Student Name: Ziglio Riccardo
Student ID: 224285

Autonomous Software Agents
Final Project Report

Introduction

This smart home domain was thought to be resident oriented, with the goal of facilitating the residents in their everyday life. The house consists of several devices, some are smart (such as the heating system, the lights in each room, and the windows blinds), and some are not (like the dishwasher and the washing machine). All smart devices have their own properties, are able to define their internal status and can be monitored and controlled using sensors and actuators. All devices in the house are controlled by the house agent, the brain of this smart home. The house agent is able to schedule tasks (i.e. start the vacuum cleaner at 15), set-up house devices based on the context (i.e. turn on the heater and the light in the kitchen when the resident is having lunch), while still being responsive to the residents behaviour (i.e. turn on the light when someone walks in a room).

In the domain, in addition to the house agent, other two agents operate: the security agent and the vacuum agent. The latter is in charge of keeping the house clean, while the former controls the security of the house by making sure that at night both the front door is locked and all the blinds of the house are closed.

In this domain the residents are able to communicate with the agent either using a smartphone application (i.e. to be notified if the fridge is empty and groceries are needed), or by using wall mounted terminals (i.e. a thermostat for setting the desired heater temperature).

Changes from previous assignments

With respect to the previous two assignments most of the project structure has changed. So, this report will focus only on the newer version of the project, without commenting again the old one. Please note that, as with the previous version of this project, the starting point was the Autonode.js project (`https://github.com/marcorobol/Autonode.js`) introduced in the lab classes, upon which the final project solution was implemented. The final project code described in this report can be accessed at the following Github repository: `https://github.com/RickyZi/ASA-project`.

House description and blueprint

The house was designed for a single resident, called Bob. It is composed by six rooms all located on one floor, as showed in Figure 1. As we can see, it is possible to access the house through the front door which leads to the entrance. From the entrance we come directly into the living room, which is connected with the kitchen and the hallway leading to the bathroom and the bedroom.

Figure 1: Blueprint of the house.

Rooms

Each room, a part from the hallway and the entrance, has an independent thermostat to control the heating and a window with blinds. Also, there is at least one main light in those rooms. There are different sensors, located throughout the house, used by the house agent to recognize the resident moving between rooms in the house and to be able to turn lights on or off accordingly.

Entrance

Is the first room we enter in the house, accessible through the front door. In the entrance we can find a light, the vacuum agent charging station and a door leading to the living room.

Living Room

The living room is accessible from the entrance, and leads to the kitchen and the hallway. In the living room we can find a big window to illuminate the room during the day, a TV, a sofa and a dining table.

Kitchen

Accessible from the living room, is furnished with a stove top, a dishwasher, a fridge, an oven, a dining table and a sink. Both the fridge and the dishwasher are not smart devices, meaning they can only be controlled manually by the resident. The only information the house agent has about them is their status, i.e. the fridge may be empty or full, and also the amount of electricity they consume.

Hallway

From the living room door, it is possible to reach the hallway that provides access to the bedroom and bathroom through two independent doors. As stated before, the hallway has no heater.

Bathroom

Accessible from the hallway, it is provided with a main light, a window, a heater and a washing machine. The washing machine is another device that can only be controlled manually by the resident. And also in this case, the house agent keeps track of its status and electricity consumption.

Bedroom

Also this room is accessible from the hallway. It is furnished with a main light, a big window, a heater, and also other bedroom furniture such as a bed and a nightstand.

Devices

Lights

Lights provide illumination to the rooms. The possible light status are: *on* or *off*. Actions that can be done on the lights are: *switchOnLight* and *switchOffLight*. Every time a light is turned on it consumes *10W* of electricity. The house agent is able to turn on and off each main light in the room when a person walks in and turn it off when the room is empty to reduce electricity consumption.

Heaters

They can be controlled either by using the thermostat located in each room where a heater is present or by the house agent according to the schedule defined by the user. This fixed schedule helps reducing the electricity consumption and maximize the resident comfort. Each heater consumes *15 W* when it is turned on.

The possible actions that can be done on the heaters are: *switchOnHeater*, *switchOffHeater*, and *setNewTemperature* to modify the preset temperature of 18 degrees.

Blinds

Are used by the resident to darken the bedroom for a better sleep. They also serve the function of providing privacy for residents at night. Thus, the security agent will make sure that every night (at 22) all the house blinds are closed. The security agent will also open all the blinds in the morning when the resident wakes up. The possible blinds status are either *open* or *closed*. The status can be modified by using the actions *openBlinds* and *closeBlinds*.

Door Lock

The front door lock is managed by the security agent, which will make sure that the front door is locked at nigh (at 23). The security agent will also unlock the door in the morning (at 7) when the resident leaves for work. The possible status of the door lock are *locked* and *not_locked*. The actions that can be performed to modify the door lock status are *lockDoor* and *unlockDoor*.

Fridge

The fridge maintains an overall status of supplies shortage. Its possible status can be either *full*, *half*, or *empty*. Actions that can be done include *set_empty*, *set_full*, *set_half*. The status of the fridge is tracked by the house agent, which will notify residents in case of low supply. The fridge consumes an average of *500W* per day, is always on and its consumption are tracked by the house agent.

Washing Machine

It is positioned in the bathroom. The possible washing machine status are *on* or *off*. Actions that can be performed are *switchOnWashingMachine* or *switchOffWashingMachine*. It consumes an average of *300Wh* per washing cycle. The washing machine is manually activated by the user, usually between 18 and 19, when electricity is cheaper. The house agent will only keep track of the washing machine electricity consumption.

Dishwasher

Positioned in the kitchen, its status may be *on* or *off*. The possible actions are *turn_on*, or *turn_off*. It consumes *1200Wh* per cleaning cycle. As for the washing machine, the dishwasher is triggered by the user (usually between 21 and 22) and the house agent only keeps track of its electricity consumption.

