equivalent classes 💿	
		Superclasses 🕲	
	Object property hierarchy Data property hierarchy Individuals by type Annotation property hierarchy Datatypes	Inherited anonymous classes	
	E Kalendari (Kalendari	Members 💿	
1.	ntopObjectProperty	Keys 💿	
		Disjoint classes 🕥	
		Disjoint union of 🕥	

In the view menu choose render by label

é org.protege.osgi.framework.Launcher File Edit	View Reasoner Tools Refactor SKOSEd Wi	Help	
O Chromoso	Render by name (rdf:id)	some-parts.owl) - [/Users	/jupp/Documents/teaching/GO_tutorial/ontology_files/ch
△ → A chromosome_parts (http://purl.obalibrany.org/	✓ Render by label (rdfs:label)		
Chromosome-parts (http://puntobombrary.org/	Render by annotation property		
	Custom rendering	es Object Properties	Data Properties Individuals DL Query
	✓ Show the imports closure of the active ontology		
Class hierarchy Class hierarchy (	Show all loaded ontologies		Class Annotations Class Usage M.
Class hierarchy: cellular_component	Show only the active ontology	Annotations: cellular co	nponent
2: 2+ · X		Annotations	
▶ ●Thing		label	
		"cellular_component"	

The cellular\_component class will now render in the hierarchy view using the value of the label annotation property.

00	chromosome-parts (http://purl.ol	oolibrary.org/obo/owl-tutorial/chro	moso	ome-parts.owl) - [/Users/jupp/Documents/teaching/GO_tutorial/ontology_files/chromoso	me-parts.owl]		
💠 🐟 chromosom	e-parts (http://purl.obolibrary.org/obo/owl-tutorial/chrome	some-parts.owl)			\$	<i>8</i> 8 Q	
		Active Ontology Entities Cl	asses	s Object Properties Data Properties Individuals DL Query			
	Class hierarchy Class hierarchy (inferred) Search Annotation	5		Class Annotations Class Usage Manchester	syntax rendering		
Class hierarchy: cellular	component		Z	Annotations: cellular_component			0800
Thing				Annotations 💿			
cellular_comp	onent		1	label "cellular component"			080
http://pu	rl.obolibrary.org/obo/GO_0005575			central_component			
			De	escription: cellular_component			0800
			Equ	iquivalent classes 🕥			
			Su	uperclasses 🗊			
Object property hi	enerchy Data property hierarchy Individuals by type Annotation pro	perty hierarchy Datatypes	Int	nherited anonymous classes			

#### 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-Edit). 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'.

Using the class hierarchy view create a small section of the cellular component branch of the GO as shown in the following screenshot. Play around with add subclass, add sibling and the drag and drop functionality.



Don't bother to add textual definitions, synonyms, etc. at this stage.

After you have become familiar, you can save your efforts or discard them – you won't need this ontology from here on.

#### **EXERCISE: Basic Subclass Hierarchy**

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

Follow the instructions in the README file

# CHAPTER 4

### Class description view

We have seen how to add subclasses in the class hierarchy. Another way to do the same thing is via the Class description view.

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 has a superclass (cellular\_component). Using the + button we could add more superclasses to the cell class.



Select the 'intracellular organelle part' class in your ontology. Using the superclasses + button, add the 'organelle' class as a super class.

	00	'intra	cellular organelle pa	ırt'		8
nherite	<ul> <li>Class expression</li> </ul>	n editor	Object restriction	creator Cla	ss hierarchy	
Membe						
Ceys 🧲						
Disjoint						
Disjoint						
				Canad		

There are various ways to assert a superclass. The simplest it to just type in the class expression editor. Hint: Pressing CTRL + SPACE allows you to autocomplete on a term.

r \varTheta	0 0 'intracellular organelle part'	×
ite	Class expression editor Object restriction creator Class hierarchy	
oe	organ organelle organelle part'	l
n		
nt		l
L		
L		
	Cancel	
L	Cancel	

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



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.

Active Ontology Entities Classes Object Proper	ties Data Properties Individuals SKOS view DL Query OPPL OPPL Patterns				
Class hierarchy Class hierarchy (inferred) Search Annotations	Class Annotations Class Usage Manchester syntax rendering				
ass hierarchy: 'intracellular organelle part' DEBO	Annotations: 'intracellular organelle part'				
	Annotations 💽				
<pre>Ining V ccellular_component Ccell Ccell part' V ccell part'</pre>	definition <sup>*</sup> A constituent part of an intracellular organelle, an organized structure of distinctive morphology and function, occurring within the cell. Includes constituent parts of the nucleus, mitochondria, plastids, vacuoles, vesicles, ribosomes and the cytoskeleton but excludes the plasma membrane. <sup>*</sup> ^string				
<ul> <li>Granelle</li> <li>Fintracellular organelle part</li> <li>Fono-membrane-bounded organelle</li> <li>Gorganelle part</li> <li>Fintracellular organelle part</li> </ul>	has_obo_namespace "cellular_component"^>tring	080			
	id "GO:0044446"^^string	080			
	in_subset ♦ gosubset_prok	080			
	label "intracellular organelle part"^^string	080			
	Description: 'intracellular organelle part'	006			
	Equivalent classes 🕥				
	Superclasses 💿 • 'organelle part'	00			
	organelle	@ X			
	Inherited anonymous classes				
	Members 💿				
	Keys 🚱				
Object property hierarchy Data property hierarchy Individuals by type	Disjoint classes 💿				
oject property hierarchy:	Disjoint union of 🕒				
■tonObjectProperty					

# CHAPTER 5

### Protégé plugins

Protégé is built on a plugin architecture. There is an activity 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.

