SWRL Practical Exercise

Download the ontology files (EntityServiceAssociationModel.owl, LocationModel.owl, qu-rec20.owl and qu.owl files) from the Github folder and put them in the same folder in your machine.

Load the entity-service association ontology (EntityServiceAssociationModel.owl) into Protégé (the other ontologies will be loaded into the workspace as imports).

The goals for this practical exercise are the following:

Goal 1:

Copy and run your first SWRL rule for entity-sensing service association as discussed in the lecture slides.

Follow the step-by-step instructions as given in **Exercise 1: Write Your First SWRL Rule** and Exercise 2: Run a SQWRL Rule.

Goal 2:

Create and run a rule for actuating service – entity association:

- complete the domain attribute specification for the Orange_Crate entity in EntityServiceAssociationModel.owl so that the Price Actuator service can be associated with it.
- Create the corresponding SWRL rule associating the ESL_DA00018252B_priceactuator_PriceText_service to the Orange_Crate entity.
- Select all associated services and entities in the SWRL rule head to check that the price_actuator service and the Orange_crate entity are correctly associated.

Goal 3:

SWRL rule 2 for actuating service – entity association:

- Create an entity with the correct location and domain attribute specification so that the E4_PhilipsHue_ceiling_lamp_actuatorservice can be associated with it.
- Select all associated services and entities in the SWRL rule head to check the results.

Goal 4:

SWRL rule for 'nearby' association:

- implement a 'nearby' association where the service's service area is adjacent to, or gives access to (i.e. inspect the indoor location model instances for asserted 'givesAcessTo' properties) the entity's current location, as follows:
    a. Create the corresponding entity with the correct location and domain attribute specification
    b. Create the corresponding service with the correct service area and input (for actuating service) or output (for sensing service) specification.
    c. Create the relevant SWRL rule associating the service to the entity and check your results in the SQWRL tab
- Hint: you can create, for instance, a noise-level detector service (i.e. a sensing service with hasOutput property of 'soundIntensity) that can be in a 'corridoor'. The entity can be a room. So, if according to the asserted properties in the location model, the corridor 'gives access to' the room, the entity and service can be associated as being 'nearby'.

Exercise 1: Write Your First SWRL Rule

1. Copy the following SWRL rule in the SWRL tab and run the Pellet reasoner (refer to the detailed steps for SWRL rule editing given below):

```
Service(?s) ^ hasOutput(?s, ?out) ^
Entity(?et) ^ hasA(?et, ?da) ^ hasAttributeType(?da, ?out) ^
hasServiceArea(?s, ?sa) ^
Entity(?et) ^ hasA(?et, ?l) ^ hasLocalLocation(?l, ?sa) ->
sqwrl:select(?s, ?et)
```