Vacuum Cleaner

It is located in the entrance, where we can also find its charging station. Its possible status are *on* or *off*, consequently its battery status can either be *discharging* or *charging*. The vacuum cleaner can perform the following actions: *turnOn*, *turnOff*, *Move(to_room)*, and *Clean(room)*. Consumes *90W* of electricity to recharge the batteries each time it is used.

The vacuum cleaner is controlled by a planning agent called **vacuum agent**. The main objective of this agent is to develop a plan to clean all the rooms autonomously. In order to do so it must be informed by the house agent about which rooms are dirty, and must also know the structure of the house (which rooms are there and how they are connected to each other).

Metrics

Cost of electricity

Since in this domain there is no way to produce electricity, the resident will always buy electricity. The electricity cost **0.37 €/KWh** during the day (from 8 to 17) and **0.3 €/KWh** during the night (form 19 to 7) and during the weekend. Thus, the house agent arranges to use the devices during times of day when electricity is cheaper: from 19 to 23 during the week, and from 7 to 23 during the weekend. Moreover, in order to reduce as much as possible the electricity consumption during the day, the house agent will turn off all the house heaters at night (at 22) and also turn off the light in a room if this is empty.

The vacuum cleaner takes almost 3 hours to clean the whole house, so it is scheduled to run from 15 to 18 when the house is empty and all the other electronic appliances are turned off.

People and Agents

People

There is only one resident in the house, called Bob. He can be either in one room at the time or outside the house (simulated with the room called *outdoor*). The modeled scenario shows the behaviour of the resident during the week using a fixed schedule. According to this schedule, Bob usually wakes up at 6, have breakfast around 6.30 in the kitchen and then leaves for work at 7. He will then comes back for lunch at 13 and leaves again for work at 14.30. Around 19 Bob finish working and will come back home, have dinner around 19.30, then relax in the living room until he decides to go to sleep (around 22).
In the weekend the resident takes the full control of the house, overriding all agents tasks.

Agents

House Agent

The house agent assists the resident by taking autonomous decisions, while still being responsive to his behaviour. It also tries to minimize energy consumption.

The house agent performs the following tasks:

1. Switch on the main light when someone walks in a room;
2. Turn off the main light when a room is empty to reduce energy consumption;
3. Turn on and off the house heaters according to the resident behaviour;
4. Tell the vacuum agent which rooms are dirty and need to be cleaned;
5. Make sure the heater and the light in the kitchen are turned on when the resident is eating in that room for maximum resident comfort.

Security Agent

The security agent is responsible for:

- Locking the front door lock at 23;
- Unlocking the front door at 7 when Bob leaves for work;
- Closing all the house blinds at 22;
- Opening all the house blinds at 6, when Bob wakes up.

Vacuum Agent

The vacuum agent has the task of keeping the house clean. It runs every day from 15 to 18, when the house is empty. It was developed as a **planning agent** that can move autonomously between all the rooms and clean them. In order to do that, it needs to know the structure of the house and how to reach all the rooms, and which rooms are dirty. These information are shared by the house agent.

Implementation

Agents and Sensors

As discussed during the theoretical part of the course, our agents follows the *BDI architecture*. BDI stands for, *Beliefs*, *Desires (Goal)*, and *Intentions*. Each agent has a different goal, or set of goals, it wants to achieve. To do so it has to develop a plan, composed by a particular sequence of actions (also called intentions) that allow the agent to satisfy its desires (goals).

In order to develop a plan the agent has to reason on what it knows about the environment. The agent's environment knowledge is encoded by a collection of facts (composed by predicates and arguments), which can be either true or false, called *belief set*. The agent knowledge is limited only to what it can perceive about the environment. In order to notify the agent about changes in the observed portion of the environment, we use *sensors*. Sensors have been implemented using the **notifyChange()** function on the property of the device we are interested in. The idea is that, every time the status of the device changes, i.e. a light is turned on or off, the agent is informed about it and updates its belief set accordingly using the **declare()** method. An example showing how the agent's belief set is updated when the lights are turned on and off is reported in the following code.

```
*exec () {
    var lightsGoals = []
    for (let l of this.lights) {
        let lightGoalPromise = new Promise( async res => {
            while (true) {
                let status = await l.notifyChange('status')
                this.log('sense: light ' + l.name + ' switched ' + status)
                this.agent.beliefs.declare('light_on '+l.name, status=='on')
                this.agent.beliefs.declare('light_off '+l.name, status=='off')
            }
        });

        lightsGoals.push(lightGoalPromise)
    }
    yield Promise.all(lightsGoals)
}
```

As previously mentioned, there are different agents interacting in our smart home scenario: the *house agent*, the *security agent*, and the *vacuum cleaner agent*.

The *house agent* will maintain the overall status of the house, being informed of everything that happens in the house. In its belief set we can find information such as:

- in which room a person is: **in_room Bob bedroom**.
- if a room is empty: **empty kitchen**.
- the lights status (on or off): **light_on kitchen_light**.
- the heater status (on or off): **on kitchen_heater**.
- the status of not smart devices (the dishwasher, the washing machine and the fridge): **empty fridge**, **on WM**, **on DW**.
- which rooms are dirty and need to be reported to the vacuum agent in order to be cleaned: **dirty entrance**.

The *security agent* has a more limited view of the environment. It only concentrates on the status of the front door lock and on the status of the blinds, i.e. **locked front_door**, **not open bedroom_blinds**. All other devices in the house are inaccessible to the security agent.

Finally, the *vacuum agent* belief set contains information not only on the structure of the house, the rooms and how they are connected to each other, but also which rooms are dirty and need to be cleaned.

