Object Designator

Contents

Object designators are used to describe objects located in the BulletWorld or the real environment and then resolve them during runtime to concrete objects.

Object designators are different from the Object class in bullet_world.py in the way that they just describe an object and do not create objects or provide methods to manipulate them. Nevertheless, object designators contain a reference to the BulletWorld object.

An Object designator takes two parameters, of which at least one has to be provided. These parameters are:

- A list of names
- A list of types

Object Designators work similar to Location designators, they get constrains describing a set of objects and when resolved return a specific instance.

For all following examples we need a BulletWorld, so let's create one.

```
from pycram.worlds.bullet_world import BulletWorld
from pycram.world_concepts.world_object import Object
from pycram.datastructures.enums import ObjectType, WorldMode
from pycram.datastructures.pose import Pose

world = BulletWorld(WorldMode.GUI)
```

Believe Object

This object designator is used to describe objects that are located in the BulletWorld. So objects that are in the belief state, hence the name. In the future when there is a perception interface, there will be a `RealObject` description which will be used to describe objects in the real world.

Since **`BelieveObject()`** describes Objects in the BulletWorld we create a few.

```
kitchen = Object("kitchen", ObjectType.ENVIRONMENT, "kitchen.urdf")
milk = Object("milk", ObjectType.MILK, "milk.stl", pose=Pose([1.3, 1, 0.9]))
cereal = Object("froot_loops", ObjectType.BREAKFAST_CEREAL, "breakfast_cereal.s
spoon = Object("spoon", ObjectType.SPOON, "spoon.stl", pose=Pose([1.3, 1.1, 0.8
```

Now that we have objects we can create an object designator to describe them. For the start we want an object designator only describing the milk. Since all objects have unique names we can create an object designator using a list with only the name of the object.

```
from pycram.designators.object_designator import BelieveObject

object_description = BelieveObject(names=["milk"])

print(object_description.resolve())
```

You can also use the type to describe objects, so now we want to have an object designator that describes every food in the world.

```
from pycram.designators.object_designator import BelieveObject

object_description = BelieveObject(types=[ObjectType.MILK, ObjectType.BREAKFAST

print(object_description.resolve())
```

Object Part

Part of object designators can be used to describe something as part of another object. For example, you could describe a specific drawer as part of the kitchen. This is nec latest drawer is no single BulletWorld Object but rather a link of the kitchen which is a BulletWorld Object.

For this example we need just need the kitchen, if you didn't spawn it in the previous example you can spawn it with the following cell.

```
kitchen = Object("kitchen", ObjectType.ENVIRONMENT, "kitchen.urdf")
```

```
from pycram.designators.object_designator import ObjectPart, BelieveObject

kitchen_desig = BelieveObject(names=["kitchen"]).resolve()

object_description = ObjectPart(names=["sink_area_left_upper_drawer_main"], par

print(object_description.resolve())
```

Object Designators as Generators

Similar to location designators object designators can be used as generators to iterate through every object that they are describing. We will see this at the example of an object designator describing every type of food.

For this we need some objects, so if you didn't already spawn them you can use the next cell for this.

```
kitchen = Object("kitchen", ObjectType.ENVIRONMENT, "kitchen.urdf")
milk = Object("milk", ObjectType.MILK, "milk.stl", pose=Pose([1.3, 1, 0.9]))
cereal = Object("froot_loops", ObjectType.BREAKFAST_CEREAL, "breakfast_cereal.s
spoon = Object("spoon", ObjectType.SPOON, "spoon.stl", pose=Pose([1.3, 1.1, 0.8
```

```
from pycram.designators.object_designator import BelieveObject

object_description = BelieveObject(types=[ObjectType.MILK, ObjectType.BREAKFAST

for obj in object_description:
    print(obj, "\n")
```

To close the world use the following exit function.


```
world.exit()
```