You can find the plugin library in the Protégé preferences. Select the check for downloads button to see the list of available plugins.

O Chromosome-parts (ht	tp://purl.obolibrary.org/obo/test/chromosome-parts	s.owl) - [/Users/jupp/Documents/teaching/GC	_tutorial/tutorial/basic-subclass/chromo	some-parts-interim.owl]
chromosome-parts (http://purl.obolibrary.org/obo/test/chro	mosome-parts.owl)			÷ 86 Q
	Active Ontology Entities Classes Object Prop	erties Data Properties Individuals SKO	S view DL Query OPPL OPPL Patte	rns
Ontology annotations:	0803		Di motrice - Ontolomi motrice	
Annotations	DI metrics:		outreates outoogy means	
has_obo_format_version	OL Expressivity			
"1.2"^^string	S			
	0			
	000	Preferences		
	General New Entities New Ontologi	es OWLViz Plugins Reasoner Re	nderer Save Tree preferences	
	Auto update			
	Automatically check for plugin upda	Automatic U	1	
	Check for updates now	Downloads		
		Inst Name Cur	rent versi Available version	
	Plugin registry	Browser View (OWLDoc)	1.2.1	
	This is the location P4 will check to see	RacerProTG Reasoner	1.1.8	
	http://smi-protege.stanford.edu/prote	OWLlink HTTPXML Reasoner	0.7.0	
		OWL2Query Plugin	0.1.0	
	Check for downloads now	SADI service plugin	1.2.1	
		Code Generation Plugin	1.0.0.2011_07_13_0	
		Annotation Template View	200	
		Plugin info		
		Author:		
		License:		
Imported ontologies:				
Direct Imports				
Indirect Imports				
		Always check for updates on startup.		
			Not now Install	
				To use the reasoner click Reasoner->Start reasoner  ✓ Show

Install the Annotation Search View and the Existential tree view

#### Annotation search plugin

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 -> annotation search. Once selected you can choose to drop this panel over any existing panel in Protégé. We recommend that you drop this panel to the right of the class hierarchy view, on top of the existing annotation view panel.

etromorome parts (http://aud.ehelibrapu/	Mindow Help		N/tutorial/basic_subclass/chromosome_parts_interin	n oull
Companelle     Companelle part     Companelle part     Companelle part	Views > Tabs > Create new tab Delete custom tab Unport tab Export current tab Store current tab.us Store current tab.us tab Reset selected tab to default state Increase font size %= Decrease font size %= Timestam [of y console Look & Feel  Refresh User Interface	Annotation property views Class views Data property views Datatype views Individual views Object property views Object property views Object property views Object views SKOSEd views SKOSEd views	VI/Lutorial/basic-subclass/chromosome-parts-interin     DL_Query_OPP_OPP_Patterns     Axiom annotations     Manchester syntax entity rendering     Query     Search Annotations     Selected entity	nowij :) 00 Q. 
		Characteristics: part 0/H0/2 Functional Inverse functional Transitive Symmetric Asymmetric	Description: part_cof Domain interaction Exopos interaction Equivalent object properties Super properties	0.88 0.88
		Reflexive	Inverse properties	

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

[the instructors will describe the concept of disjoint classes here]

At the top of our class hierarchy we have cell, cell 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.

We want to assert that organelle and 'organelle part' are disjoint. To do this first select the cell class. In the class description view select the + button next to disjoints. You can use CTRL to select multiple classes.



Active Ontology Entities Classes Object Properties	Data Properties   Individuals   SKOS view   DL Query   OPPL   OPPL Patterns	
Class hierarchy Class hierarchy (inferred) Search Annotations hierarchy: organelle	Class Annotations Class Usage Manchester syntax rendering	Mein
X Thing Cellular_component Cell Cell Cell Cell Cell Cell Cell Cel	Annotations definition "Organized structure of distinctive morphology and function. Includes the nucleus, mitochondria, plastids, vacuoles, vesicles, ribosomes and the cytoskeleton, and prokaryotic structures such as anammoxosomes and pireluluosomes. Excludes the plasma membrane."Avstring	@ <b>XO</b>
► ●'organelle part'	id "GO:0043226"^^string	080
	in_subset ♦ gosubset_prok	@XO
	database_cross_reference Wikipedia:Organelle"^^string	@XO
	database_cross_reference "NIF Subcellular:sao1539965131"^^string	@X0
	has oho namesnare	<b>MX0</b>
	Description: organolle O organelle	
	Guivalent class     Class hierarchy     Expression editor       Superclasses     Image: Class hierarchy     Expression editor       Cellular     Image: Class hierarchy     Expression editor	@×
	mentee anon Members > > cell part'	
	Y organelle part > ● 'intracellular organelle part'	
Bject property hierarchy Individuals by type	Keys   Keys	H

Note that the directionality is irrelevant.

