**OBOEdit tutorial**

Table of Contents

Table of Contents

Preface 1

Starting OBOEdit 2

The OBOEdit UI 3

Searching in OBOEdit 7

Exercise 1: Searching ontologies 11

Graph visualization in OBOEdit 12

Rendering in OBOEdit 14

Exercise 2: Combining search, rendering, and the Graph Editor and Tree Editors. 16

Creating an ontology in OBOEdit 16

Editing in OBOEdit 21

Saving your ontology in OBOEdit 24

Exercise 3: DIY! 26

# Preface

This tutorial is intended primarily to provide an introduction to some basic ontology principles, as well as to demonstrate some of the advantageous features of OBOEdit. In particular, we focus on the search and visualization capabilities. The tutorial is not intended to provide a complete overview of the OBOEdit tool, for this we recommend reading the help menu available in OBOEdit, which has been contributed to and refined over a number of years by the OBOEdit working group and is quite extensive. Further, questions about OBOEdit should be sent to the working group listserve:

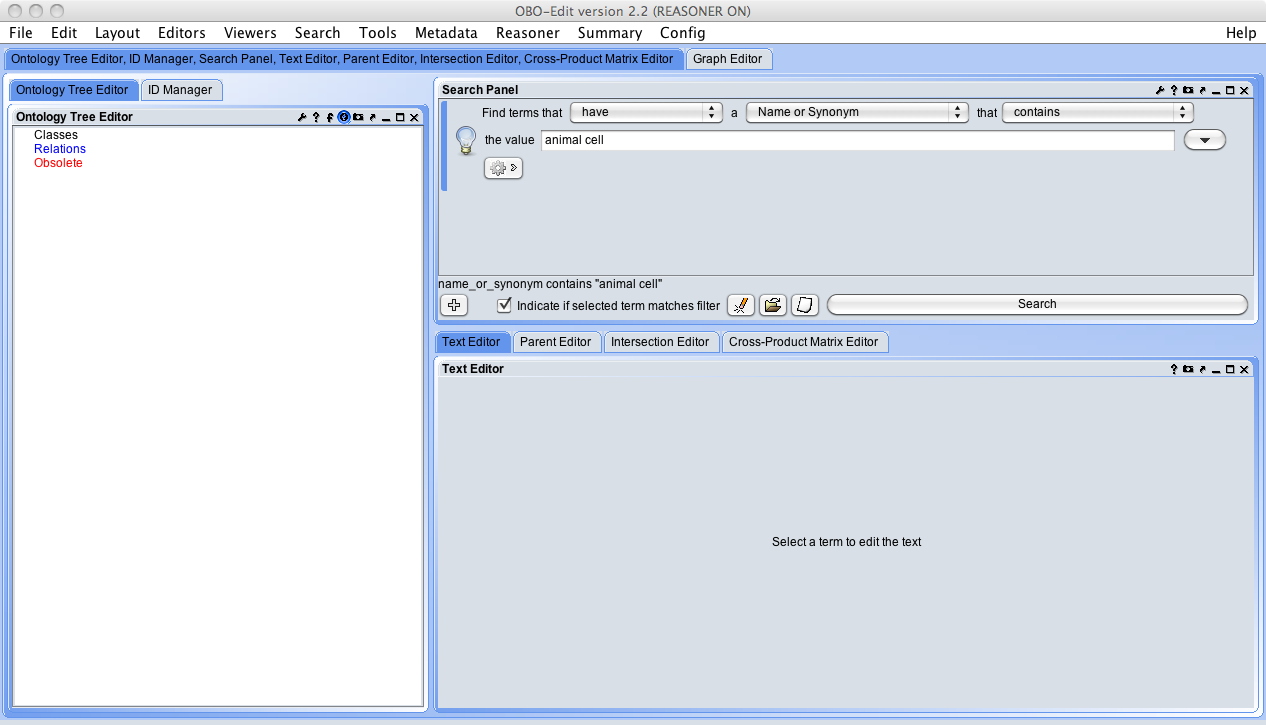
OBO-Edit Working Group [geneontology-oboedit-working-group@lists.sourceforge.net](mailto:geneontology-oboedit-working-group@lists.sourceforge.net)

Bugs should be reported here:

<http://sourceforge.net/tracker/?group_id=36855&atid=418257>

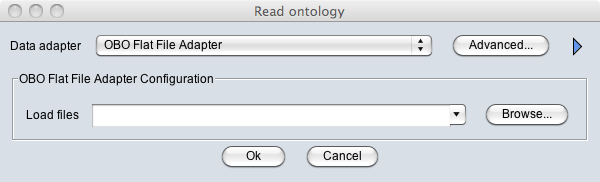
# Starting OBOEdit

When you start OBOEdit you are presented with something that looks like this (note that the layout is very configurable, more about this below):



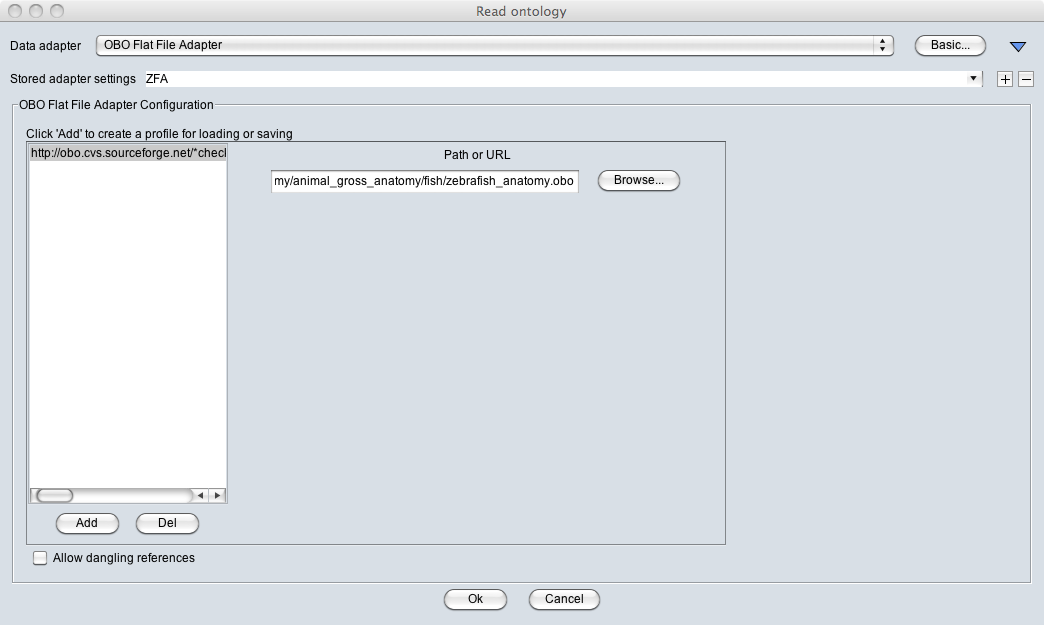
Go to the "File" menu and choose "Load Ontologies..." (shortcut is command-O). You can also make OBOEdit automatically load an OBO-format file on the command line by giving the filename as the command-line argument:  
/Applications/OBO-Edit2/oe-command-line /blahblahblah/foo.obo

A box will appear that looks like this:



From here, you can either navigate to a local file, or paste a URL into the Load Files box. You also want to choose OBO Flat File Adapter for loading an OBO Format ontology.

Alternatively, you can create load profiles by clicking on the advanced button, where you will see the following:



You can create an adapter name, such as ZFA, for the zebrafish anatomy ontology; to do so, click the **+** button next to the Stored Adapter Settings menu and enter the desired name. Click “Add” to start configuring your profile, and then paste the following URL into the Path or URL box:

[http://obo.cvs.sourceforge.net/\*checkout\*/obo/obo/ontology/anatomy/gross\_anatomy/animal\_gross\_anatomy/fish/zebrafish\_anatomy.obo](http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/anatomy/gross_anatomy/animal_gross_anatomy/fish/zebrafish_anatomy.obo)

OBO-Edit can load as many ontology files as you want at one time, just add them to your adapter profile using the Add button below the left panel.

Note that in the advanced load panel, you can select in the lower corner, “allow dangling references”. This is a VERY important button, as it allows one to load an ontology with references to an external ontology. The ZFA doesn’t contain any external references currently, so you don’t need this for now.

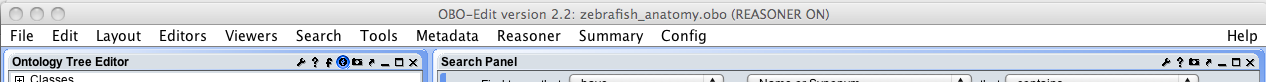
Click “OK”.

You should now see the basic OBOEdit UI.

## The OBOEdit UI

The OBOEdit interface follows a basic paradigm of perspectives and components. The perspective, or arrangement of components is configurable by the user. Each component can be opened to the where the cursor has most recently clicked, and then dragged around to where you like. You can save your favourite perspective in the layout menu, and OBOEdit will remember where you last left things as well.

