SWRL Lab

Today's lab aims to be kind of fun, while illustrating to you what SWRL can do (and some of it's limitations).

1. Boot up Protege. Look under **Window → Tabs** and then select **DLQuery** and **SWRLTab** (if you haven't already). You should also have **Entities**, **Object Properties**, and **Classes** Selected.
2. Load **Fam.owl** from QMplus.
3. Let the fun begin!

Part 1

What you see before you is a mock-family with various classes, individuals, properties, and SWRL rules defined. Here you'll insert classes, individuals, and properties of your own.

1. Let's get you started. Define a new class, **Stepbrother**. How would you do that, and where do you place the class within the class hierarchy?
2. Populate the class with some individuals of your choice (name them as you see fit). How would you do that?
3. Now we want to add some object properties to describe **Stepbrother**. How would you do that? What will the domain and range of these properties be? How will you relate them to the est of the ontology? Be creative!
4. Take a stab at adding some data properties. (Examples: **Age**, **Gender**, **Height**, . . .)

Part 2

Now we have a look at SWRL. As you know from the lecture, SWRL extends OWL to be a more robust and expressive language via rules of a certain constrained form. The idea is to chain certain forms of "if-then" rules together, known as "Horn Clauses."

The idea is to read these as universal statements with universal quantifier outside the brackets of an implication, giving "wide scope"; for example:

$\forall x\ (Man(x) \rightarrow Mortal(x))$

. . .Is a Horn clause.

In Protege, we pretend that the statement is prefixed with a universal quantifier, so this is written:

```
Man(?x) -> Mortal(?x)
```

Now obviously one will want to expand these into more complex rules. One common way to do this:

$\forall x(M(x) \wedge N(x) \wedge N`(x) \wedge \ldots \wedge R(x) \rightarrow S(x))$

And this is quite natural in Protege.

1. Go to the **SWRL** tab in Protege. In the top pane you will have a preset list of rules. Ignore them for now. We are going to add our own rules. Click **New** : in the pop-up window, name your new rule, “Stepbrother Rule.” Now add the following:

   ```
   Stepbrother(?x) -> Relative(?x)
   ```

   What would you expect the result of this to be?

2. Now click on the **Control** tab then press the buttons: **[OWL + SWRL → DROOLS]**, then **[Run Drools]**, and then **[Drools →OWL]**.

3. Click on the **Inferred Axioms** tab. What do you see? In light of our insertion of the Stepbrother Rule, did you see what you expected to?

4. Now that we have an idea what this thing does, let’s experiment. Add some new rules, and perhaps individuals, classes and properties of Stepbrothers. Be creative! Try to add rules expanding the left side of the implication.

5. Now repeat Part 1, but with new classes, properties, etc. of your choice.

Part 3

Now we show some things SWRL doesn’t like or won’t do.

1. Under the SWRL Tab, click New and try to add the following rule (using a name of your choice):

   ```
   Stepbrother(?x) ^ not Neighbour(?x) -> moral_reprobate(?x)
   ```

   What do you see? Why?

2. Try the following, making sure you’ve added age as a data property first for the parents.

   ```
   Parent(?x) ^ Age(?x,?age) ^ swrlb:add(?newage, ?age,1) ->
   Age(?x, ?newage)
   ```

   What do you see? Why?

3. Now add:

   ```
   Parent(?x) ^ HasChild(?x, ?y) ^ HasChild(?x, ?z) ^
   differentFrom(?y, ?z) -> Is_Relative(?y, ?z)
   ```

   What do you see? Now what happens when you remove `differentFrom(?y, ?z)` above?