### 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 all 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 (HermiT, Pellet or Fact++ will work). Once a reasoner is highlighted, select "Start reasoner" from the menu.



A progress bar will indicate when classification is complete. The 'intracellular organelle part' class will have changed to red indicating that the class is now *unsatisfiable*.



You can also see unsatisfiable classes using the "Class hierarchy (inferred)" panel next to the "Class hierarchy" panel. (Note - in Protege 5.1, the toggle the "Asserted button" to switch to the inferred view.) Here you will a special class called Nothing. When we previously said that all OWL classes are sublasses 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 sublasses or equivalent to OWL Nothing. The "class hierarchy (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 following screen shot. 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.



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.

● ○ ○ Explanation for 'intracellular organelle part' EquivalentTo Nothin	g
Axioms	
DisjointClasses: cell, organelle, 'organelle part', 'cell part'	<b>@</b> ×
intracellular organelle part' SubClassOf 'organelle part'	@X
Intracellular organelle part' SubClassOf organelle	<b>@</b> ×
OK	
	ation contraction with the set to a book

Remove the axiom, and resynchronise the reasoner from the reasoner menu.

## **EXERCISE: Basic Disjoint**

Go to the basic-disjoint folder in the tutorial directory and follow the instructions in the README

## Making Ontologies from Spreadsheets and templates (Optional)

#### Go to the template-examples folder

Open "test1.csv" in Excel. This has a sample set of plant traits

	A1 🛟 🛞	💿 ( <i>fx</i> iri				
_	A	В	С	D	E	F G
1	iri	label	entity	entity label	quality	quality label
2	DEMOTRAIT_0000002	seed morphology	PO_0009010	seed	PATO_0000051	morphology
3	DEMOTRAIT_0000003	seed shape	PO_0009010	seed	PATO_0000052	shape
4	DEMOTRAIT_0000004	seed size	PO_0009010	seed	PATO_0000117	size
5	DEMOTRAIT_0000005	seed weight	PO_0009010	seed	PATO_0000128	weight
6	DEMOTRAIT_0000007	morphology	PO_0009011	plant structure	PATO_0000051	morphology
7	DEMOTRAIT_0000008	shape	PO_0009011	plant structure	PATO_0000052	shape
8	DEMOTRAIT_0000009	size	PO_0009011	plant structure	PATO_0000117	size
9	DEMOTRAIT_0000010	weight	PO_0009011	plant structure	PATO_0000128	weight
10	DEMOTRAIT_0000012	leaf morphology	PO_0009025	vascular leaf	PATO_0000051	morphology
11	DEMOTRAIT_0000013	leaf shape	PO_0009025	vascular leaf	PATO_0000052	shape
12	DEMOTRAIT_0000014	leaf size	PO_0009025	vascular leaf	PATO_0000117	size
13	DEMOTRAIT_0000015	leaf weight	PO_0009025	vascular leaf	PATO_0000128	weight
14	DEMOTRAIT_0000017	flower morphology	PO_0009046	flower	PATO_0000051	morphology
15	DEMOTRAIT_0000018	flower shape	PO_0009046	flower	PATO_0000052	shape
<b>16</b>	DEMOTRAIT_0000019	flower size	PO_0009046	flower	PATO_0000117	size
17	DEMOTRAIT_0000020	flower weight	PO_0009046	flower	PATO_0000128	weight
<b>18</b>	DEMOTRAIT_0000022	perianth morphology	PO_0009058	perianth	PATO_0000051	morphology
19	DEMOTRAIT_0000023	perianth shape	PO_0009058	perianth	PATO_0000052	shape
20	DEMOTRAIT_0000024	perianth size	PO_0009058	perianth	PATO_0000117	size
21	DEMOTRAIT_0000025	perianth weight	PO_0009058	perianth	PATO_0000128	weight
22						
22						

If you have python installed, you can convert this to OWL:

./apply-pattern.py -p simple\_measurable.yaml -i test1.csv > test1.owl

Otherwise, just use the existing "test1.owl" file in the directory.

Open this in Protégé and look under plant trait. You should see a flat list of terms. You can navigate to one and look at the description tab. As you can see, the values from the spreadsheet have been slotted in as specific object property

restrictions:



Next, turn on the reasoner. You will see superclasses show up in yellow. These are inferred superclasses.

