**Pi2Go Simulator Programming: The Cognitive Agent Reasoning Cycle**

**AIM:** After completing this worksheet you should be able to explain the Reasoning cycle for your Python agent and program simple rules for it.

**You Need:** To complete this worksheet you need to have a virtual Pi2Go simulator (see WS1) and understand its sensors and actuators (WS3 & WS4). You should know about Python’s time module (WS6), Python Functions (WS16), Data Structures (WS19-21) and objects (WS27). You should also be familiar with basic Cognitive Agent constructs (WS28).

**If the simulator isn’t already running: Start the Simulator, Select the Pi2go Simulation and default\_world.xml, then start IDLE (open a *new IDLE window* if you have used IDLE to start the simulator).**

**The Reasoning Cycle:** As well as beliefs and goals the Pi2Go Agent has a *reasoning cycle.* When the cycle is active the agent does four things in turn:

1. It checks all its sensors and updates its belief base
2. It checks to see if any goals have been achieved and removes them
3. It checks to see if any *rules* are applicable. If some are it picks one and,
4. executes it

**How do you make the reasoning cycle active?** By calling the method run\_agent (run\_agent will also initialise the agent if you have not already done so)

**What is a rule?** A rule is a python function that has been added to the agent.

**How do you add a rule to the agent?** By calling the method add\_rule

**When is a rule applicable?** Rules are always applicable unless they have a condition. We will talk about conditions in the next section.

Consider the following program:

import bdi.pi2goagent as cognitive

agent = cognitive.Pi2GoAgent()

def print\_beliefs():

print(agent.beliefbase)

return

agent.add\_rule(print\_beliefs)

agent.run\_agent()

What will happen when it is run?

**Exercise 1:** Modify the program so that it prints out just the value of the distance sensor on each cycle through the reasoning cycle.

**Adding conditions to rules:** It isn’t very useful having a rule that is always applicable. It would be better if rules were only executed in certain situations. In order to do this, we the method add\_condition\_rule this takes two arguments – the first is a function that be executed to decide if the rule is applicable and the second is a rule function.

Consider the following program:

import bdi.pi2goagent as cognitive

agent = cognitive.Pi2GoAgent()

def check\_distance():

if (agent.beliefbase['distance'] < 50):

return True

return False

def print\_beliefs():

print(agent.beliefbase)

return

agent.add\_condition\_rule(check\_distance, print\_beliefs)

agent.run\_agent()

What will happen when it is run?

**Belief Support:** The Pi2GoAgent has methods that will help with referring to beliefs in conditions:

*agent*.believe(*key*)creates a function that checks whether *key* takes the value 1 or True in the agent’s belief base

*agent*.B(*key*)is the same as *agent.*believe(*key*) but is quicker to type.

**Exercise 2:** Write a program that will make the robot reverse if it believes there is an obstacle in front of the centre infra-red sensor. **Remember:** you can use agent.robot to access the robot’s functions.

**Stopping the Reasoning Cycle:** You can stop the agent reasoning cycle by calling the function done()

**Exercise 3:** Modify your program from the last exercise that will make the robot reverse for five seconds if it believes there is an obstacle in front of it and then the robot will stop, and the agent will stop the reasoning cycle.



University of Liverpool, 2019

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).