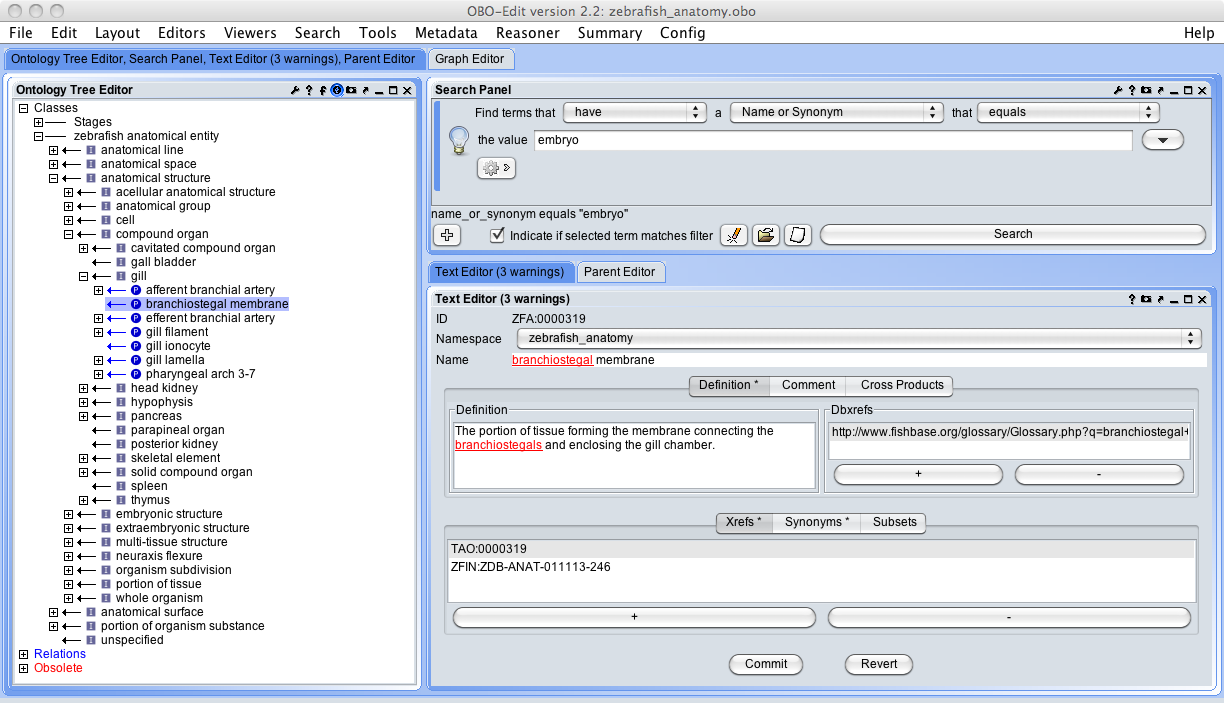
At the very top you see the OBOEdit version number and file name of the ontology you are viewing, and whether the reasoner is on (more about the reasoner later).



A basic configuration usually includes the following components, which we will review.

* Ontology Tree Editor
* Parent Editor
* Text Editor
* Search panel
* Reasoner manager
* Graph Editor

Here is a favorite layout, you may not have this exact layout when you open.

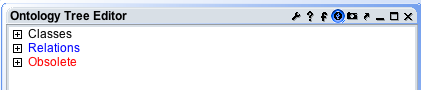


If any of these components is not open, you can go to the appropriate menu item and select it, and then move the panel where you wish. For example, if the Parent Editor isn’t open, go to Editor menu and select Parent Editor.

Each component is configurable and if you hover your mouse over an icon, a phrase will appear that explains what the button does. There is further documentation in the help menu. The exact set of icons depends on the type of component, but it may include:

* A wrench or spanner icon, which means "Configure this component." When you are done setting the configuration, click the checkmark icon.
* A lowercase 'f', which means "Quick filtering options."
* A question mark, which means "Help."
* A tiny globe (looks kind of like an octagon with a lightning bolt in it), which means "Switch to local selection mode". If you're in local selection mode, this icon will appear as a little house, which you can click to switch back to global selection mode”. \* This feature enables all components in global mode to respond to selections and operations on a term, those in local mode will “hold their view” – useful when you want to operate on multiple parts of an ontology at once.
* A little camera, which means "Save an image."
* An arrow that points upwards and to the right, which means "Undock."
* An underscore, which means "Minimize." If you click this icon, the component will disappear and be replaced by a tab at the bottom of the OBO-Edit window.
* A rectangle, which means "Maximize."
* An X, which means "Close."

**The Ontology Tree Editor**



The Ontology Tree Editor is your main view of the classes and properties in your ontology. Whenever OBO-Edit loads an ontology, the Ontology Tree Editor panel always contains three root nodes named "Classes", "Relations", and "Obsolete".

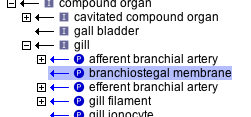
Click on the Classes “+”, what do you see? Do the same with the Relations and the Obsolete entries.

All of the classes (aka "terms") in your ontology are listed under the "Classes" nodes. Relations (aka relationship types) are listed under "Relations", and all obsolete objects (which may be classes, relations, or instances) will be listed under the Obsolete node. These top level nodes **do not represent objects in the ontology**. They exist simply to help organize the display. You cannot select these top-level nodes or modify them in any way.

The Ontology Tree Editor panel displays the ontology as a tree. Branches of the tree can be expanded or collapsed, so that only a part of the tree is visible at any one time. The expand/collapse controls look like boxes containing a plus or minus sign.

If the expand/collapse control is marked with a plus, that means that there are other branches of the tree below, and clicking on the control will cause those branches to become visible. If the control is marked with a minus, it means that child branches are already visible, and clicking on the control will cause them to be collapsed. If a line in the display has no expand/collapse, it means that there are no branches beneath that node.

Each line of the display describes a relationship between a child term and its parents (superclasses). The arrow is there as a reminder that these relationships are to be read from right-to-left (NOTE this is semi-unintuitive, you will sometimes have to remind yourself to read in this direction for non-subclass relations).

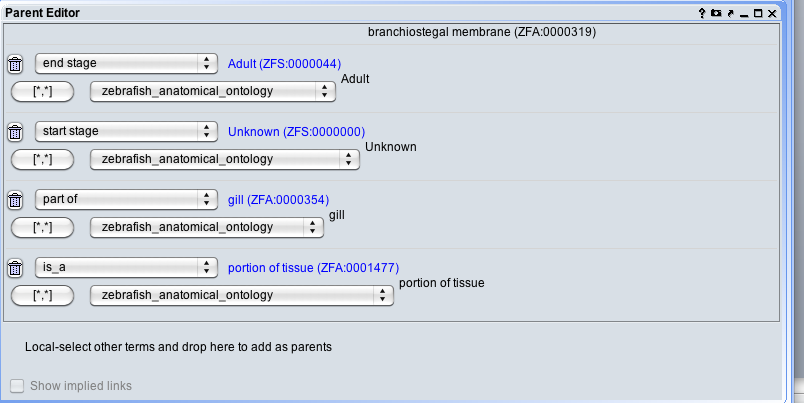


In the above figure, ‘branchiostegal membrane’ is read as being a part\_of the gill (the “P” is the symbol for the part\_of relation in OE). Similarly, the ‘gall bladder’ is\_a ‘compound organ’.

**The Parent Editor**

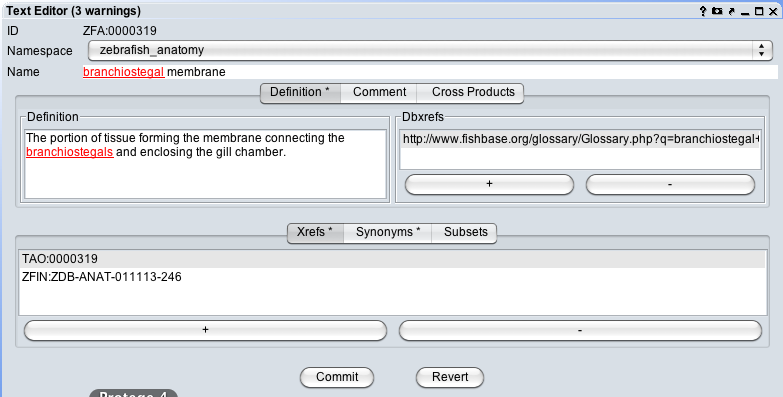
The Parent Editor displays the immediate parents of the currently selected term.

This is a convenient way in which to view all classes that have an asserted superclass or other relation to the selected term. Here you can also delete a relationship, change a relationship type using the menu next to each parent, or click on a parent’s name to go to its information.