Description OBO Graph View Component	
Description: 'vascular leaf shape'	
Equivalent To 🕀 'plant trait' and ('occurs in' some 'vascular leaf') and (affects_quality some shape)	<b>?@</b> ×0
SubClass Of +	
plant structure shape'	?@
😑 'vascular leaf morphology'	?@

Navigate to "Class hierarchy (inferred)"

Active Ontology × Entities × Classes × Object Properties × Annotation Properties	× Ir	ndivio	ł
Class hierarchy Class hierarchy (inferred)			
Class hierarchy (inferred): 'vascular leaf shape'	ШВ		
<ul> <li>Thing</li> <li>'plant anatomical entity'</li> <li>'plant structure development stage'</li> <li>'plant trait'</li> <li>'plant trait'</li> <li>'plant structure morphology'</li> <li>'plant structure morphology'</li> <li>'prianth stape'</li> <li>'perianth shape'</li> <li>'perianth size'</li> <li>'plant structure shape'</li> <li>'plant structure shape'</li> <li>'perianth shape'</li> <li>'perianth shape'</li> <li>'perianth shape'</li> <li>'perianth shape'</li> <li>'perianth shape'</li> <li>'seed shape'</li> <li>'seed shape'</li> <li>'seed morphology'</li> <li>'seed morphology'</li> <li>'seed shape'</li> <li>'seed size'</li> <li>'seed size'</li> <li>'ascular leaf morphology'</li> <li>'yascular leaf shape'</li> <li>'yascular leaf size'</li> <li>'ascular leaf size'</li> </ul>			

You will see the entire polyhierarchy is inferred automatically

### **Object properties**

At this point load: chromosome-parts-interim.owl form the tutorial directory 'basic-restriction'

We will now create an object property and use this to add some restriction onto classes. In OWL properties are used to assert relationships between individuals (or instance). 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.

### Create an object property

We will use the object property view circled below to create a part\_of property. In OWL all properties are a sub property of topObjectProperty.

e o o chromosome-parts (http://purl.obolibrary.org/obo/test/chromosome-parts.owl) - [/U	Jsers/jupp/Documents/teaching/GO_tutorial/tutorial/basic-subclass/chromosome-parts-interim.owl]	
Comparent (http://purl.obolibrary.org/obo/test/chromosome-parts.owl)	¢) 00 (Q.	
Active Ontology Entities Classes Object Properties D	ata Properties Individuals SKOS view DL Query OPPL OPPL Patterns	
Churchiersette Churchiersette Sefarradt Caurch Ameritations		
Class hierarchy: 'chromosomal part'	Class Annotations Class Usage Manchester syntax rendering	0999
Ching     Component     Cell	definition "Any constituent part of a chromosome, a structure composed of a very long molecule of DNA and associated proteins (e.g. histones) that carries hand/lazu information."Astrino	080
v ●'cell part' ● intracellular ▶ ● "intracellular part'	has_exact_synonym " hhs_exact_synonym " hhs_exact_synonym " has_exact_synonyment"	000
▼ ●organelle ▶ ●'intracellular organelle'	has_exact_synonym "chromosome.component"Astring	080
▶ @'non-membrane-bounded organelle' ▼ @'organelle part' ▼ @'intracellular organelle part'	has_exact_synonym *chromsome part^^string	080
Chromosomal part     DNA replication termination region	has_obo_namespace "cellular_component"^^string	080
chromocenter	id "GO:0044427"^^string	080
<ul> <li>Chromosome, celtometer region</li> <li>Chromosome, telometer region</li> </ul>	in_subset ♦ gosubset_prok	080
	label "chromosomal part"^^string	080
	Description: "chromosomal part"	
	"intracellular organelle part'	080
Object property hierarchy Data property hierarchy Individuals by type Annotation property hierarchy Datatypes	Inherited anonymous classes	
Objectshopenty hierarchy:	Members 💿	
-topObjectProperty	Keys 🕥	
	Disjoint classes 🕥	
	Disjoint union of 💿	
	J [	

Select the "add sub property button" circled below and name the property part\_of.

Object property hierarchy Data property hierarchy Individuals by type Annotation property hierarchy Datatype Detatype Datatype Detatype De	xes Lieux
Create a new OWLObjectProperty	
Cancel OK	

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 Protege by simply selecting the transitive check box.

	Characteristics: part □⊟■⊠	Description: part_of DEBO
	Eunctional	Domains (intersection) 🕞
1	Inverse functional	Ranges (intersection) 🕣
(	✓ Transitive	Equivalent object properties 🙃
Ì	Symmetric	
	Asymmetric	Super properties 💮
	Reflexive	Inverse properties 💿
	Irreflexive	Disjoint properties 🕣
		Property chains 💮

To use the reasoner click Reasoner->Start reasoner 🛛 🗹 Show Inferences

### **OWL** class restrictions

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 restriction are further categorised 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 class'". In Protégé 4 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 closure to the existential. That is we can assert that a cellular component is capable of these function, and is only capable of those function 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. Note that in OBO-Format, all relationships are implicitly existentially qualified.

### **Superclass restrictions**

In OBO-Edit you will be familiar with creating relationships between classes. 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.

Select 'organelle part' in the class hierarchy and look at it's current class description. At the top of this view there are two slots for defining equivalent classes and superclasses. 'organelle part' already has one superclass named cellular\_component.