Agents Intentions

As introduced in the previous section, different agents have different goals. In order to achieve them, the agents have to reason on their belief set to find the best sequence of actions satisfying the goal. This kind of reasoning is only performed by the house agent and by the vacuum agent. Instead, the security agent simply verifies that the entrance door and the house blinds are respectively locked and closed at night, by using the method *notifyChange()* to check if their status is defined as *locked front_door* or as *closed blinds_name* in its belief set. So, if at 23 the front door is not locked or the blinds are not closed, the security agent will close them. The same idea is applied for unlocking the front door and opening all the blinds in the morning. The *blindsManager.js* and in the *DoorLockManager.js* files implement the security agent intentions.

In contrast to the security agent, both the *house agent* and the *vacuum agent* perform some form of autonomous reasoning.

The vacuum agent implementation will be described in the next section. In the following we will just concentrate on how the house agent's tasks are implemented.

As previously stated, one of the tasks of the house agent is to turn on and off the house heaters according to the resident schedule. Following the same reasoning as the security agent, the house agent verifies that the heaters are in the desired status using the *notifyChange()* method (as shown in the *heatersManager.js* file).

The other task performed by the house agent is to **automatically** turn on and off the room lights when a person enters or leaves the room. This helps reduce electricity wasted due to lights accidentally left on by the resident. In order to perceive the movement of the resident in the house, the agents uses some motion sensors scattered throughout the house (not modelled, as we assumed this is something the house agent has direct access to). These sensor will alert the house agent when a resident is moving to another room. Thus, every time there is a change in the *in_room* attribute, meaning the person is moving to a different room, the agent will be notified and will update its beliefs. Then, the agent will check if the room to which the resident is moving to has the main light turned on. If it does not, the agent will turn it on. The agent will also turn off the main light in the previous room to reduce electricity consumption. In doing so, the agent will update its own belief on the status of the lights. The implementation of this house agent behaviour is shown in the code below (taken from *PersonSensor.js*).

```
*exec () {
        var PeopleGoals = []
        for (let p of this.People) {
            // let lightGoalPromise = this.agent.postSubGoal( new SenseOneLightGoal(l) )
            // lightsGoals.push(lightGoalPromise)

            let PeopleGoalPromise = new Promise( async res => {
                while (true) {

                    let prev_room = p.in_room
                    let new_room  = await p.notifyChange('in_room')


                    this.log('sense: person ' + p.name + ' moved to ' + new_room+ ' from ' + prev_room)

                    // num_ppl used to kepp track of how many people are in a room (in this case since there is only one resident it will be either 1 or 0)

                    if(new_room != prev_room && new_room != 'outdoor' && p.house.rooms[new_room].num_ppl > 0 && p.house.devices[new_room+'_light'].status == 'off'){
                        p.house.devices[new_room+'_light'].switchOnLight();
                    }

                    if(prev_room != 'outdoor' && p.house.rooms[prev_room].num_ppl == 0){
                        //console.log('person in room '+ prev_room + ': 0');
                        p.house.devices[prev_room+'_light'].switchOffLight();
                    }

                    // prev room should be already empty

                    this.agent.beliefs.declare('in_room ' +p.name + ' ' + new_room); // person_in_room bob bathroom
                    this.agent.beliefs.undeclare('empty '+ new_room);
                    if(prev_room != new_room){
                        this.agent.beliefs.undeclare('in_room ' +p.name + ' ' + prev_room); // person_in_room bob bathroom
                        this.agent.beliefs.declare('empty '+ prev_room);
                    }

                }
            });

            PeopleGoals.push(PeopleGoalPromise)
        }
        yield Promise.all(PeopleGoals)
    }
```

the following is an excerpt of the resulting log file from the execution of the previous code showing how the belief set of the house agent is changed each time the person moves from one room to another.

```
0:06:5  Bob  moved to hallway
house_agent>SensePeopleIntention#1 sense: person Bob moved to hallway from bedroom
hallway_light turned on
electricity consumption 550
house_agent                    Belief changed: in_room Bob hallway
house_agent                    Belief changed: not empty hallway
house_agent                    Belief changed: not in_room Bob bedroom
house_agent                    Belief changed: empty bedroom
house_agent>SenseLightsIntention#2 sense: light hallway_light switched on
house_agent                    Belief changed: light_on hallway_light
house_agent                    Belief changed: not light_off hallway_light
0:06:10 Bob  moved to bathroom
house_agent>SensePeopleIntention#1 sense: person Bob moved to bathroom from hallway
bathroom_light turned on
hallway_light turned off
electricity consumption 560
electricity consumption 550
house_agent                    Belief changed: in_room Bob bathroom
house_agent                    Belief changed: not empty bathroom
house_agent                    Belief changed: not in_room Bob hallway
house_agent                    Belief changed: empty hallway
house_agent>SenseLightsIntention#2 sense: light bathroom_light switched on
house_agent>SenseLightsIntention#2 sense: light hallway_light switched off
house_agent                    Belief changed: light_on bathroom_light
house_agent                    Belief changed: not light_off bathroom_light
house_agent                    Belief changed: not light_on hallway_light
house_agent                    Belief changed: light_off hallway_light
0:06:15 Bob  moved to hallway
house_agent>SensePeopleIntention#1 sense: person Bob moved to hallway from bathroom
hallway_light turned on
bathroom_light turned off
electricity consumption 560
electricity consumption 550
house_agent                    Belief changed: in_room Bob hallway
house_agent                    Belief changed: not empty hallway
house_agent                    Belief changed: not in_room Bob bathroom
house_agent                    Belief changed: empty bathroom
house_agent>SenseLightsIntention#2 sense: light hallway_light switched on
house_agent>SenseLightsIntention#2 sense: light bathroom_light switched off
house_agent                    Belief changed: light_on hallway_light
house_agent                    Belief changed: not light_off hallway_light
house_agent                    Belief changed: not light_on bathroom_light
house_agent                    Belief changed: light_off bathroom_light
```

Vacuum Agent (planning agent)