**The Text Editor**

The Text Editor panel allows you to edit the attributes of a term, including its label, text definition, comments, synonyms, and dbxrefs (definition references). You cannot edit a term ID – these come from the file that you have opened; new class IDs (described below) are specified in the ID manager configuration. Select a term in the Ontology Tree Editor Panel to load it into the text editor.

****

Note that the red indicates that there are potential errors in spelling, character encoding, logical errors, etc. This error checking is entirely configurable and is a whole topic unto itself that we are not going to go into detail here.

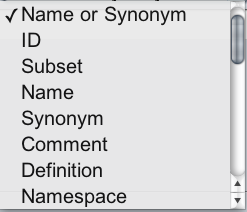
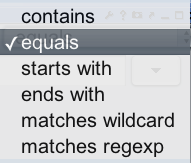
# Searching in OBOEdit

**The Search Panel**

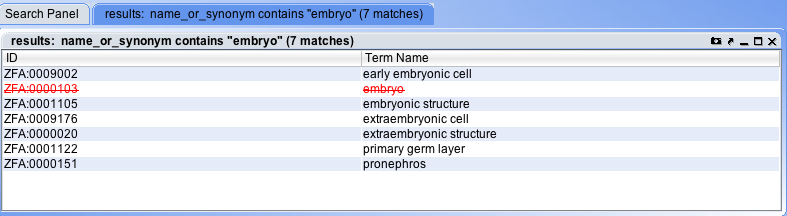
The OBOEdit Search function is really quite nice. It allows a combination of search strategies in an advanced feature. Basically, you can search class annotations as well as perform logical queries.



Each filter dropdown lets you specify the fields, properties or semantics of what you are searching for. Here are some of the options:

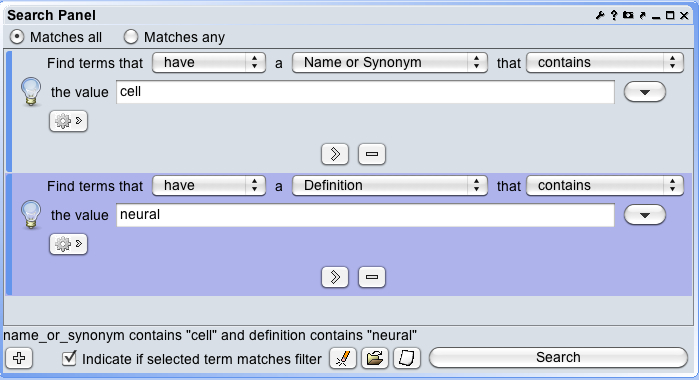
To do a basic search, first use the filter designer to create the filter you want to use. Then click the "Search" button. When the search is complete, a list of results will appear and a new tab will be added to the search interface. Term search results consist of a list of terms that match your search query. Each term's ID and name are listed in another tab, as in this search for “embryo”, using the filters “have”, “name or synonym” and “contains”:



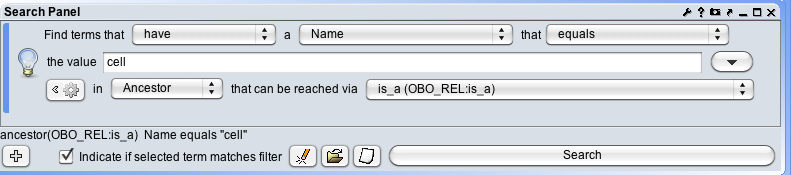
What does the red crossed out term indicate?

To sort the search results by a particular field (e.g., ID), click the appropriate header at the top of the search column. Clicking again will reverse-sort by the same column. Click a search result to select it. The search result will automatically be selected in any panel set to global selection mode.

To do an advanced search, you click the plus in the lower right corner and a second box pops up. In the following search we are searching for classes that contain in their name or synonyms the word “cell” and have a in their definition the string “neural”. Note that the values autocomplete from the appropriate field to guide your search, and that you must choose at the top left whether you wish the search to be for entities that match both queries (“matches all”) or either query (“matches any”).

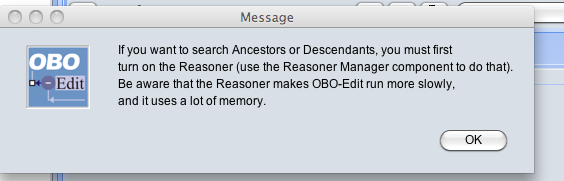


These queries above all used search of annotation properties. Now lets instead perform a semantic query by using the gear icon:



In this example, we are searching for classes that satisfy this query (or filter in OBO Edit lingo). The result will be all classes that are subsumed (follow up the is\_a hierarchy) by the class “cell”:

You will now see this:

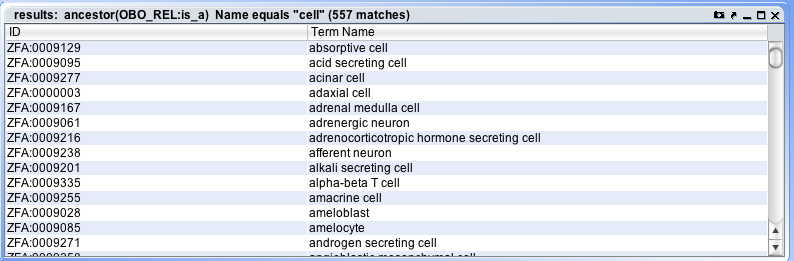


OOPS! This is a semantic query and therefore requires the reasoner to run. Go now to the menu titled “Reasoner” and select “Reasoner Manager”.

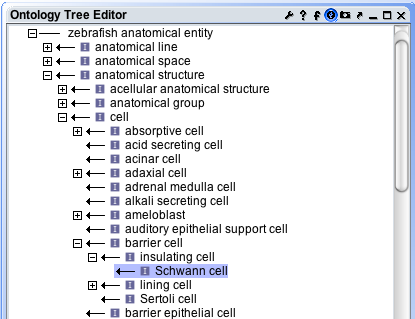


Select the RuleBasedReasoner, and let the reasoner run. This should only take a few seconds with ZFA, but note that some ontologies can run for a very long time. This is why it is sometimes preferable to turn off the reasoner, though for editing purposes it is always nicer to see the effects of your work as you go. NOTE: the OBOEdit reasoner is limited, those available in Protégé are more powerful, we will cover reasoners more in the Protégé tutorial.

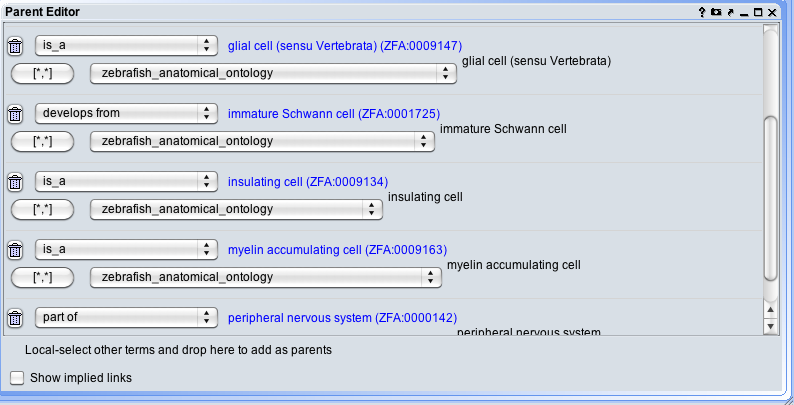
OK, now the query results –- there are 557 cells that have the class “cell” as a superclass:



If you click on ‘Schwann cell’, it will be selected in the Ontology Tree Editor. You can see here how the class satisfied the original query: ‘Schwann cell’ is\_a ‘insulating cell’ is\_a ‘barrier cell’ is\_a ‘cell’. Never mind if you agree with this classification ;-).



Note that the class can occur in numerous places in the Ontology Tree Editor, to see all the terms’ parents, go to the Parent Editor:



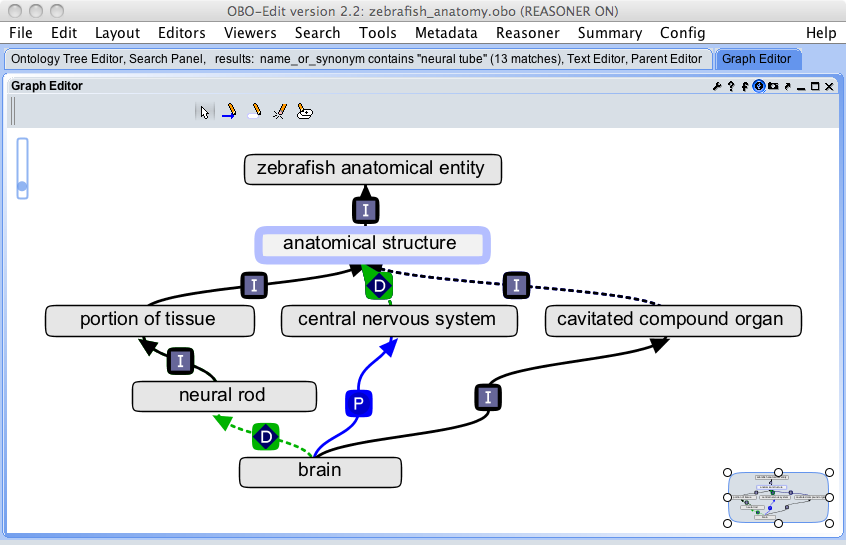
Here you can see multiple asserted superclasses (BAD!) as well as some other relations- this cell is part of the PNS, for example.