O O chromosome-parts (http://purl.obolibrary.org/obo/test/chromosome-parts.owl) - [/Users	s/jupp/Documents/teaching/GO_tutorial/tutorial/basic-subclass/chromosome-parts-interim.owl]
Chromosome-parts (http://purl.obolibrary.org/obo/test/chromosome-parts.owl)	÷) 88 (Q,
Active Ontology Entitics Classes Object Properties Data	Properties Individuals SKOS view DL Query OPPL OPPL Patterns
Class bierarchy Class bierarchy 0nferred) Search Annotations	Class Annotations Class lisage Marchester suntax rendering
Class hierarchy: 'organelle part'	Annotations: 'organelle part'
	Annotations 🕥
v cellular_component	definition "Any constituent part of an organelle, an organized structure of distinctive morphology and function. Includes constituent parts of the nucleus, mitchondria, plastids, vacuoles, vesicles, ribosomes and the cotoskeleton, but excludes the plasma membrane."Anstring
♥ © cei part ● intracellular ▶ ● 'intracellular part'	has_obo_namespace OCO 'cellular_component'^^string
▼ ©organelle ▶ © intracellular organelle' ▶ © non-membrane-bounded organelle'	Id 'G0:0044422'^^string
▼ <mark>■ organelle part</mark> ▼ ● intracellular organelle part	in_subset ♦ gosubset_prok
ONA replication termination region'     Ochromatin	abel OCO 'organelle part'^^string
	Deventions (organile part) BERS Equilier datus ()
	Superclasses
Older woods Means	Centual_component     Inherited anonymous classes
Object property hierarchy: part_of	Members 🕥
v topObjectProperty	Keys 🜑
	Disjoint classes 🕥
	• organelle 0000
	Disjoint union of 💿

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 superclasses slot. We will define this anonymous superclass in Manchester OWL syntax as

'part of' some 'organelle'.

escription: 'organell	le part'			0800
quivalent classes	000	'organelle part'		-
uperclasses 🔶 Cellular_compon	Class expressio	on editor Data restriction creat	tor Class hierarchy <b>&gt;</b>	@×0
nherited anonymous class Aembers	es	organelle Porganelle part'		
Keys 🕂 Disjoint classes 🕂				
organelle				080
			Cancel OK	

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

uivalent classes +	
inerclasses 👝	
callular component	
rential_component	
part_or some organetie	
herited anonymous classes	
lembers 🕕	
-	
eys 🚯	
isjoint classes 🕀	
organelle	@×
isjoint union of 🕘	

Using Protégé create some of your own part\_of restriction the 'cell part', 'intracellular part' and 'chromosomal part' classes.

[the instructors may elucidate more on the nature of these class restrictions here]

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.

### **Existential tree plugin**

The class hierarchy view in protégé shows subclass/superclass[2] relationships between classes. The default class hierarchy view is restricted to showing strict is-a, or sub/super class relationships. The existential tree is an alternate class hierarchy view that organise classes into hierarchies based on existential restriction. For example, viewing a partonomy along the 'part of' existential restriction.

The existential tree view can be found under Window -> Views -> Existential Tree

ge.osgi.framework.Launcher File Edit View Reasoner Tools Ref	actor SKUSEd	Window Help					
	/purl.obolibrary.o	Views	Annotation property views	I/tutorial/basic-subclass/chromosome	e-parts-interim.owl]		
The sector of th	ome-parts.owl)	Tabs	Class views	Annotations	\$	88 Q	
Activ	e Ontology En	Create new tab Delete custom tabs Import tab Export current tab Store current layout Reset selected tab to default state	Data property views Datatype views Individual views Misc views Object property views Ontology views SKOSEd views	Class hierarchy     Class hierarchy     Class hierarchy (inferred)     Description     Existential tree     Existential tree (reversed)     General class axioms     OW Viz	ng Search Annotations	)	(Hec
Thing     Cellular_component     cell		Increase font size # Decrease font size #		OntoGraf Outline tree			regexp - +
► • 'cell part'		Timestamp log / console	on termination region'	Outline tree (text) Patterns			
organelle intracellular organelle		Look & Feel	•	Superclass hierarchy Superclass hierarchy (inferred)			
Interpretation of the second secon		Refresh User Interface		Usage			
<ul> <li>♥ ●'intracellular organelle part'</li> <li>▶ ●'chromosomal part'</li> </ul>		has_obo	_namespace " <mark>cell</mark> ular_component"^	^string			

Drop the view over the class description view.

	; 8 Q
Active Ontology Entities Classes	Object Properties Data Properties Individuals SKOS view DL Query OPPL OPPL OPPL Patterns
Class hierarchy Class hierarchy (inferred) Search Annotations	Class Annotations Class Usage Manchester syntax rendering Search Accountions
Class hierarchy: 'organelle part' UEBU	Search Annotations:
▼ ●Thing	All annotations cell regexp - +
v ellular_component	results (17)
	"DNA replication termination region"     transport to the second se
	Cedi     label cell <sup>*</sup> A^string     Annotation(-http://purl.obolbrary.org/obo/A0_0000115>*Any constituent part of a cell, the basic structural and functional unit of all organisms:
	Call part'     definition 'Anv constituent nart of a cell. the basic structural and functional unit of all organisms.'AAstring
	Class description Existential tree (revented)
	exacting river an properties) of game part
	• organelle
Object property hierarchy Data property hierarchy Individuals by type Annotation property hierarchy Datatypes	
Object property hierarchy: part_of BB®	
Tere X	
part_of	
	To use the reasoner click Reasoner->Start reasoner 🥑 Show Inferences

Select the part of property to render your partonomy in the existential tree panel.

