Exercise sheet: Knowrob Tutorial

6. November 2010

1 Querying the KnowRob ontology

Launch the system using the rosprolog script which takes the name of the package to be launched as parameter. When calling a package with rosprolog, the init.pl file is loaded, and all init.pl of referenced packages as well. That procedure ensures that all packages are initialized when being loaded.

```
rosrun rosprolog rosprolog knowrob_tutorial
```

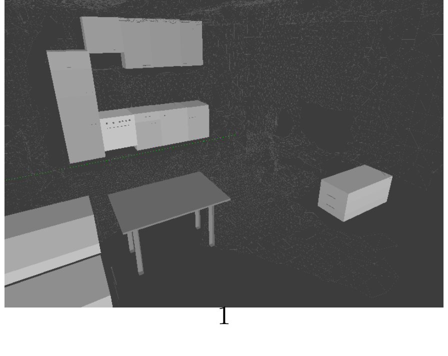
You should now see the Prolog console. The KnowRob taxonomy is loaded by default since mod_vis, the module we loaded, depends on ias_knowledge_base which contains the taxonomy. So you can start exploring the available classes, e.g. with

```
?- owl_subclass_of(A, knowrob:'FoodOrDrink'). <ENTER>
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#FoodOrDrink' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drink' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Coffee-Beverage' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#InfusionDrink' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Tea-Beverage' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Tea-Iced'
Yes.
```

Some notes on the query syntax: Predicates in a query can be linked with a comma, denoting the logical AND, or a semicolon for the logical OR. Each query is finished with a full stop. You can step through the results with the semicolon or just hit <ENTER> again.
Explore the KnowRob taxonomy starting from some of the classes listed in
`http://ias.in.tum.de/kb/wiki/index.php/KnowRob_Taxonomy`.
For more information on query predicates, have a look at the Semweb library documentation at
`http://www.swi-prolog.org/pldoc/package/semweb.html`.

1.1 Loading and querying environment information

So far, we only used the class level. For robotic applications, it is also important to reason about instances of these classes, for example observed objects or actions. An example set of instances are the semantic environment maps, contained as OWL file in the ias_semantic_map package and visualized in the picture on the next page.



Instances are queried using the or owl_has(S,P,O) predicate, which retrieves all triples with matching Subject, Predicate, or Object from the knowledge base. The following query, for example, asks for all objects of type knowrob:'Drawer'.

```
?- owl_has(A, rdf:type, knowrob:'Drawer').

A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer1' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer103' ;
A = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer109' ;
true.
```

For getting an overview of the information that is available about one object instance, we can query for all triples where the respective instance fills the Subject slot:

```
?- owl_has('http://ias.cs.tum.edu/kb/knowrob.owl#Drawer1', P, O).

  P = 'http://ias.cs.tum.edu/kb/knowrob.owl#describedInMap',
  O = 'http://ias.cs.tum.edu/kb/ccrl2_semantic_map.owl#SemanticEnvironmentMap0' ;
  P = 'http://ias.cs.tum.edu/kb/knowrob.owl#properPhysicalParts',
  O = 'http://ias.cs.tum.edu/kb/knowrob.owl#Door4' ;
  P = 'http://ias.cs.tum.edu/kb/knowrob.owl#depthOfObject',
  O = literal(type('http://www.w3.org/2001/XMLSchema#float', '0.574538')) ;
  P = 'http://ias.cs.tum.edu/kb/knowrob.owl#heightOfObject',
  O = literal(type('http://www.w3.org/2001/XMLSchema#float', '0.338121')) ;
  P = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',
  O = 'http://www.w3.org/2002/07/owl#NamedIndividual' ;
  P = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',
  O = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer' ;
  P = 'http://ias.cs.tum.edu/kb/knowrob.owl#widthOfObject',
  O = literal(type('http://www.w3.org/2001/XMLSchema#float', '0.58045006')) ;
  false.
```

1.2 Visualizing objects

A visualization often helps to find problems with information in the knowledge base, and is also useful to demonstrate what the robot knows about the world. Therefore, we created a visualization canvas that accepts object instances and allows to visualize them, highlight some objects, and retrieve information by clicking on the items.
For launching the launch visualization module, type

```
?- visualisation_canvas(C).
```