The vacuum agent is the only agent in the domain that utilize planning to achieve its goal. It was developed as a robotic agent able to autonomously move among all the rooms, clean them, and return to the charging station located in the entrance to always have a full charge. To define the agent,

first we need to define its *domain*. In the domain (contained in the *domain-vacuum_cleaner.pddl* file), we define the predicates (the properties of the object we are interested in) and actions that can be performed by the agent. Then, we need to define the *problem*, so the initial state of the domain, which objects are present in the domain, and describe the goal we want to achieve. The problem definition can be found in the *problem-vacuum_cleaner.pddl* file. The domain an problem files contain all the information needed by the planning agent to solve the planning problem. In the proposed implementation, we use the *OnlinePlanner* (located in the *pddl* folder) to automatically generate the domain and problem files starting from the actions definition contained in the *vacuumCleanerScenario.js* file. The plan will be then obtained by sending a solving request to an online planner (`http://solver.planning.domains/solve`). Both the domain file and the problem file can be found in the /src/myWorld/tmp folder.

The definition of the *domain* is reported in the following code.

```
(define (domain vacuum_cleaner)
    (:requirements :strips)
    (:predicates
        (robot ?vacuum)
        (room ?source)
        (at ?vacuum ?source)
        (connected ?source ?destination)
        (on ?vacuum)
        (dirty ?room)
        (clean ?room)
        (off ?vacuum)
        (charging ?vacuum)
        (discharging ?vacuum)
    )

    (:action Move
        :parameters (?vacuum ?source ?destination)
        :precondition (and
            (robot ?vacuum)
            (room ?source)
            (room ?destination)
            (at ?vacuum ?source)
            (connected ?source ?destination)
            (on ?vacuum)
        )
        :effect (and
            (at ?vacuum ?destination)
            (not (at ?vacuum ?source))
        )
    )

    (:action CleanRoom
        :parameters (?vacuum ?room)
        :precondition (and
            (robot ?vacuum)
            (room ?room)
            (dirty ?room)
            (at ?vacuum ?room)
            (on ?vacuum)
        )
```

```
        :effect (and
            (clean ?room)
            (not (dirty ?room))
        )
    )

    (:action TurnOn
        :parameters (?vacuum)
        :precondition (and
            (robot ?vacuum)
            (off ?vacuum)
            (charging ?vacuum)
        )
        :effect (and
            (on ?vacuum)
            (not (off ?vacuum))
            (discharging ?vacuum)
            (not (charging ?vacuum))
        )
    )

    (:action TurnOff
        :parameters (?vacuum)
        :precondition (and
            (robot ?vacuum)
            (on ?vacuum)
            (discharging ?vacuum)
        )
        :effect (and
            (off ?vacuum)
            (not (on ?vacuum))
            (charging ?vacuum)
            (not (discharging ?vacuum))
        )
    )
)
```

As we can see, some predicates define some basic properties of the vacuum agent, such as: **robot ?vacuum**, **on ?vacuum**, **off ?vacuum**, **charging ?vacuum**, and **discharging ?vacuum**. Others define some basic properties of the room, like **dirty ?room** or **clean ?room**. Some are more complex and are used to localize the robot in the scenario, **at ?vacuum ?source**, or to define the structure of the house indicating which rooms are connected to each other, **connected ?source ?destination**. These predicates are used to define the actions our robotic agent can perform: **TurnOn**, **TurnOff**, **Move**, **CleanRoom**. The Move action simulates the robot moving from one room to another. To do so, the two rooms must be connected, the robot must be in the source room and of course must be turned on (the battery is discharging as a consequence of the robot being turned on). As a result, the robot will be able to move from the source room to the destination room. The CleanRoom action requires that the robot is turned on and that it is in the dirty room in order to clean it. As a result the room is now cleaned. The turn on and turn off actions are respectively used to turn the robot on or off and are part of the prerequisites for being able to perform the other actions.

In the definition of the problem file it was supposed that the agent knows the structure of the

house, so which rooms there are and how they are connected.

As mentioned above, the problem and domain file are generated automatically by calling the Online Planner on top of the actions definition in vacuumClenaerScenario.js file.

The code of vacuumCleanerScenario.js is the following.

```
const pddlActionIntention = require('../pddl/actions/pddlActionIntention')
const Agent = require('../bdi/Agent')
const Goal = require('../bdi/Goal')
const Intention = require('../bdi/Intention')
const PlanningGoal = require('../pddl/PlanningGoal')

class VacuumAction extends pddlActionIntention{

    async checkPreconditionAndApplyEffect (duration) {
        if ( this.checkPrecondition() ) {
            this.applyEffect()
            await new Promise(res=>setTimeout(res,duration))
        }
        else
            throw new Error('pddl precondition not valid'); //Promise is rejected!
    }
}


class Move extends VacuumAction {
    static parameters = ['vacuum','source', 'destination'];
    static precondition = [['robot','vacuum'], ['room', 'source'],
                        ['room', 'destination'],['at','vacuum','source'],
                        ['connected','source','destination'], ['on' ,'vacuum'] ];
    static effect = [['at', 'vacuum','destination'],['not at', 'vacuum','source']];
    *exec ({source, destination}=parameters) {
        this.agent.device.move(destination)
        yield this.checkPreconditionAndApplyEffect(40)
    }
}

class CleanRoom extends VacuumAction {
    static parameters = ['vacuum','room'];
    static precondition = [ ['robot','vacuum'], ['room', 'room'],['dirty', 'room'],
                            ['at','vacuum', 'room'], ['on' ,'vacuum']];
    static effect = [['clean','room'], ['not dirty', 'room']];
    *exec ({room}=parameters) {
        this.agent.device.cleanRoom(room);
        yield this.checkPreconditionAndApplyEffect(90);
    }
}


class TurnOn extends VacuumAction {
    static parameters = ['vacuum'];
    static precondition = [ ['robot','vacuum'], ['off', 'vacuum'],
                            ['charging', 'vacuum']];
    static effect = [['on','vacuum'], ['not off', 'vacuum'],
                        ['discharging', 'vacuum'],
```

```
  ['not charging', 'vacuum']];
    *exec ({}=parameters) {
        this.agent.device.turnOn()
        yield this.checkPreconditionAndApplyEffect(1)
    }
}

class TurnOff extends VacuumAction {
    static parameters = ['vacuum'];
    static precondition = [ ['robot','vacuum'], ['on', 'vacuum'],
                            ['discharging', 'vacuum']];
    static effect = [['off','vacuum'], ['not on', 'vacuum'], ['charging', 'vacuum'],
['not discharging', 'vacuum']];
    *exec ({}=parameters) {
        this.agent.device.turnOff()
        yield this.checkPreconditionAndApplyEffect(1)
    }
}

class RetryGoal extends Goal {}
class RetryFourTimesIntention extends Intention { // replanning behaviour (agent tries up
     to 4 times to achieve goal)
    static applicable (goal) {
        return goal instanceof RetryGoal
    }
    *exec ({goal}=parameters) {
        for(let i=0; i<4; i++) {
            let goalAchieved = yield this.agent.postSubGoal( goal )
            if (goalAchieved)
                return;
            this.log('wait for something to change on beliefset before retrying for the '
                 + (i+2) + 'th time goal', goal.toString())
            yield this.agent.beliefs.notifyAnyChange()
        }
    }
}

module.exports = {Move, CleanRoom, TurnOff, TurnOn, RetryGoal, RetryFourTimesIntention}
```