00

### **EXERCISE:** Basic Restrictions

Stay in the "basic-restriction" folder in the tutorial directory and follow the instructions in the README.txt

The instructors may demonstrate some of the additional options for navigating the ontology at this point:

- Existential Tree Plugin
- OntoGraf

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". Navigate to the DL Query tab.

go (http://purl.obolibrary.org/obo/go.owl) : [/Users/cjm/repos/protege-tutorial/basic-dl-query/cc.owl]				
< > (* go (http://purl.obolibrary.org/obo/go.owl) Search.				
Active Ontology × Entities × Individuals by class × D	DL Query ×			
Class hierarchy: IIIIII	DL Query Snap SPARQL Query			
👫 🕼 🔀 Asserted 🗘	DL query:			
<ul> <li>▼ ● owl:Thing</li> <li>▼ ● cellular_component</li> <li>▶ ● 'axon part'</li> <li>▶ ● 'axoneme part'</li> <li>▶ ● 'bacterial-type flagellum part'</li> <li>● cell</li> <li>▶ ● 'cell cortex part'</li> </ul>	Query (class expression) Execute Add to ontology			
<ul> <li>'cell division site part'</li> <li>'cell junciton'</li> <li>'cell part'</li> <li>'cell septum part'</li> <li>'cell septum part'</li> <li>'cell septum part'</li> <li>'cell septum part'</li> <li>'chloroplast part'</li> <li>'chloroplasmic part'</li> <li>'cytoplasmic part'</li> <li>'cytoplasmic rest'</li> <li>'cytoskeltal part'</li> <li>'cytoskeltal part'</li> <li>'cytoskeltal part'</li> <li>'endosomia part'</li> <li>'endosomia part'</li> <li>'endosomia part'</li> <li>'endosomia part'</li> <li>'ertacellular matrix part'</li> <li>'extracellular matrix part'</li> <li>'extracellular region part'</li> <li>'flagellum part'</li> <li>'flagellum part'</li> <li>'flagellum part'</li> <li>'fost cell part'</li> <li>'instracellular complex'</li> <li>'membrane part'</li> <li>'membrane part'</li> <li>'membrane part'</li> </ul>	Query results	Query for  Direct superclasses  Equivalent classes Direct subclasses Subclasses Instances  Result filters Name contains  Display owl:Thing (in superclass results)  Display owl:Nothing (in subclass results)		

Type "organelle" into the box, and make sure "subclasses" and "descendent classes" are ticked. "subclasses" is the direct subclasses calculated by the reasoner (which may often but not always be the same as the asserted subclasses). "descendants" is all subclasses.

Query:	
Query (class expression)	
organelle	
Execute Add to ontology	
Query results	
Sub classes (5)	Supr
extracellular organelle	🖸 🗌 Anc
'intracellular organelle'	OU 🗖 Equi
'membrane-bounded organelle'	⑦ √ sub
on-membrane-bounded organelle'	0 State
vesicle	1 U Des
December (1999)	
•'acrosomal vesicle'	0
actin cytoskeleton'	0
•'aleurone grain'	0
e'alveolar lamellar body'	0
• 'antipodal cell nucleus'	0
attachment organelle'	0
•'autophagic vacuole'	0
e'azurophil granule'	0
bacterial nucleoid'	0
bacterial-type flagellum'	6

In general for GO you should never need the "individuals" box 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".

Query:	
Query (class expression)	
part_of some membrane	
Execute Add to ontology	
Query results	
Sub classes (1)	Super classes
O'membrane part'	Ancestor classe
	Equivalent class
Descendant casses (s.s.)	✓ Subclasses
I.S-beta-D-glucan synthase complex	Descendant cla
saturboline-rated channel complex'	
e activin receptor complex'	0
e/alpha-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid selective glutamate receptor complex'	0
alpha-beta T cell receptor complex'	0
•'alpha1-beta1 integrin complex'	0
alpha1-beta1 integrin-alpha3(VI) complex	0
e'alpha1-beta1 integrin-tissue transglutaminase complex'	0
alpha1-beta1 integrin-tyrosine-protein phosphatase non-receptor type 2 complex	0
alpha10-beta1 integrin complex	0
alpha11-beta1 integrin complex'	0
alpha11-beta1 integrin-collagen type I complex	0

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'

	organelles things that are part of an endoplasmic reticulum	
ctive Onto	logy Entities Classes Object Properties Data Properties Individuals OWLViz DL Query OntoGraf	
	Query (class expression)	
	organene and endoplasmic reticulum part	
	Execute Add to ontology	
	Query results	
	Sub classes (1)	🗌 Super c
	•'plasmodesmatal endoplasmic reticulum'	Ancest
	Descendant classes (1)	Equival
	•'plasmodesmatal endoplasmic reticulum'	Subclas
		Descen     Individu
		- marvia

