**Virtual Initio Programming: WS30 & Ex30 Sample Answers**

**NOTE:** Exercises have become sufficiently complex that quite wide variability in answers can be expected.

**WS30**

**Question 1:** When there is an obstacle on the left and on the right.

**Question 2:** When the there is an obstacle on the left and the agent does not believe ‘started’ because this has not been added to the belief base.

**Question 3:** When the switch is pressed and when the agent believes ‘started’ because this has been add to the belief base.

**Question 4:** When ‘started’ has been added to the belief base.

**Question 5:** When there is a line on the left ‘started’ is added to the belief base (and the agent sleeps for 5 seconds – to allow the line to “vanish”) Once it believes it has started the robot starts to move forward. When a line appears on the left again again (or if it is not removed within 5 seconds) then the started belief is dropped and a stopping belief is added. Once it believes it is stopped the agent stops the robot, the reasoning cycle and drops the stopping belief.

**Exercise 1:**

import bdi.initioagent as cognitive

import time

agent = cognitive.InitioAgent()

def start\_agent():

agent.add\_belief('started')

time.sleep(5)

return

def stop\_agent():

agent.drop\_belief('started')

agent.add\_belief('stopping')

time.sleep(5)

return

def forward():

agent.robot.forward(10)

return

def turn():

agent.robot.spinLeft(10)

return

def stop\_rule():

agent.robot.stop()

agent.done()

agent.drop\_belief('stopping')

return

start = agent.AND(agent.B('line\_left'), agent.NOT(agent.B('started')))

stop = agent.AND(agent.B('line\_left'), agent.B('started'))

no\_obstacle = agent.AND(agent.B('started'), agent.AND(agent.NOT(agent.B('obstacle\_right')), agent.NOT(agent.B('obstalce\_left'))))

obstacle = agent.AND(agent.B('started'), agent.OR(agent.B('obstacle\_left'), agent.B('obstacle\_right')))

agent.add\_condition\_rule(start, start\_agent)

agent.add\_condition\_rule(stop, stop\_agent)

agent.add\_condition\_rule(no\_obstacle, forward)

agent.add\_condition\_rule(obstacle, turn)

agent.add\_condition\_rule(agent.B('stopping'), stop\_rule)

agent.run\_agent()

**Ex30**

**Exercise 1:** Note that because this is a line following agent, I’m using obstacle\_left to start and stop it.

import bdi.initioagent as cognitive

import time

agent = cognitive.InitioAgent()

def start\_agent():

agent.add\_belief('started')

time.sleep(5)

return

def stop\_agent():

agent.drop\_belief('started')

agent.add\_belief('stopping')

time.sleep(5)

return

def stop\_rule():

agent.robot.stop()

agent.done()

agent.drop\_belief('stopping')

return

def forward():

agent.robot.forward(10)

return

def left():

agent.robot.spinLeft(10)

return

def right():

agent.robot.spinRight(10)

start = agent.AND(agent.B('obstacle\_left'), agent.NOT(agent.B('started')))

stop = agent.AND(agent.B('obstacle\_left'), agent.B('started'))

on\_line = agent.AND(agent.B('started'), agent.NOT(agent.OR(agent.B('line\_left'), agent.B('line\_right'))))

line\_on\_left = agent.AND(agent.B('started'), agent.B('line\_left'))

line\_on\_right = agent.AND(agent.B('started'), agent.B('line\_right'))

agent.add\_condition\_rule(start, start\_agent)

agent.add\_condition\_rule(stop, stop\_agent)

agent.add\_condition\_rule(on\_line, forward)

agent.add\_condition\_rule(line\_on\_left, left)

agent.add\_condition\_rule(line\_on\_right, right)

agent.add\_condition\_rule(agent.B('stopping'), stop\_rule)

agent.run\_agent()

**Exercise 2:**

import bdi.initioagent as cognitive

import time

agent = cognitive.InitioAgent()

def start\_agent():

agent.add\_belief('started')

time.sleep(5)

return

def stop\_agent():

agent.drop\_belief('started')

agent.add\_belief('stopping')

time.sleep(5)

return

def stop\_rule():

agent.robot.stop()

agent.done()

agent.drop\_belief('stopping')

return

def forward():

agent.robot.forward(10)

return

def left():

agent.robot.spinLeft(10)

return

def right():

agent.robot.forward(10)

time.sleep(1)

agent.robot.spinRight(10)

time.sleep(2)

return

def b\_obstacle\_centre():

if (agent.beliefbase['distance'] < 30):

return True

return False

start = agent.AND(agent.B('obstacle\_left'), agent.NOT(agent.B('started')))

stop = agent.AND(agent.B('obstacle\_left'), agent.B('started'))

wall\_on\_right = agent.AND(agent.B('started'), agent.AND(agent.B('obstacle\_right'), agent.NOT(b\_obstacle\_centre)))

wall\_in\_front = agent.AND(agent.B('started'), b\_obstacle\_centre)

lost\_wall = agent.AND(agent.B('started'), agent.NOT(agent.OR(agent.B('obstacle\_right'), agent.B('obstacle\_right'))))

floor = agent.AND(agent.B('started'), agent.B('line\_left'))

agent.add\_condition\_rule(start, start\_agent)

agent.add\_condition\_rule(stop, stop\_agent)

agent.add\_condition\_rule(floor, stop\_rule)

agent.add\_condition\_rule(wall\_on\_right, forward)

agent.add\_condition\_rule(wall\_in\_front, left)

agent.add\_condition\_rule(lost\_wall, right)

agent.add\_condition\_rule(agent.B('stopping'), stop\_rule)

agent.run\_agent()



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/).