This code was inspired by the pddl examples seen during the laboratory lectures, in particular the *blocksworldScenario3.js* (located in the *Autonode.js-master/src/blocksworld* folder), which introduced the **world agent** concept. The scenario presented in this example is as follows, some blocks are positioned on a table, and can be picked up or be putted down by a gripper agent. The goal of the gripper agent is to stack the blocks in a certain order. In this example the world agent is used to simulate the environment. It defines how the blocks are disposed on the table and if the gripper agent is holding something or not. The gripper agent will share the knowledge of the environment with the world agent. The idea is that the gripper agent will be able to perform actions in the environment by calling the world agent methods. The world agent actions extend a FakeAction class which allows to use the *checkPreconditionAndApplyEffect* mehtod to, as the name suggest, verify the action preconditions are met and apply the effect to the agent's beliefs. Also here, a plan is generated by calling the Online Planner. In case no plan is found, the gripper agent will wait for something to change in the beliefset, before trying again to achieve the goal (up to

four times) using the *RetryFourTimesIntentions* class.

Since the assignment explicitly asked to call a device method and then sense the effects, this solution was modified. In particular, a class called **VacuumAction** was defined. This class, similarly to the FakeAction class discussed above, extends the **pddlActionIntention** class which contains the **checkPreconditionAndApplyEffect** method. All the vacuum agent actions are then defined as classes extending the VacuumAction class. Each action contains a definition of the parameters, predicates, and effects as well as a method executing it (*exec) which calls the vacuum agent methods (turnOn, turnOff, move, cleanRoom) and also verifies that the action's preconditions are satisfied before applying the effects. In order to connect the agent with the device, the agent definition contained in the *Agent.js* file (located in *src/bdi/*), was modified by including the device property in the constructor. By doing so, the vacuum agent will now be able to access the device methods when executing of the action. In *myScenario.js* the vacuum agent is defined as follows:

```
var vacuum_agent = new Agent('vacuum_cleaner', myHouse.devices.vacuum_cleaner);
```

In *myScenario.js*, the function *init_vacuum_belief()* is used for instantiating the initial beliefs of the vacuum agent, such as where the robot is located at the start of the plan, which rooms are in the house and how those rooms are connected to each other, and it is called as soon as the agent is instantiated.

```
function init_vacuum_belief(){
    // robot declaration
    vacuum_agent.beliefs.declare('robot vacuum')

    // room definition
    vacuum_agent.beliefs.declare('room entrance')
    vacuum_agent.beliefs.declare('room living-room')
    vacuum_agent.beliefs.declare('room kitchen')
    vacuum_agent.beliefs.declare('room hallway')
    vacuum_agent.beliefs.declare('room bathroom')
    vacuum_agent.beliefs.declare('room bedroom')

    // house structure definition
    vacuum_agent.beliefs.declare('connected entrance living-room')
    vacuum_agent.beliefs.declare('connected living-room entrance')

    vacuum_agent.beliefs.declare('connected living-room kitchen')
    vacuum_agent.beliefs.declare('connected kitchen living-room')

    vacuum_agent.beliefs.declare('connected living-room hallway')
    vacuum_agent.beliefs.declare('connected hallway living-room')

    vacuum_agent.beliefs.declare('connected hallway bathroom')
    vacuum_agent.beliefs.declare('connected bathroom hallway')

    vacuum_agent.beliefs.declare('connected hallway bedroom')
    vacuum_agent.beliefs.declare('connected bedroom hallway')

    // def init location vacuum
    vacuum_agent.beliefs.declare('at vacuum entrance')

    //robot initially off
    vacuum_agent.beliefs.declare('off vacuum')
```

```
34
35      // robot is charging at the base station located in the entrance
36      vacuum_agent.beliefs.declare('charging vacuum')
37  }
```

The only information the vacuum agent is missing is knowing which rooms are dirty. In order to make the house agent communicates to the vacuum agent the missing information, the following code was used (also this is a modified code from the *blocksWorldScenario3.js*).

```
1  var dirty_rooms_sensor = (agent) => (value,key,observable) => {
2      let predicate = key.split(' ')[0]
3      let arg1 = key.split(' ')[1]
4      if (predicate=='dirty') // dirty room_name
5              key = 'dirty '+arg1;
6      else
7          return;
8      value?agent.beliefs.declare(key):agent.beliefs.undeclare(key)
9  }
10
11 house_agent.beliefs.observeAny(dirty_rooms_sensor(vacuum_agent))
```

In this way, each time the house agent beliefs change, the vacuum agent will be informed if those beliefs defines a room as dirty. Similarly to the blocksWorld scenario, where the gripper agent was informed about the status of the gripper (holding or empty; here we are interested in the status of the room, which can be either clean or dirty). The status of the room is changed using the *declare_dirty_rooms()* function that declares all rooms as dirty since the resident has used them.