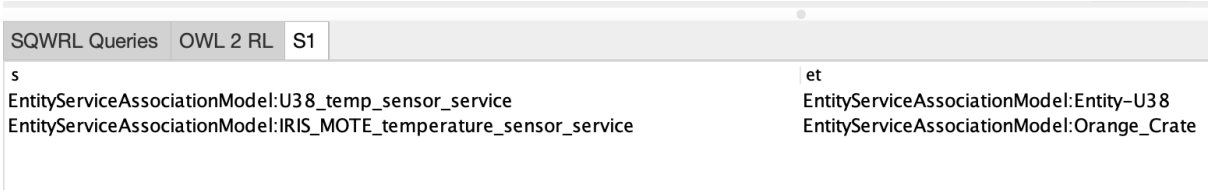
2. To begin with, navigate to or create the SWRLTab. If it doesn't already exist use Window>Tabs>SWRLTab to create and select it. If you don't have the SWRLTab under the Window>Tabs menu then use File>Check for plugins and select the SWRLTab plugin. Remember if you do this you need to restart Protégé for the plugin to be available.
3. The SWRLTab is divided into two main views and then some buttons on the bottom of the tab that relate to DROOLS. In this exercise, we are not going to use it. We will be using the Pellet reasoner as it supports SWRL and when you run the reasoner it will also automatically run any SWRL rules you have.
4. Check to see if the Pellet reasoner is installed. Click on the Reasoner menu. At the bottom of the menu there will be a list of the installed reasoners such as Hermit and possibly Pellet. If Pellet is visible in that menu.
    a. If Pellet is not visible then do File>Check for plugins and select Pellet from the list of available plugins and then select Install. This will install Pellet and you should get a message that says it will take effect the next time you start Protégé. Do a File>Save to save your work then quit Protégé and restart it.
    b. Then go to File>Open recent. You should see your saved ontology in the list of recent ontologies. Select it to load it. Now you should see Pellet under the Reasoner menu and be able to select it so do so.
5. Click on the New button at the bottom of the top view. The other buttons should be grayed out since they only apply if you have at least one rule written. This will give you a new pop-up window to write your rule. In the Name field at the top call the rule: S1. You can skip the comment.
6. Now go to the bottom part of the rule window and start writing the rule. To start you want to bind a parameter to each instance of the Service class. To do this all you need to do is to write: Service(?s). Note that auto-complete should work in this window but sometimes it may not and you may need to type the complete name. Also, you will see various hints or error messages in the Status field as you type which you can mostly ignore for now. E.g., as you type out Service you will see messages like: Invalid SWRL atom predicate 'Ser' until you complete the name of the Service class. Those messages can help you understand why your rule won't parse as you develop more rules but for now you should be able to ignore them.
7. Now we want to test the object property hasOutput. The first parameter will also be `?s`. i.e., we are iterating through each instance of `Service`, binding it to `?s` and then testing the values of these properties. So, in this case, we bind the hasOutput property of a Service to a parameter ?out and reference it in the expression resulting in: ^ hasOutput(?s, ?out).
8. We build the rest of the rule, binding parameters to relevant OWL class and property objects.
9. Finally, we write the consequent of the rule, the part after the arrow that signifies what to do each time the rule succeeds. So, we write -> to signal the beginning of the consequent.
10. In this simple example, we simply want to display in a table the relevant Service and Entity instances that meet the rule criteria for entity-service association. For this, we use the

SQWRL built-in sqwrl:select and pass it the corresponding service and entity parameters ?s and ?e

11. Whereas the expressions on the left-hand side are tests to see if the rule should fire, the expression on the right is an assertion.
12. Thus the whole rule should look like as shown above. Take note that the OK button at the bottom is only possible to select when the rule has a valid syntax. It should be selectable now so select it. You should see the new rule show up at the top of the top-most view.
13. Note that there is a minor bug in SWRL where sometimes the prefix for the current ontology will be added to all the expressions without a prefix. So at some point you may see that your expressions end up looking like this: EntityServiceAssociationModel:Service(?s). If this happens don't worry it won't affect the way the rule works at all.

Exercise 2: Run a SQWRL Rule

1. Now we want to run our rules. Remember there is no need to use those DROOLS buttons. Just synchronize the reasoner and your rules should fire just as other DL axioms that assert values based on inverses, defined classes, etc.
2. To view the result of the SQWRL query, we need to bring up the SQWRLTab if it doesn't already exist using Windows>Tabs>SQWRLTab. You will see it looks almost identical to the SWRLTab.
3. Synchronize the reasoner.
4. Select S1 then select the Run button at the bottom of the tab. This will create a new tab in the lower view called S1. You should see the message at the bottom of the SQWRL Queries tab that the query took 1212 milliseconds. 2 rows were returned, as below:



Exercise 3: Write Another SWRL Rule

1. Make sure you are still in the SWRLTab. Click on the S1 rule and select Clone.
2. This should bring up the same window you used to create your first rule with the code for that rule in the window. Change the name of this rule from S1 to S2.
3. Next edit the test for the domain attribute specification or for new instance creation as specified in the Goal 2-4 requirements.