Note that we do not need to use the "part" grouping classes in 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 see. We can also ask for superclasses by ticking the boxes above:

ology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query On	ItoGraf	
Ouerv		
Query (class expression)		
organelle and part_of some 'endoplasmic reticulum'		
Execute Add to ontology		
Query results		
Ancestor classes (9)		☑ Super class
⊖'cell part'	0	Ancestor cl
⊜'cytoplasmic part'	0	Equivalent o
erendoplasmic reticulum part'	0	✓ Subclasses
Intracellular organelle part	0	
⊖'intracellular part'	0	
e'organelle part'	0	
ellular_component	0	
erganelle	0	
Thing	0	
Super classes (2)		
endoplasmic reticulum part'	0	
organelle	0	
Sub classes (1)		
plasmodesmatal endoplasmic reticulum'	0	
Descendant classes (1)		
plasmodesmatal endoplasmic reticulum	0	

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

ve Onto	logy Entities Classes Object Properties Data Properties Individuals OWLViz DL Query OntoGraf		
	Duery		
	Query (class expression)		
	nucleus or part of some nucleus		
	(Execute) (Add to onlongy)		
	Query results		
	Sub classes (2)	ñ	Super classes
	⊖'nuclear part'	ØU	Ancestor class
	enucleus	0	Equivalent clas
	Descendant classes (432)		✓ Subclasses
	•'5-lipoxygenase complex'	0	🗹 Descendant cla
	• 'ACF complex'	0	🗌 Individuals
	• 'activator ecdysone receptor complex'	0	
	• activin responsive factor complex'	0	
	Olympical Activity of the second s	0	
	•'alpha DNA polymerase:primase complex'	0	
	anaphase-promoting complex'	0	
	• antipodal cell nucleus'	0	
	AP1 complex'	0	
	• apolipoprotein B mRNA editing enzyme complex'	0	
	ARC complex'	0	
	ASTRA complex'	0	

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 tutorial directory and follow the instructions in the README.txt

## 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.



## 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.



## cell\_part equivalentTo part\_of some cell

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

Description: 'cell part'	08
iquivalent classes 🚯	
cellular_component and (part_of some cell)	08
uperclasses 🕞	
cellular_component	08
part_of some cell	08
nherited anonymous classes	
lembers 💿	
eys 💮	
isjoint classes 🕕	
isjoint union of 💿	

### Automatic classification

### **EXERCISE:** Basic classification

Go to the basic-classification folder and follow the instructions in the README.txt

### **EXERCISE:** Classification using Unions (Optional)

Optional additional exercise if time permits: Go to taxon-union folder This introduces classification using "or" and "not"

### **Futher reading**

owl primer - advanced class relationships

### Introduction to property axioms

#### **Property heirarchy:**

Navigate to 'occurs in' (object property tab) in Protege to see this.

### **Excercise:**

- make a git branch of the go-ontology repo
- open go-edit.obo in Protege

- GO has the class 'regulation of cvt pathway'
- Create a new class, using the same pattern, for 'negative regulation of cvt pathway' (Hint you can use duplicate to do this, editing the newly created class).
- Run the reasoner. Do you understand the inferred classifications? (Use the explanation plugin to check).

### **Property chains:**

Navigate to 'occurs in' (object property tab) in Protege to see this.

### **EXERCISE:** transport & property chains

Go to the transport property chain exercise folder and follow the instructions in the README

## Imports

OWL ontologies may import one or more other OWL ontologies. Ontology imports are managed using the ontology URI (or IRI). Protégé allows you to import ontologies from both the web and your local files system. The imports panel is found in the Active Ontology tab by default.

O O chromosome-parts (http://purl.obolibrary.org/obo/test/chromosome-parts.owl) - [/Users/jupp/Documents/teaching/GO_tutorial/tutorial/basic-subclass/chromosome-parts.owl]			
🗢 🔿 chromosome-parts	Image: A state of the state		
	Addee Chablerow Entities Classes Object Properties Data Properties Undividuals SKOS view DL Overy OPPL OPPL OPPL Patterns		
Out the second states of			
Ontology annotations:		DL metrics Ontology metrics	
has obo format version		Refress: With	
"1.2"^^string			
		ILC abal kau	
	-341		
		Attributive language. This is the base language which allows:     Attributive language. This is the base language which allows:     Attributive language is the base language which allows:	
		<ul> <li>Concept intersection</li> <li>Universal restrictions</li> <li>Limited existential qualification (restrictions that only have fillers of Thing)</li> </ul>	
	Ĵ	$^{arsigma}\mathcal{L}^{-}$ A sub-langauge of AL, which is obtained by disallowing atomic negation	
	5	${}^{-}\mathcal{L}_{o}$ A sub-language of R <sup>2</sup> , which is obtained by disallowing limited existential quantification	
	$\mathcal{L}$	Complex concept negation	
	2	An abbreviation for AL and C with transitive properties	
	7	Cole hierarchy (subproperties - rdfs:subPropertyOf)	
	C	Nominals. (Enumerated classes or object value restrictions - owl:oneOf, owl:hasValue)	
	2	, Inverse properties	
	J	Cardinality restrictions (owl:Cardinality, owl:minCardianlity, owl:maxCardinality)	
	<u> </u>	2 Qualified cardinality restrictions (available in OWL 1.1)	
	Ċ	C Functional properties	
	Ontology imp	Onto Craf Import View Ontology Prefixes General class axioms RDF/XML rendering	
Imported ontologies:			
Direct Imports 🕒			
Indirect Imports			
		Bostoner active II Charu Information	