Now the vacuum agent has all the information it needs and can develop the plan for cleaning the house. The log of resulting plan will be shown in the next section that discusses the Vacuum Cleaner Scenario.

Scenarios

The following scenarios simulate the use and scheduling of different devices by the house agent based on the resident behaviour.

Meal Scenario

This scenario is executed when the resident wakes up in the morning or when it comes back home from work either for lunch or for dinner. The heaters usage have been programmed by the user according to his daily routine. Thus, the house agent will turn on only some of the heaters according to the resident desire. For example, the kitchen and bathroom heaters are turned on in the morning when the resident wakes up and are turned off when the resident leaves for work (in the same period of time the security agent will open all the house blinds and unlock the front door). When the resident then comes back home for lunch (around 13), the house agent will turn on and off only the kitchen and living room heaters. Instead, when the resident comes back home for dinner, all the heaters will be turned on. To conclude, at night (at 22) the house agent will make sure that all the heaters are turned off to reduce electricity consumption.

The following is an extract from the *scenario.log* file showing the execution of this scenario when the resident comes back home for lunch.

```
1  0:13:00 Bob  moved to entrance
```

```
house_agent>HeatersIntention#4 turned on kitchen_heater
kitchen_heater  on temperature 18
house_agent>HeatersIntention#4 turned on living_room_heater
living_room_heater  on temperature 18
electricity consumption 1025
electricity consumption 1040
house_agent>SensePeopleIntention#1 sense: person Bob moved to entrance from outdoor
entrance_light turned on
electricity consumption 1050
house_agent                    Belief changed: in_room Bob entrance
house_agent                    Belief changed: not empty entrance
house_agent                    Belief changed: not in_room Bob outdoor
house_agent                    Belief changed: empty outdoor
house_agent>SenseHeatersIntention#3 sense: heater kitchen_heater switched on
house_agent>SenseHeatersIntention#3 sense: heater living_room_heater switched on
house_agent                    Belief changed: on kitchen_heater
house_agent                    Belief changed: not off kitchen_heater
house_agent                    Belief changed: on living_room_heater
house_agent                    Belief changed: not off living_room_heater
0:13:5  Bob  moved to living_room
house_agent>SensePeopleIntention#1 sense: person Bob moved to living_room from entrance
living_room_light turned on
entrance_light turned off
electricity consumption 1060
electricity consumption 1050
house_agent                    Belief changed: in_room Bob living_room
house_agent                    Belief changed: not empty living_room
house_agent                    Belief changed: not in_room Bob entrance
house_agent                    Belief changed: empty entrance
house_agent>SenseLightsIntention#2 sense: light living_room_light switched on
house_agent                    Belief changed: light_on living_room_light
house_agent                    Belief changed: not light_off living_room_light
0:13:10 Bob  moved to kitchen
electricity consumption 1550
house_agent>SensePeopleIntention#1 sense: person Bob moved to kitchen from living_room
kitchen_light turned on
living_room_light turned off
electricity consumption 1560
electricity consumption 1550
house_agent                    Belief changed: in_room Bob kitchen
house_agent                    Belief changed: not empty kitchen
house_agent                    Belief changed: not in_room Bob living_room
house_agent                    Belief changed: empty living_room
house_agent>SenseLightsIntention#2 sense: light kitchen_light switched on
house_agent>SenseLightsIntention#2 sense: light living_room_light switched off
house_agent                    Belief changed: light_on kitchen_light
house_agent                    Belief changed: not light_off kitchen_light
house_agent                    Belief changed: not light_on living_room_light
house_agent                    Belief changed: light_off living_room_light
house_agent>SenseFridgeIntention#5 sense:  fridge empty
firdge EMPTY! Need to buy groceries!
house_agent                    Belief changed: fridge_empty
0:14:00 house_agent>HeatersIntention#4 turned off kitchen_heater
kitchen_heater off
```

```
house_agent>HeatersIntention#4 turned off living_room_heater
living_room_heater off
electricity consumption 1535
electricity consumption 1520
house_agent>SenseHeatersIntention#3 sense: heater kitchen_heater switched off
house_agent>SenseHeatersIntention#3 sense: heater living_room_heater switched off
house_agent                    Belief changed: not on kitchen_heater
house_agent                    Belief changed: off kitchen_heater
house_agent                    Belief changed: not on living_room_heater
house_agent                    Belief changed: off living_room_heater
0:14:20 Bob  moved to living_room
house_agent>SensePeopleIntention#1 sense: person Bob moved to living_room from kitchen
living_room_light turned on
kitchen_light turned off
electricity consumption 1530
electricity consumption 1520
house_agent                    Belief changed: in_room Bob living_room
house_agent                    Belief changed: not empty living_room
house_agent                    Belief changed: not in_room Bob kitchen
house_agent                    Belief changed: empty kitchen
house_agent>SenseLightsIntention#2 sense: light living_room_light switched on
house_agent>SenseLightsIntention#2 sense: light kitchen_light switched off
house_agent                    Belief changed: light_on living_room_light
house_agent                    Belief changed: not light_off living_room_light
house_agent                    Belief changed: not light_on kitchen_light
house_agent                    Belief changed: light_off kitchen_light
0:14:25 Bob  moved to entrance
house_agent>SensePeopleIntention#1 sense: person Bob moved to entrance from living_room
entrance_light turned on
living_room_light turned off
electricity consumption 1530
electricity consumption 1520
house_agent                    Belief changed: in_room Bob entrance
house_agent                    Belief changed: not empty entrance
house_agent                    Belief changed: not in_room Bob living_room
house_agent                    Belief changed: empty living_room
house_agent>SenseLightsIntention#2 sense: light living_room_light switched off
house_agent                    Belief changed: not light_on living_room_light
house_agent                    Belief changed: light_off living_room_light
0:14:30 Bob  moved to outdoor
house_agent                    Belief changed: dirty entrance
house_agent                    Belief changed: dirty living-room
house_agent                    Belief changed: dirty kitchen
house_agent                    Belief changed: dirty hallway
house_agent                    Belief changed: dirty bathroom
house_agent                    Belief changed: dirty bedroom
vacuum_cleaner                 Belief changed: dirty entrance
vacuum_cleaner                 Belief changed: dirty living-room
vacuum_cleaner                 Belief changed: dirty kitchen
vacuum_cleaner                 Belief changed: dirty hallway
vacuum_cleaner                 Belief changed: dirty bathroom
vacuum_cleaner                 Belief changed: dirty bedroom
house_agent>SensePeopleIntention#1 sense: person Bob moved to outdoor from entrance
entrance_light turned off
```

```
electricity consumption 1510
house_agent                    Belief changed: in_room Bob outdoor
house_agent                    Belief changed: not empty outdoor
house_agent                    Belief changed: not in_room Bob entrance
house_agent                    Belief changed: empty entrance
```