## Exercise 1: Searching ontologies

1. Search ZFA for any text field containing “kidney”.
   1. What are the synonyms of “head kidney” and of what type are they?
   2. What are the parents (any type of relation) of “head kidney”?
   3. In which fields does ‘pronephros’ reference the string “kidney”?
2. Search for all the parts of the retina. To specify your search, click the button with the gear icon under the search box.
   1. How many results do you get?
   2. Look at the details of the first result. Is “retina” an immediate parent of this term? If not, why did it appear in the search results?
3. Add to the search criteria for #2 using the plus icon at the lower-left of the Search panel. Find terms that are part of the retina and have “cell” in their name.
4. Find all cells that have an Xref to CL.
5. Bonus Question: Find anatomical structures that start their existence at ‘Pharyngula:Prim-5’ and are present until ‘Adult’ OR have an ‘Unknown’ stage ending? HINT: This is a nested query. How many results do you get?

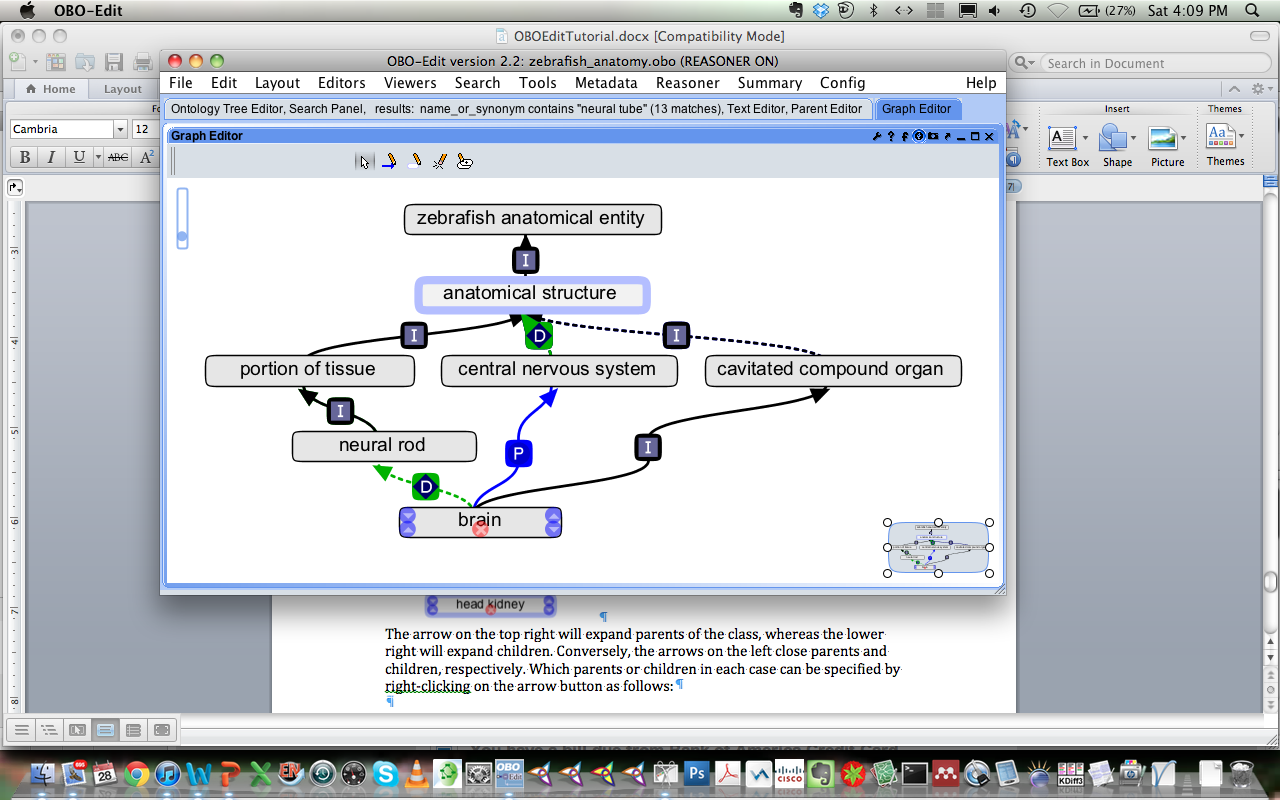
# Graph visualization in OBOEdit

OBOEdit has a Graph Editor tool, which can be opened by going to the Editors menu and clicking Graph Editor. It is recommended to either undock this component by clicking the X button, or have it on a separate tab as it is nice to have some space for the visualization. Here is ‘brain’ shown in the Graph Editor:

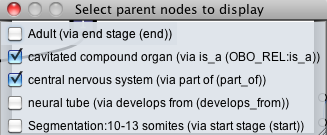


Note a few things here- first the vertical bar on the upper right is a zoom tool. Second, you can click and grab to pan around, with the box on the lower right showing you where you are in the view.

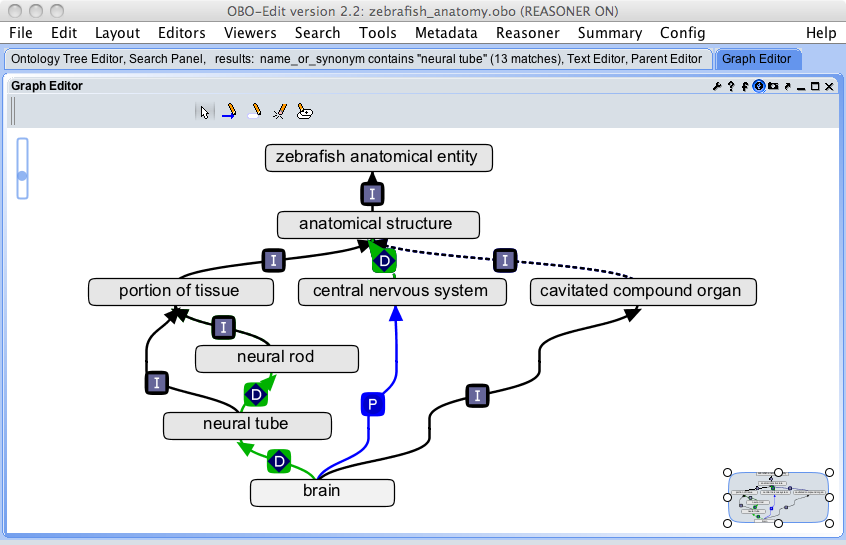
In this example, we see the graphical relations between ‘brain’ and the ‘central nervous system’, of which it is a part, and that the ‘brain’ is a subclass of ‘cavitated compound organ’. These direct relations are shown in solid lines, whereas the inferred relationships are shown in dashed lines (note that the link colors, styles, etc. are configurable, but the default is to show inferred relations in dashed lines). In this case, it is inferred that the ‘brain’ develops\_from some ‘neural rod’. Let’s find out why. Click on the ‘brain’ and you will see some arrows on the right and left sides of the box:



The arrow on the top right will expand parents of the class, whereas the lower right will expand children. Conversely, the arrows on the left close parents and children, respectively. The “x” will remove the class from the display. Which parents or children in each case can be specified by right-clicking on the arrow button as follows:

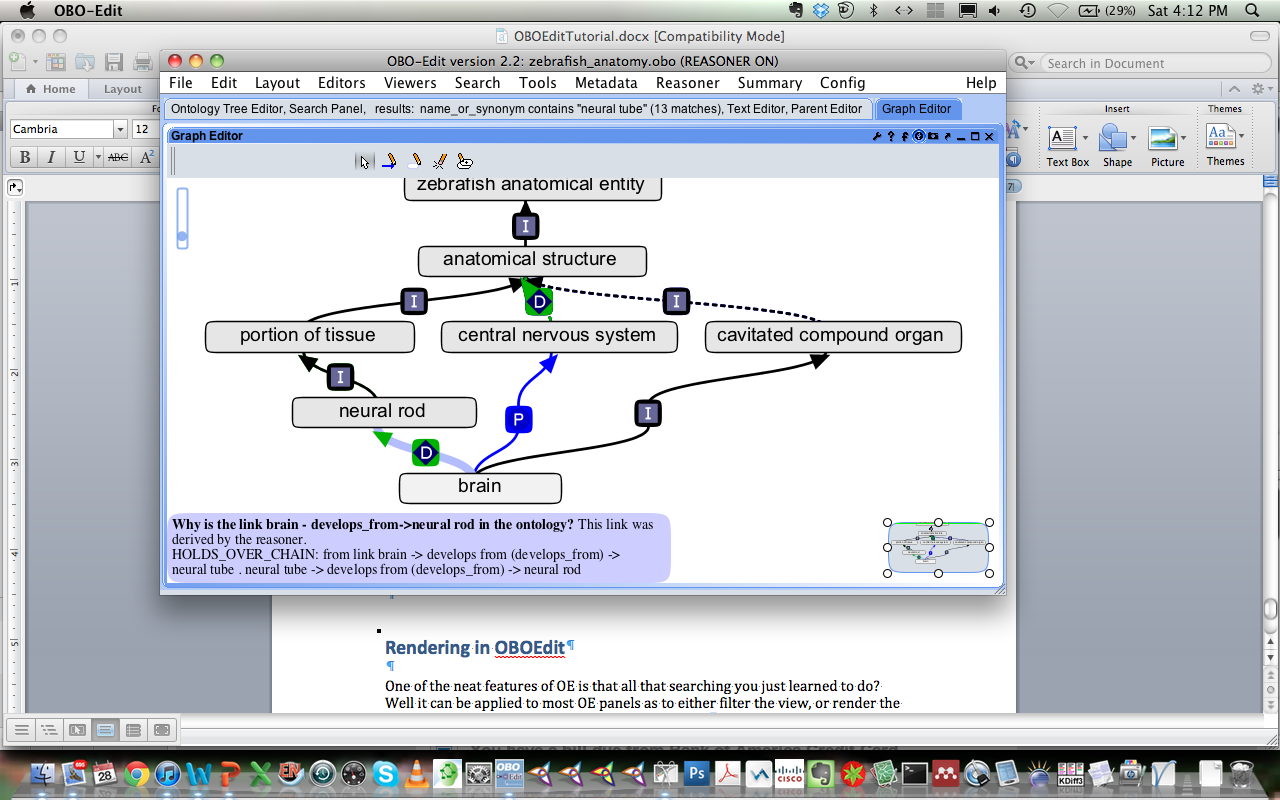


Click on ‘neural tube’ and you now see:



Since develops\_from is transitive, you can see that if ‘brain’ develops\_from ‘neural tube’ and ‘neural tube’ develops\_from ‘neural rod’, it would be inferred that ‘brain’ develops\_from ‘neural rod’.

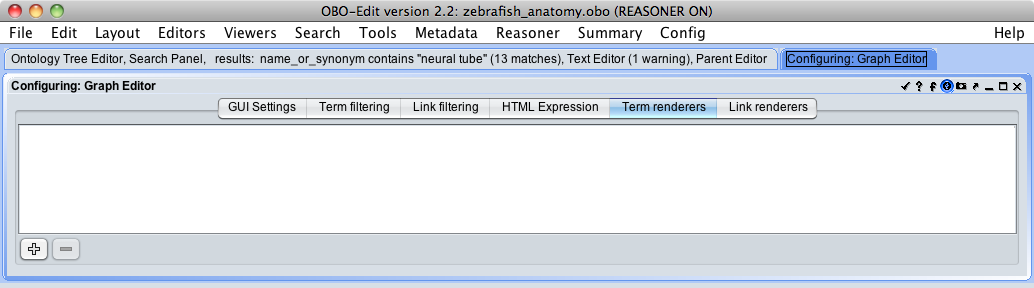
Alternatively, you can click on the inferred link, which will provide information about why the link was inferred:



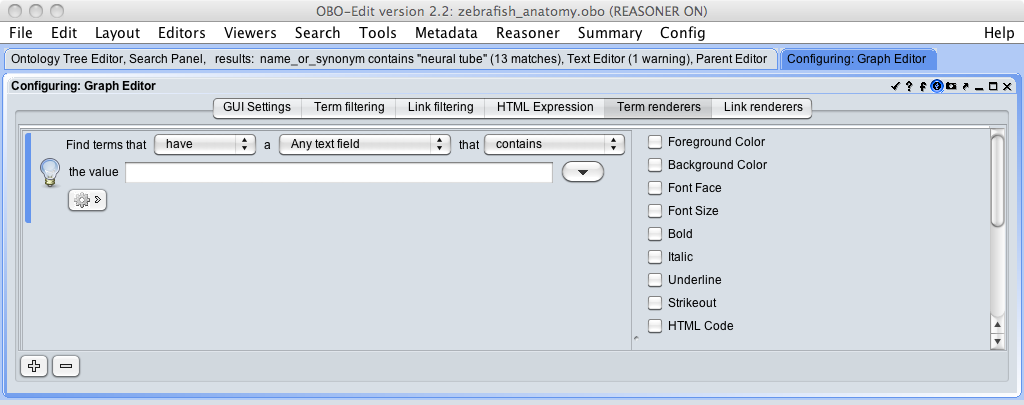
# Rendering in OBOEdit

One of the neat features of OBOEdit is that all that searching you just learned to do? Well, it can be applied to most OE panels as to either filter the view, or render the view. Today we will just review rendering options.

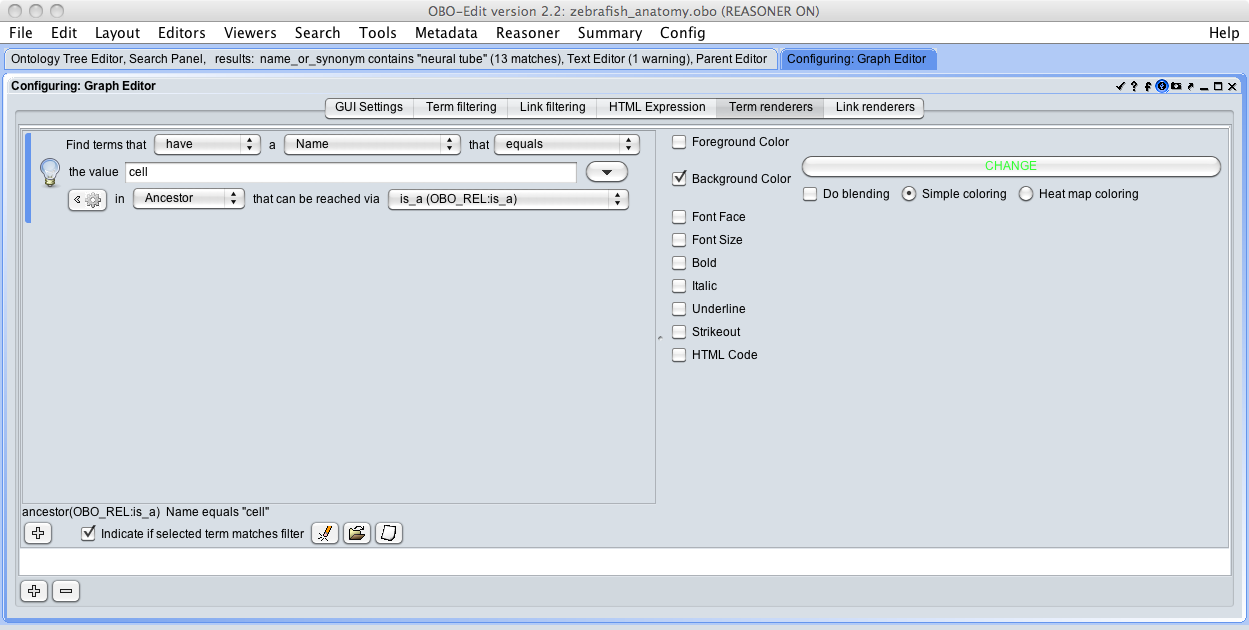
Click on the wrench icon of the Graph Editor, for example:



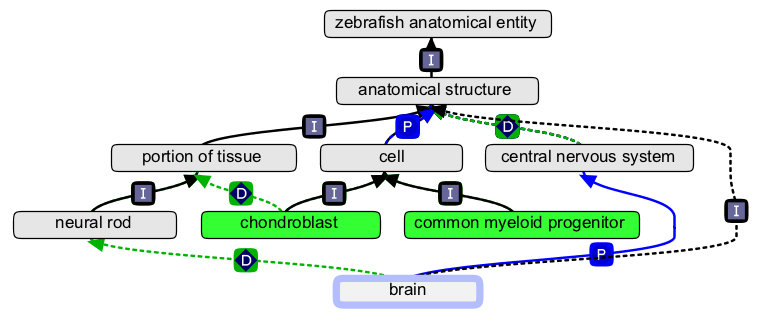
You can see that there are a number of things that can be configured about the Graph Editor, but for today, we will just be interested in the Term rendering tab highlighted above. Choose this, and then click on the plus icon in the lower left corner. You will see the following:



OH LOOK! It’s our friend the friendly Search box, but now with additional rendering features on the right panel. Basically, anything that answers your query above, can be rendered in the display according to a different font style, color, etc.



In the above panel, we have performed the same search as in the search section, for any classes that are subsumed by the class ‘cell’, but here have changed the background color to a nice shade of green. Click enter, and then the check icon in the upper right to see the results.



