**Virtual Pi2Go Programming: Introduction to Cognitive Agents**

**AIM:** After completing this worksheet you should be able to describe what a cognitive agent is.

**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 objects (WS27), and Data Structures (WS19-21)

**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).**

### What are Cognitive Agents?

Cognitive agents (sometimes called *Beliefs-Desires-Intentions* or *BDI* agents) are an artificial intelligence technology. A cognitive agent has beliefs about the world and goals it wants to achieve. Its actions are governed by its beliefs and goals. In this worksheet we will explore a cognitive agent module for use with the Pi2Go simulator.

At the idle command line type the following:

import bdi.pi2goagent as cognitive

This imports a Python module that implements cognitive agent programming. The first thing we need to do is create an agent. This is an object of Pi2GoAgent class.

Type:

agent = cognitive.Pi2GoAgent()

agent.init()

The agent will control our Initio simulator. This means we still need to initialise our simulation.

### Beliefs

The agent can check all its sensors for *perceptions*. It can do this using the getPercepts() method. When it checks its sensors, it stores their values in its *BeliefBase.*

Type

agent.getPercepts()

print(agent.beliefbase)

**Question 1:** What is printed?

The belief base is a dictionary of pairs of strings and values. So we have the following strings for each sensor

|  |  |
| --- | --- |
| String | Sensor |
| distance | Ultrasonic distance Sensor |
| Obstacle\_centre | Centre infra-red sensor |
| obstacle\_right | Right infra-red sensor |
| obstacle\_left | Left infra-red sensor |
| line\_right | Right line sensor |
| line\_left | Left line sensor |
| switch\_pressed | The switch |
| lightFL | Front left light sensor |
| lightFR | Front right light sensor |
| lightBL | Back left light sensor |
| lightBR | Back right light sensor |

**Question 2:** What does your agent believe is the current value of the ultrasonic distance sensor?

You can add your own beliefs to the belief base using the methods add\_belief and change\_belief.

You use add\_belief if the belief you are adding has a true/false value. So, for instance you could add a belief that you are currently testing the agent:

Type:

agent.add\_belief(‘test’)

**Question 3:** Now print out the contents of the belief base. How has it changed?

You use change\_belief if the belief you are adding as some other value, such as a string. So, for instance, you could add a belief that the agent’s name is “pi2go”.

Type:

agent.change\_belief(‘name’,’pi2go’)

**Quesiton 4:** Now print out the contents of the belief base. How has it changed?

### You can remove a belief from the agent’s belief base using remove\_belief

Type:

agent.remove\_belief(‘name’)

**Question 5:** Now print out the contents of the belief base. How has it changed?

### Important: Since the belief base is a dictionary you can check if something is in the belief base using dictionary functions. For instance

Type:

‘distance’ in agent.beliefbase

**Question 6:** What happens? Explain why.

Type:

agent.beliefbase[‘distance’]

**Question 7:** What happens? Explain why.

### Goals

The agents goals are things it wants to achieve. That is things it wants to believe are true. The agent’s *goalbase* is a list of strings representing its goals. You can add a goal using the method add\_goal.

Type:

agent.add\_goal(‘a\_goal’)

print(agent.goalbase)

What is printed?

When an agent believes a goal then it can be removed. This is done using the method check\_goals().

Type:

agent.add\_belief(‘a\_goal’)

agent.check\_goals()

print(agent.goalbase)

What is printed and why has the goal base changed?

**Controlling the Robot**

You may wish to control the robot through the agent. The Simulated Pi2Go is stored as a data field, robot.

Type:

agent.robot.forward(10)

agent.robot.stop()

What happens and why?



 University of Liverpool, 2020

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