Vacuum Cleaner Scenario

The vacuum cleaner agent runs every day from 15 to 18 in the proposed scenario. At this time the house is empty since the resident is at work. So, the agent can fulfill its goal of cleaning the whole house without having to worry about disturbing the resident. The following extract from the *scenario.log* file shows the vacuum agent finding and executing the plan.

```
0:15:00 vacuum_cleaner                 Trying to use intention RetryFourTimesIntention to
     achieve goal {RetryGoal#13:{goal:{PddlGoal#12:{goal:[["clean entrance"],["clean
    living-room"],["clean kitchen"],["clean hallway"],["clean bedroom"],["clean bathroom
    "],["at vacuum entrance"],["off vacuum"]]}}}}
vacuum_cleaner>RetryFourTimesIntention#12 Intention started
vacuum_cleaner                 Trying to use intention OnlinePlanning to achieve goal {
    PddlGoal#12:{goal:[["clean entrance"],["clean living-room"],["clean kitchen"],["clean
     hallway"],["clean bedroom"],["clean bathroom"],["at vacuum entrance"],["off vacuum
    "]]}}
vacuum_cleaner>OnlinePlanning#13 Intention started
0:15:40 vacuum_cleaner>OnlinePlanning#13 Plan found:
vacuum_cleaner>OnlinePlanning#13 - (turnon vacuum)
vacuum_cleaner>OnlinePlanning#13 - (cleanroom vacuum entrance)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum entrance living-room)
vacuum_cleaner>OnlinePlanning#13 - (cleanroom vacuum living-room)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum living-room kitchen)
vacuum_cleaner>OnlinePlanning#13 - (cleanroom vacuum kitchen)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum kitchen living-room)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum living-room hallway)
vacuum_cleaner>OnlinePlanning#13 - (cleanroom vacuum hallway)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum hallway bathroom)
vacuum_cleaner>OnlinePlanning#13 - (cleanroom vacuum bathroom)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum bathroom hallway)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum hallway bedroom)
vacuum_cleaner>OnlinePlanning#13 - (cleanroom vacuum bedroom)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum bedroom hallway)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum hallway living-room)
vacuum_cleaner>OnlinePlanning#13 - (move vacuum living-room entrance)
vacuum_cleaner>OnlinePlanning#13 - (turnoff vacuum)
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (TurnOn vacuum) Effect: on
    vacuum,not off vacuum,discharging vacuum,not charging vacuum
vacuum_cleaner>TurnOn#14       Intention started
turn on vacuum cleaner
electricity consumption 920
vacuum_cleaner                 Belief changed: on vacuum
vacuum_cleaner                 Belief changed: not off vacuum
vacuum_cleaner                 Belief changed: discharging vacuum
vacuum_cleaner                 Belief changed: not charging vacuum
0:15:45 vacuum_cleaner>TurnOn#14       Intention success
```

```
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (CleanRoom vacuum entrance)
    Effect: clean entrance,not dirty entrance
vacuum_cleaner>CleanRoom#15    Intention started
cleaning entrance
vacuum_cleaner                 Belief changed: clean entrance
vacuum_cleaner                 Belief changed: not dirty entrance
0:15:55 vacuum_cleaner>CleanRoom#15    Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum entrance living-
    room) Effect: at vacuum living-room,not at vacuum entrance
vacuum_cleaner>Move#16         Intention started
vacuum moved from entrance to living-room
vacuum_cleaner                 Belief changed: at vacuum living-room
vacuum_cleaner                 Belief changed: not at vacuum entrance
0:16:5  vacuum_cleaner>Move#16         Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (CleanRoom vacuum living-room)
    Effect: clean living-room,not dirty living-room
vacuum_cleaner>CleanRoom#17    Intention started
cleaning living-room
vacuum_cleaner                 Belief changed: clean living-room
vacuum_cleaner                 Belief changed: not dirty living-room
0:16:15 vacuum_cleaner>CleanRoom#17    Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum living-room
    kitchen) Effect: at vacuum kitchen,not at vacuum living-room
vacuum_cleaner>Move#18         Intention started
vacuum moved from living-room to kitchen
vacuum_cleaner                 Belief changed: at vacuum kitchen
vacuum_cleaner                 Belief changed: not at vacuum living-room
0:16:20 vacuum_cleaner>Move#18         Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (CleanRoom vacuum kitchen)
    Effect: clean kitchen,not dirty kitchen
vacuum_cleaner>CleanRoom#19    Intention started
cleaning kitchen
vacuum_cleaner                 Belief changed: clean kitchen
vacuum_cleaner                 Belief changed: not dirty kitchen
0:16:30 vacuum_cleaner>CleanRoom#19    Intention success
0:16:35 vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum kitchen
    living-room) Effect: at vacuum living-room,not at vacuum kitchen
vacuum_cleaner>Move#20         Intention started
vacuum moved from kitchen to living-room
vacuum_cleaner                 Belief changed: at vacuum living-room
vacuum_cleaner                 Belief changed: not at vacuum kitchen
