EASE Fall School 2021

Sascha Jongebloed

November 2021

This exercises are based on exercises written by Daniel Beßler and Sascha Jongebloed.

1 Exercise

In this exercise you will write your own predicates and queries to reason about the symbolic representation of a robot, and the events performed by the robot within a recorded NEEM.

You will write the code for this exercise in a Jupyter Notebook offered here: `https://github.com/sasjonge/ease_fs_neems_2021`. Please follow the instructions given in the README of the repository.

1.1 Robot Structure

In the first part of this exercise you will define relationships that are implemented via reasoning about the symbolic representation of the robot.

1. Write the predicate robotPart(*Robot*, *Part*, *PartType*) where *Part* is reachable by *Robot* through the transitive closure of the parthood relationship *dul:'hasPart'*, and *PartType* is a type of *Part*, and a subclass of *dul:'PhysicalObject'*. Tip: The type of an object can be obtained by *rdf:type*, and its subclasses by *rdfs:subClassOf*.

2. Write a query that answers the following questions: (1) What are the reference frames associated to the base link of arms of the PR2 robot, and (2) how are the arms positioned relative to the base of the robot? Tips: The PR2 robot is identified by `'http://knowrob.org/kb/PR2.owl#PR2_0'`, the type of arm components is `'http://www.ease-crc.org/ont/SOMA.owl#Arm'`, and their base link reference frame can be obtained through their property *urdf:hasBaseLinkName*. The base of the robot is of type: `'http://knowrob.org/kb/knowrob.owl#MobileBase'`. The position of an object O can be obtained via the 2-ary predicate tf:tf_get_pose, written as tf:tf_get_pose(O, [F, *Position*, _]), where the position of O is expressed relative to the reference frame F.

1.2 Episode Structure

In this exercise, you will write queries for a recorded robotic NEEM.
Write queries that:

1. Write a query that yields the shortest action in which a transporting task was executed. Tip: You can use event_interval(*Action*, *Start*, *End*), where *Action* is the given action, and *Start* and *End* the start- and endtimepoint of the action. Use number(*End*) to ensure that *Start* and *End* is bound to a rational number. To obtain the executed task you can use the property *dul:executesTask*. The type of the task is *soma:'Navigating'*.

2. Write a query that yields the type of tasks that are executed in actions in which an agent manipulates an item. Tip: Being an item in an action is the role an object plays when a task is executed, the roles of a task are denoted by the *dul:isTaskOf* property. The type of the role of manipulated items is *soma:'Item'*, and, for agents in a task, the role type is *soma:'AgentRole'*.

You can also run the queries of this task in openEASE. We recommend the following NEEM for this purpose: `https://data.open-ease.org/QA?neem_id=6155c8416dff5d779421feba`

2 Solutions

2.1 Robot Structure

1.
```
robotPart(Robot, Part, PartType) :-
        triple(Robot, dul:'hasPart', Component),
        (       robotPart1(Component, Part, PartType)
        ;       robotPart(Component, Part, PartType)
        ).

robotPart1(Component, Component, PartType) :-
        triple(Component, rdf:type, PartType),
        triple(PartType, rdfs:subClassOf, dul:'PhysicalObject').
```

2.
```
% Get the reference frame of the robot base
robotPart('http://knowrob.org/kb/PR2.owl#PR2_0', Base,
    'http://knowrob.org/kb/knowrob.owl#MobileBase'),
triple(Base, urdf:hasBaseLinkName, BaseFrame),
% Get the arm and the corresponding reference frame
robotPart('http://knowrob.org/kb/PR2.owl#PR2_0', Arm,
    'http://www.ease-crc.org/ont/SOMA.owl#Arm'),
triple(Arm, urdf:hasBaseLinkName, ArmFrame),
% Get the position of the arm relative to the baseframe
tf:tf_get_pose(Arm, [BaseFrame, Position, _]).
```

2.2 Episode Structure

1. ```
   % Get all events and their durations
   findall([Duration, Act],
     (
        % Get the time interval
        event_interval(Act, Begin, End),
        % Ensure that the Action executes a Transporting task
        triple(Tsk, rdf:type, soma:'Navigating'),
        triple(Act, dul:'executesTask', Tsk),
        % Ensure that the end is bound to a number
        number(End),
        % Calculate the duration
        Duration is End - Begin
     ),
     Durations),
   % Get the minimal member of the solutions
   min_member([MinDuration, ShortestEvent], Durations).
   ```

2. ```
   % Get the executed Task
   triple(Act, dul:'executesTask', Tsk),
   % Ensure that there is a agent
   % and item during the Action
   triple(Tsk, dul:isTaskOf, ObjectRole),
   triple(ObjectRole, rdf:type, soma:'Item'),
   triple(Tsk, dul:isTaskOf, AgentRole),
   triple(AgentRole, rdf:type, soma:'AgentRole'),
   % Return the type of the task
   triple(Tsk, rdf:type, TskType).
   ```