When a file is imported into an ontology, only the IRI of the imported ontology is stored. Protégé uses the IRI to try and locate the imported ontology the next time you open the file in Protégé. Most OBO ontologies have an IRI that will refer to document via a URL on the web e.g. the Gene Ontology IRI is http://purl.obolibrary.org/obo/go.owl[4]. Navigating to this URL in a web browser will retrieve the latest gene ontology. If your ontology imports an ontology that does not resolve to a web URL, or you are not connected to the internet, then Protégé will prompt you to resolve the ontology IRI to a file on your system.

Some examples of imports can be found in

http://wiki.geneontology.org/index.php/Ontology\_extensions

### **EXERCISE:** response to stimulus

Go to the response-to-stimulus directory and follow the README.txt

This example makes use of a small ontology called STIMO - the stimulus ontology.
000	Import ontology wizard
	Import type
	Please choose an option:
	• Import an ontology contained in a specific file.
	O Import an ontology contained in a document located on the web.
	O Import an ontology that is already loaded in the workspace.
	Import an ontology that is contained in one of the ontology libraries.
	Nhrotodo
	Go Back Continue Cancel

## **Ontology libraries**

Having to wait for Protégé to download ontologies from the web can be rather time consuming, so luckily Protégé has a mechanism for you define ontology libraries that enable you to store mapping between ontology IRIs and files on your filesystem. You can create edit your ontology library in the File > Edit ontology libraries... menu item.

## CHAPTER 14

ege	.osgi.framework.Launcher	File Edit	View	Reasoner	Tools	Refactor	SKOSEd	Window
(	<ul> <li>O</li> <li>O</li></ul>	New Open Open from Open reco	n URL ent		ዕዝ ዝ ዕዝ	€N /purl. €O €O ▶ ve Ont	obolibrary. parts.owl) tology Er	org/obo/tes
e	Ontology annotations: Annotations 🛨 has_obo_format_version	Save Save as Gather or Export in	itologies. ferred axi	 oms as on	ዝ ጉዝ ጉዝ tology	S S G		B DL me rDL Eb
	"1.2"^^string	Edit active Edit onto Loaded o Check for	e ontolog ogy libra ntology s	y library ries ources	H	6L		Symb
	-	Close	F3		Э	ew		$\mathcal{A}$
								$rac{\mathcal{F}}{\mathcal{F}}$

Ontology libraries are stored in a file on your system called catalog-001.xml. When you start working with Protégé you will begin to notice many of these files cropping up on your file system. Chose a catalog-001.xml file to edit and add ontology files to your library.

000	Ontology libraries							
Ontolog	y libraries:							
ď ď	₽ ta							
i Ontolo 📄 Fo	ogy Repository from catalog file /Users/jupp/Documents/teaching/GO_tutorial/tutorial/basic-subclass/catalog-v001.xml Ider Repository for /Users/jupp/Documents/teaching/GO_tutorial/tutorial/basic-subclass (Recursive)							
	Single Ontology Redirect Folder Repository							
	Physical Location: Browse							
	Generate import suggestions							
	Import Declaration:							
	OK Cancel							
	Cancel OK							
Class hierarchy: 'organell	Class hierarchy (Glass hierarchy (efened)   Search Annotations   Class Annotations   Class Linge   Manchester syntax rendering   Search Annotations   Search Annotations : (Search Annotations   Search Annotations   Searc							
Second States (Second States) Thing Cellular_composition	onent esubs(17)							
● cell ▶ ●'cell part' ▶ ● organelle								
► <mark>●</mark> 'organelle p	tabel "cell ^^string ▼ ● Cell part' definition 'Any constituent part of a <u>sell</u> , the basic structural and functional unit of all organisms."^Astring but obe promessare. Cellular composent"/Actring							
	label <b>'cell</b> part <sup>A</sup> string <b>Cellular component</b> comment Note that, in addition to forming the root of the <b>cell</b> ular component ontology, this term is recommended for use for the annotation of ger							
	database.cross_reference "III. Subcellular sano 1337158144"^Astring definition "The part of a cell or its extracellular environment in which a gene product is located. A gene product may be located in one or more part has_exact_synonym "cellular component ^Astring							
	has, narrow, synonym "cellular component unknown"Astring has, obb, namespace "cellular component"Astring label "cellular; component"Astring							
	v vernomatur     database_cross_reference "NIF_Sub <mark>Cell</mark> /ularsao1615953555*^string     has_obo_namespace "cellular_component"^string     v chomogeneter							
	A sobo namespace 'cellular component'^^string     A sobo namespace 'cellular component'^Astring     A sobo namespace 'cellular component'AAstring							

Existential tree: (all properties) 'organelle	Class description Existential tree Existential tree (reversed)	
✓ <u>organelle part</u> ● organelle	Select an object property       Select an object property       TopObjectProperty       part_of	

## **Catalog files**

TODO