Note that there exists some default rendering in OBOEdit. In particular, redundant classes (that is, those that are asserted but can be inferred) are marked with a zigzag red arrow, is\_a relations in black, etc. These can all be modified. There are global filters as well as component filters, and commonly used filters can be saved. See the OE User Guide for advanced instructions.

## Exercise 2: Combining search, rendering, and the Graph Editor and Tree Editors.

OBOEdit is a great tool for making figures, as well as to troubleshoot integration issues. The following exercise leverages your advanced search skills as well as your inner artist.

1. In the Graph Editor, color classes that have Xrefs to TAO, CARO, and CL differently. Take a picture showing your colourful distribution of classes using the camera icon in the upper right corner. Include classes with no xrefs, one xref of each, and multiple xrefs. NOTE: There are a lot of Xrefs to TAO because it was cloned from ZFA. What happens when there are Xrefs to more than one ontology?
2. In the Tree Editor, color code classes that develop\_from ‘mesoderm’ and are types of ‘compound organ’.

# Creating an ontology in OBOEdit

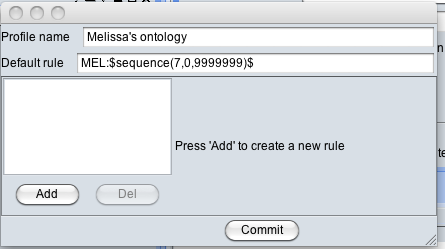
To create a new ontology, instead of loading an existing ontology, you will need to configure OBOEdit to start creating new classes and properties with the specifications that you provide.

To start work on your own ontology (!), go to the File menu and choose “New Ontology”.

Go to the Metadata Menu and choose ID Manager. If you recall from the “Anatomy of an OBO file talk”, OBO IDs are of the format ID:XXXXXXX, where the ID is replaced with a prefix of your choosing (note though, that prior to making your ontology public, you may wish to choose a prefix that has (a) not been previously used in the OBO Foundry, and (b) has been approved by OBO Administrators, and (c) is representative of your content). For now, you can create a prefix with your initials. To do this, press the gear button and you will see:



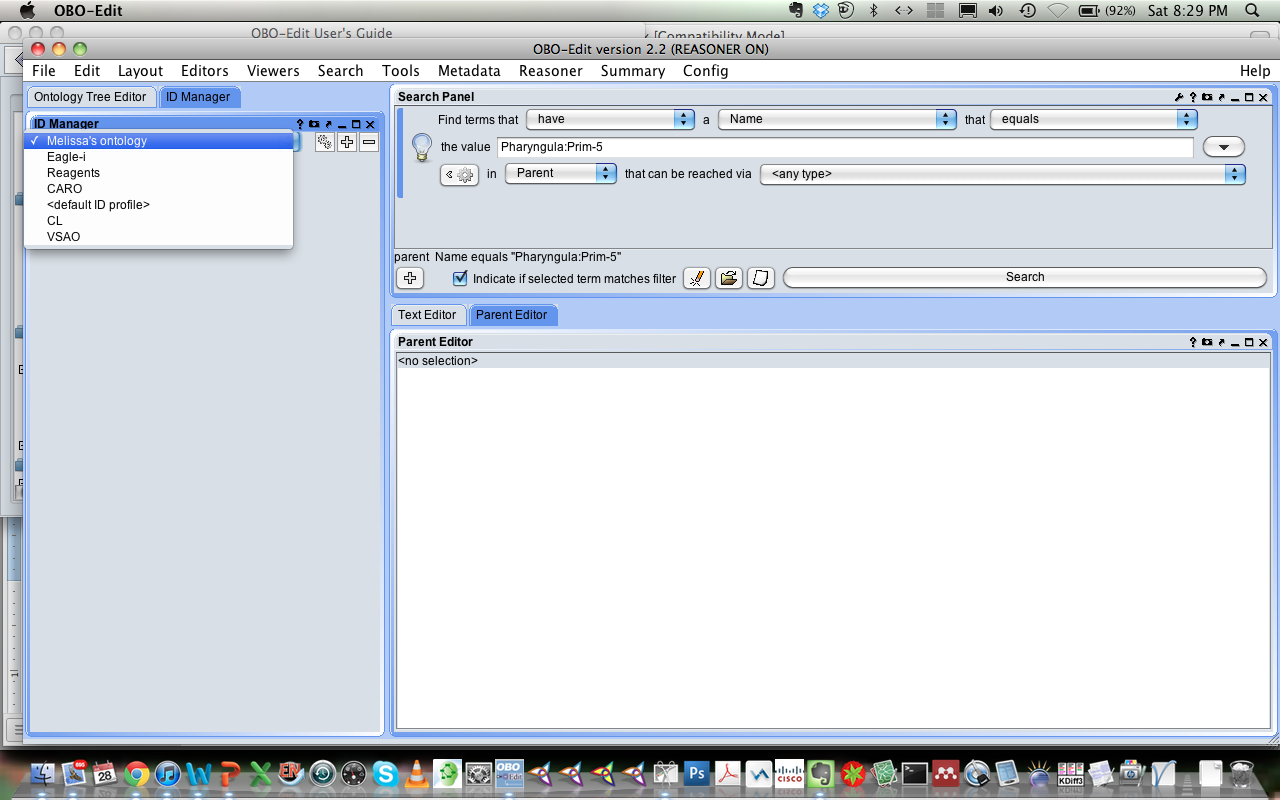
To save your own configuration, change “default” to “your config here”:



The Default rule MEL:$sequence(7,0,9999999)$, means that there is a “MEL” namespace (also called a prefix), and that the IDs generated will be 7 digits long, starting with 0 and ending with 9999999, of the form MEL:0000001. It is frequently helpful to assign different people different ID ranges – to do so, you would simply change the numeric range to instead be a smaller range, such as 0001000-0002000.

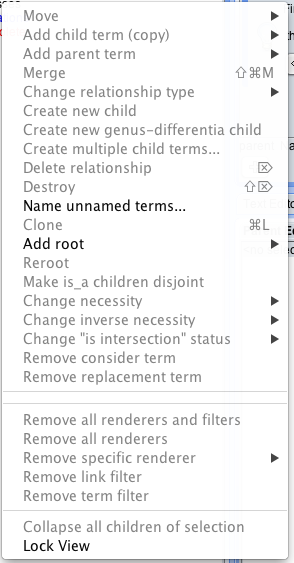
Click the Commit button. (Note: by clicking “Add” you can add rules that are basically search filters to use certain requirements for different ID ranges. We don’t need this for a simple ontology with one namespace).

After you commit, you will see in the pulldown, a list of all ID profiles:

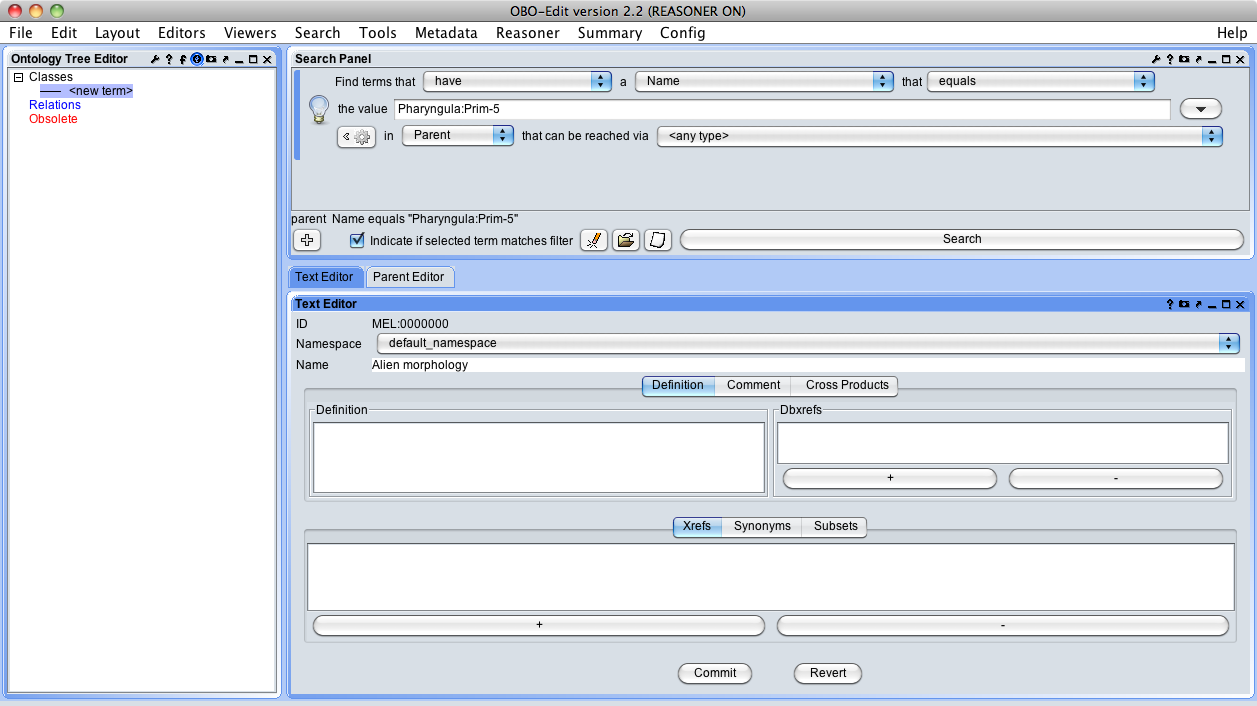


You want to make sure that the one you are using for your ontology is checked. You can then close the ID Manager.