By default, the system loads the kitchen background. We will now clear the canvas, manually select some objects from the map, and push them to the visualization.

```
?- clear_canvas($C).
```

Note: $C refers to the last binding of the top-level variable C, in this case the handle identifying the canvas.
To select and visualize object instances, we call

```
?- owl_has(A, rdf:type, knowrob:'Drawer'), add_object(A, $C).
```

The small window that has opened in addition to the visualization canvas contains some control routines that mainly interact with the right section of the canvas, which displays action sequences and is not used here. In addition, it can show information about any kind of instance in the system, for instance the objects we just added to the canvas. One way of displaying this information is simply by clicking on the object, but you can also update the content from Prolog:

```
?- owl_has(A, rdf:type, knowrob:'Drawer'), display_information_for(A, $C).
```

2 Extending the Knowledge Base

The goal of this exercise is to describe the class *CoffeeCup* such that a cup that contains coffee can automatically be inferred to belong to this class. If you feel familiar with Protege, you should first try to create the class description on your own before resorting to the following step-by-step instructions.

Detailed instructions:

- Load the file coffeecup.owl from the owl folder in the tutorial package into Protege.
- Create a new object property *containsDrinkableLiquid* as a sub-property of *contains* with the domain *Container* and the range *Drink*.
- Create a new data property *volumeOfObject* with domain *Container* and range *float*.
- Create the subclasses *CoffeeCup* and *TeaCup* below the class *Cup*.
- Create a restriction on the two classes that they *containsDrinkableLiquid* some *Coffee-Beverage* or *Tea-Beverage* respectively (in the 'Equivalent classes' section).
- Create an instance *cup0* of a *Cup* with *volumeOfObject* 0.1.
- Create an instance of *Coffee-Beverage* and state that it is contained (*containsLiquid*) in *cup0*.
- You can now check if the system recognizes that *cup0*, which is asserted to be only a *Cup* is actually a *CoffeeCup*. For doing this, you select the HermiT reasoner by clicking on "Reasoner → HermiT", then click on "Reasoner → Classify...". If you look at *cup0* in the "Individuals" tab, you should see that it has the types *Cup* as well as *CoffeeCup*.
- Save the file.

Now you can load the file into KnowRob:

```
$ roscd knowrob_tutorial
$ rosrun rosprolog rosprolog knowrob_tutorial
?- owl_parser:owl_parse('owl/coffeecup.owl', false, false, true).
```

You should be able to query for the classes and instances you just created. Try also to query for all instances of CoffeeCup:

```
?- owl_individual_of(A, 'http://ias.cs.tum.edu/kb/coffeecup.owl#CoffeeCup').
A = 'http://ias.cs.tum.edu/kb/coffeecup.owl#cup0'
```

The system thus exploits its knowledge about cups to determine that *cup0* is not only an instance of *Cup*, but actually a *CoffeeCup*.

3 Interfacing the table_object detector

3.1 Launching the system

- Launch with the launch file in knowrob_tutorial `roslaunch knowrob_tutorial tabletop.launch`
- Start the KnowRob system `rosrun rosprolog rosprolog knowrob_tutorial`
- Play bag file `rosbag play tabletop.bag`
- Send the queries to the tabletop_object_detector while the bag file is playing

3.2 Useful predicates

- **+** indicates that the argument has to be passes as a parameter
- **-** indicates that the argument is returned as result

```
jpl_new(+Class, +Arguments, -Value)
jpl_call(+Class, +Method, +Args, -Result)
jpl_call(+Object, +Method, +Args, -Result)
jpl_get(+Object, +Field, -Value)
jpl_array_to_list(+JArray, -PlList)

create_perception_instance(+ModelTypes, -Perception)

create_object_instance(+ObjTypes, +CopID, -Obj)

set_object_perception(?A, ?B)

set_perception_pose(Perception, [M00, M01, M02, M03,
                                  M10, M11, M12, M13,
                                  M20, M21, M22, M23,
                                  M30, M31, M32, M33])
```