**Virtual Pi2Go Programming: BDI Goals**



**AIM:** After completing this worksheet you should be able to explain how BDI goals differ from beliefs and use them in programs.

**You Need:** To complete this worksheet you need to have a virtual Pi2Go simulator (see WS1), and to be able to use files to store Programs (WS5). You also need to know the commands to operate the Pi2Go motors, LEDs and sensors (WS3 & WS4). You should be able to use Python’s time module (WS6), variables (WS12), functions (WS16) and objects (WS27 & WS31). You should also understand how to use rules with cognitive agents (WS29 & WS30).

**If the simulator isn’t already running: Start the Simulator, Select the Pi2Go Simulation and square.xml, then start IDLE (open a *new IDLE window* if you have used IDLE to start the simulator).**

We have seen how we can use beliefs in rule conditions to control an agent’s behaviour. We can also use goals. Unlike beliefs, goals are sometimes automatically removed. This happens when they are *achieved.*

Consider the following program

import bdi.pi2goagent as cognitive

import time

class FindSquareAgent(cognitive.Pi2GoAgent):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.add\_condition\_rule(self.G('line\_left'), self.forward\_rule)

self.add\_goal('line\_left')

def forward\_rule(self):

self.robot.forward(10)

time.sleep(5)

self.robot.stop()

**Question 1:** What does this agent do?

**Question 2:** Create an instance of this agent and run its reasoning cycle. What happens?

**Exercise 1:** Change to **house.xml** world. Create an agent that starts with a goal to enter the house (by achieving line\_left). While it has this goal, the agent should move forward until it detects the wall of the house and then follow the wall until it enters the house. You may want to look back at your answers to WS18 To help with this.

**Hint:** Consider creating the following variables to represent conditions you will need in your \_\_init\_\_ method

only\_line\_goal = self.AND(self.G('line\_left'), self.NOT(self.G('obstacle\_centre')))

goal\_and\_obstacle = self.AND(self.G('line\_left'), self.B('obstacle\_centre'))

goal\_and\_no\_wall = self.AND(self.G('line\_left'), self.NOT(self.B('obstacle\_right')))

goal\_and\_wall = self.AND(self.G('line\_left'), self.AND(self.B('obstacle\_right'), self.NOT(self.B('obstacle\_centre'))))



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/).