Lets go to the creation of our first class. Right click on “Classes” in the Ontology Tree Editor. You will see the following menu:



In order to create your first class, you must first create the root class – the class that will subsume all other classes.



Note that the first (root) class has ID: MEL:0000000. I have labelled it in the Text Editor, and once I press commit, “alien morphology” will officially be the root of my ontology.

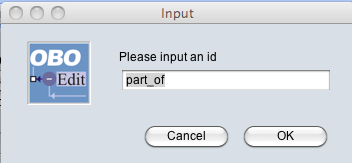
Now that we have our first class, we will need some relations. OBOEdit does provide a small number of pre-defined relations that are present in all ontologies and cannot be modified: **is\_a**, **disjoint\_from**, **inverse\_of**, and **union\_of**.

The built-in relations are defined as follows:

* **is\_a** - The subtyping relation. This relation indicates that one entity is a subtype of another. If the entities are relations, **is\_a** denotes that one relation is a sub-relation of the other. If the entities are classes, **is\_a** denotes subclassing. **is\_a** may relate two classes or two relations, but cannot relate a class to a relation (or vice versa). **is\_a** is similar to the OWL relations owl:subClassOf and owl:subPropertyOf.
* **disjoint\_from** – This symmetric relation indicates that two classes are disjoint. If two classes A and B are marked disjoint, no subclass of A may be a subclass of B, and no subclass of B may be a subclass of A. **disjoint\_from** is similar to the owl relation owl:disjointWith.
* **union\_of** - This relation indicates that a class is defined as a union of other classes. The class apples\_or\_oranges is a **union\_of** the class apples and the class oranges. **union\_of** can only be used to relate classes to other classes; **union\_of** cannot be allowed to relate relations. Use of this relation usually indicates poor ontology design, but it is included here for compatibility with OWL.
* **inverse\_of** – This symmetric relation indicates that a relation is the inverse of another relation (**part\_of** is the **inverse\_of** **has\_part, is\_left\_of** is the **inverse of** **is\_right\_of**). This relation is similar to the OWL relation owl:inverseOf.

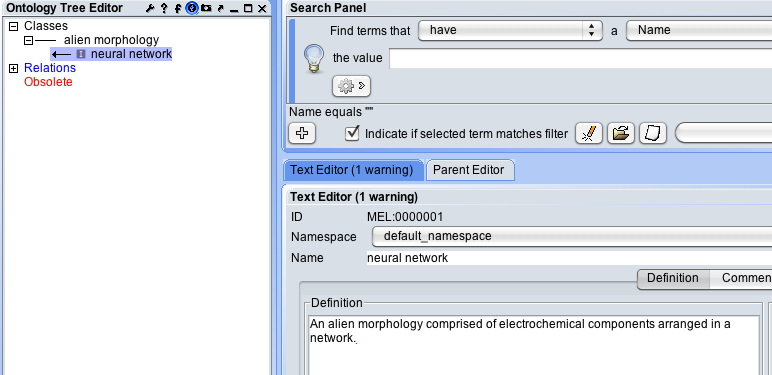
However, in order to get started, you will need to add some relations in addition to these, in particular, part\_of, develops\_from, and any other that you’ve determined to be relevant in your modelling exercise.

To add a new relation, right click on the blue relations and choose “Add root” and then choose “Add Root Relation”. OBOEdit will ask you for an ID. OE handles IDs for properties slightly differently than classes, and expects a label for the ID (this is now considered less than ideal practice). Add your new relation:

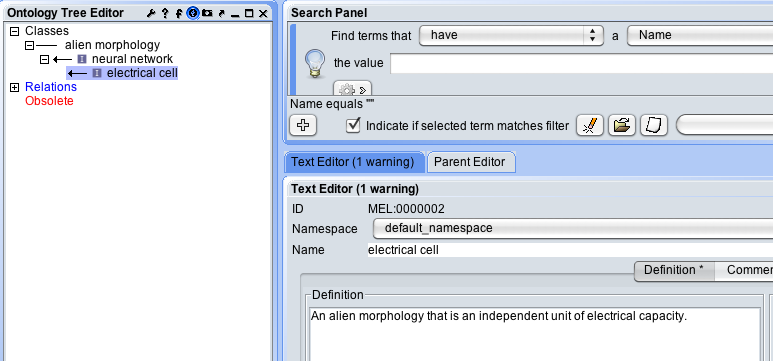


# Editing in OBOEdit

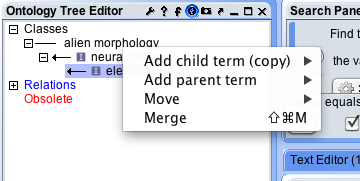
OBOEdit works on a drag and drop system for relating classes to another, however you can only create new classes or properties by right-clicking on a parent class or property. Choose create new child, label it, define it and click commit at the bottom:



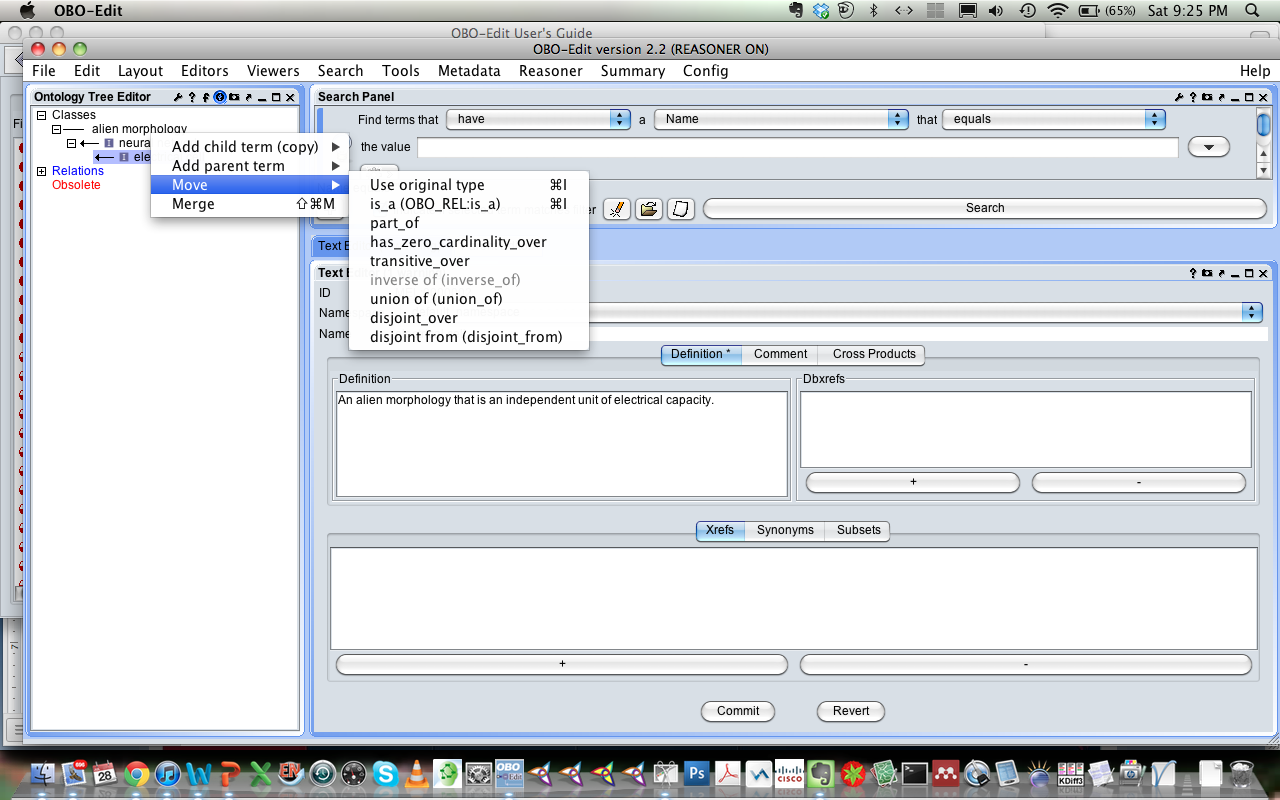
Keep going adding classes:



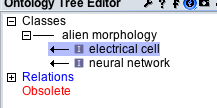
Oops! In this case, the class was meant to be a part of the neural network, not a subclass. I can first move the class to its proper superclass parent by drag and dropping it on “alien morphology”:



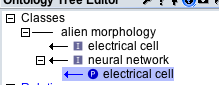
Here OE asks, do you want to move it? Or add add it as a child? This latter feature is a bit confusing, because what it is asking is do you want to move it vs. ADD a relationship to another term. In this case, we first want to move the class using the is\_a relation:



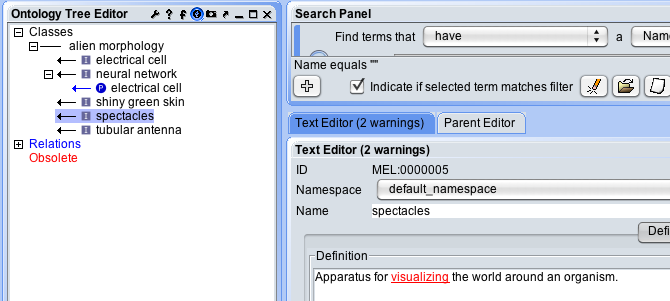
Now we have the electrical cell in its correct location as a subclass of ‘alien morphology’:



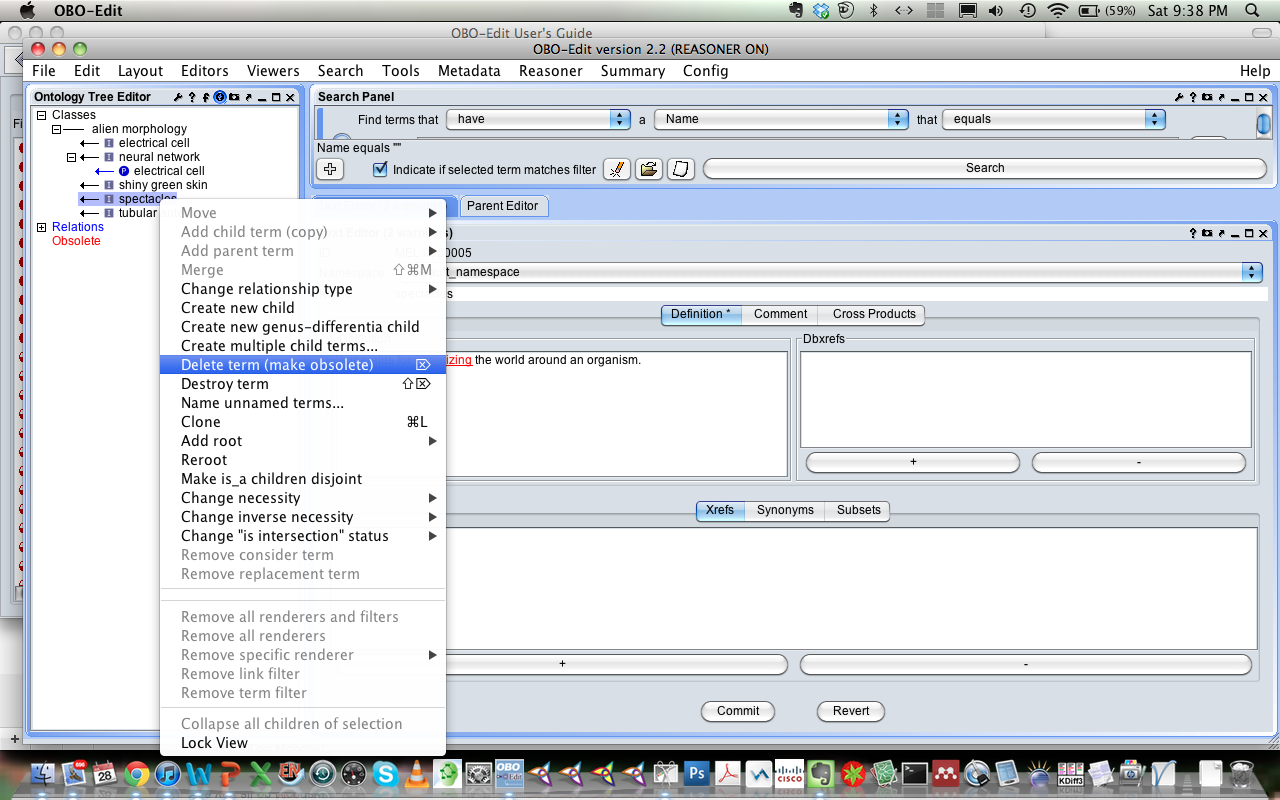
and can work on adding the part relation between the ‘electrical cell’ class and the ‘neural network’ class. Drag and drop again, but this time, instead of choosing “move”, choose “Add child term (copy)”. OE will then ask you which relation you would like, choose part\_of.



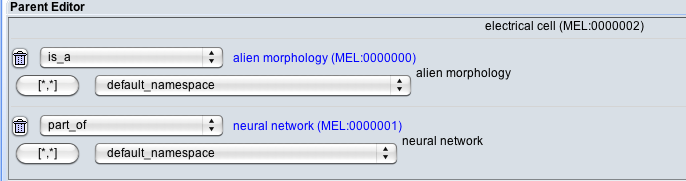
Of course, sometimes you want to remove a relationship, or a class altogether. To do this, use the same right click menu. For example, I determine that “spectacles” really don’t belong in an ontology of alien morphology:



So I obsolete the class by choosing “delete term (obsolete)”.



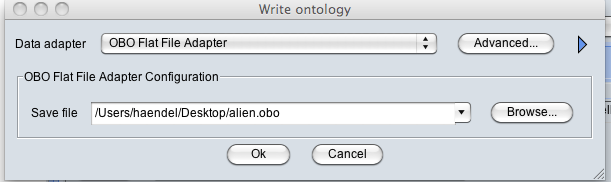
After you do this, you will see it in the red Obsolete classes section. It is STRONGLY encouraged to use the obsolete feature rather than the “Destroy term” feature, which will permanently remove the class and ID from the file. The Obsolete function merely makes the class or property obsolete, retaining its ID so that it cannot be reused. Destroyed IDs WILL BE REUSED. That said, while you are inventing and not sharing with anyone, it is ok to destroy terms in the very early stages, but as soon as you make a file available publicly, you need to be very careful to maintain IDs. Note also, that you will need to remove all parents from a term before obsoleting it. This can easily be done from the parent editor using the trash can button on the left.



# Saving your ontology in OBOEdit

Ok, now that you have created a monster, lets save it and take a look.

The OBOEdit save menu is basically the same as the load menu. Click the File menu and then “Save As”. Here you will want to create a new profile to tell OE where to save your file. Click the Browse button on the right and name your file with a .obo extension.



The advanced save feature allows you to construct filtered saves, saves with dangling references to external ontologies, and the inferred hierarchy. We don’t need these for now.

Once your file is saved, open it in a text editor. Here is alien.obo:

format-version: 1.2

date: 28:07:2012 21:52

saved-by: haendel

auto-generated-by: OBO-Edit 2.2

default-namespace: default\_namespace

[Term]

id: MEL:0000000

name: alien morphology

created\_by: haendel

creation\_date: 2012-07-28T21:01:00Z

[Term]

id: MEL:0000001

name: neural network

def: "An alien morphology comprised of electrochemical components arranged in a network." []

is\_a: MEL:0000000 ! alien morphology

created\_by: haendel

creation\_date: 2012-07-28T21:10:40Z

[Term]

id: MEL:0000002

name: electrical cell

def: "An alien morphology that is an independent unit of electrical capacity." []

is\_a: MEL:0000000 ! alien morphology

relationship: part\_of MEL:0000001 ! neural network

created\_by: haendel

creation\_date: 2012-07-28T21:19:31Z

[Term]

id: MEL:0000003

name: tubular antenna

def: "Alien morphology that consists of two tube projections from the head subdivision of the body." []

is\_a: MEL:0000000 ! alien morphology

created\_by: haendel

creation\_date: 2012-07-28T21:32:19Z

[Term]

id: MEL:0000004

name: shiny green skin

def: "Alien morphology that consists of a shiny green covering of a the organism, or some parts of the organism." []

is\_a: MEL:0000000 ! alien morphology

created\_by: haendel

creation\_date: 2012-07-28T21:33:29Z

[Term]

id: MEL:0000005

name: spectacles

def: "Apparatus for visualizing the world around an organism." []

is\_obsolete: true

created\_by: haendel

creation\_date: 2012-07-28T21:34:48Z

[Typedef]

id: part\_of

name: part\_of

created\_by: haendel

creation\_date: 2012-07-28T21:04:03Z

Some things to note:

* The header saves the name of the user who saved the file, and the date.
* The IDs are of the form MEL:XXXXXXX
* Each class has a creation date and person.
* There is a typedef, or relation, which has been given the same ID as the label, in order to get the icons to show properly in OE.
* The “spectacles” class is marked as is\_obsolete: true

## Exercise 3: DIY!