0:16:40 vacuum_cleaner>Move#20         Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum living-room
    hallway) Effect: at vacuum hallway,not at vacuum living-room
vacuum_cleaner>Move#21         Intention started
vacuum moved from living-room to hallway
vacuum_cleaner                 Belief changed: at vacuum hallway
vacuum_cleaner                 Belief changed: not at vacuum living-room
0:16:45 vacuum_cleaner>Move#21         Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (CleanRoom vacuum hallway)
    Effect: clean hallway,not dirty hallway
vacuum_cleaner>CleanRoom#22    Intention started
cleaning hallway
vacuum_cleaner                 Belief changed: clean hallway
```

```
vacuum_cleaner                  Belief changed: not dirty hallway
0:16:55 vacuum_cleaner>CleanRoom#22    Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum hallway bathroom)
    Effect: at vacuum bathroom,not at vacuum hallway
vacuum_cleaner>Move#23          Intention started
vacuum moved from hallway to bathroom
vacuum_cleaner                  Belief changed: at vacuum bathroom
vacuum_cleaner                  Belief changed: not at vacuum hallway
0:17:00 vacuum_cleaner>Move#23          Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (CleanRoom vacuum bathroom)
    Effect: clean bathroom,not dirty bathroom
vacuum_cleaner>CleanRoom#24    Intention started
cleaning bathroom
vacuum_cleaner                  Belief changed: clean bathroom
vacuum_cleaner                  Belief changed: not dirty bathroom
0:17:10 vacuum_cleaner>CleanRoom#24    Intention success
0:17:15 vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum bathroom
    hallway) Effect: at vacuum hallway,not at vacuum bathroom
vacuum_cleaner>Move#25          Intention started
vacuum moved from bathroom to hallway
vacuum_cleaner                  Belief changed: at vacuum hallway
vacuum_cleaner                  Belief changed: not at vacuum bathroom
0:17:20 vacuum_cleaner>Move#25          Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum hallway bedroom)
    Effect: at vacuum bedroom,not at vacuum hallway
vacuum_cleaner>Move#26          Intention started
vacuum moved from hallway to bedroom
vacuum_cleaner                  Belief changed: at vacuum bedroom
vacuum_cleaner                  Belief changed: not at vacuum hallway
0:17:25 vacuum_cleaner>Move#26          Intention success
0:17:30 vacuum_cleaner>OnlinePlanning#13 Starting sequential step (CleanRoom vacuum
    bedroom) Effect: clean bedroom,not dirty bedroom
vacuum_cleaner>CleanRoom#27    Intention started
cleaning bedroom
vacuum_cleaner                  Belief changed: clean bedroom
vacuum_cleaner                  Belief changed: not dirty bedroom
0:17:35 vacuum_cleaner>CleanRoom#27    Intention success
0:17:40 vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum bedroom
    hallway) Effect: at vacuum hallway,not at vacuum bedroom
vacuum_cleaner>Move#28          Intention started
vacuum moved from bedroom to hallway
vacuum_cleaner                  Belief changed: at vacuum hallway
vacuum_cleaner                  Belief changed: not at vacuum bedroom
0:17:45 vacuum_cleaner>Move#28          Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum hallway living-
    room) Effect: at vacuum living-room,not at vacuum hallway
vacuum_cleaner>Move#29          Intention started
vacuum moved from hallway to living-room
vacuum_cleaner                  Belief changed: at vacuum living-room
vacuum_cleaner                  Belief changed: not at vacuum hallway
0:17:55 vacuum_cleaner>Move#29          Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (Move vacuum living-room
    entrance) Effect: at vacuum entrance,not at vacuum living-room
vacuum_cleaner>Move#30          Intention started
```

```
vacuum moved from living-room to entrance
vacuum_cleaner                  Belief changed: at vacuum entrance
vacuum_cleaner                  Belief changed: not at vacuum living-room
0:18:00 vacuum_cleaner>Move#30          Intention success
vacuum_cleaner>OnlinePlanning#13 Starting sequential step (TurnOff vacuum) Effect: off
    vacuum,not on vacuum,charging vacuum,not discharging vacuum
vacuum_cleaner>TurnOff#31       Intention started
turn off vacuum cleaner
electricity consumption 1010
vacuum_cleaner                  Belief changed: off vacuum
vacuum_cleaner                  Belief changed: not on vacuum
vacuum_cleaner                  Belief changed: charging vacuum
vacuum_cleaner                  Belief changed: not discharging vacuum
0:18:5  vacuum_cleaner>TurnOff#31       Intention success
vacuum_cleaner>OnlinePlanning#13 Intention success
vacuum_cleaner                  Succesfully used intention OnlinePlanning to achieve goal
    {PddlGoal#12:{goal:[["clean entrance"],["clean living-room"],["clean kitchen"],["
    clean hallway"],["clean bedroom"],["clean bathroom"],["at vacuum entrance"],["off
    vacuum"]]}}
vacuum_cleaner>RetryFourTimesIntention#12 Intention success
vacuum_cleaner                  Succesfully used intention RetryFourTimesIntention to
    achieve goal {RetryGoal#13:{goal:{PddlGoal#12:{goal:[["clean entrance"],["clean
    living-room"],["clean kitchen"],["clean hallway"],["clean bedroom"],["clean bathroom
    "],["at vacuum entrance"],["off vacuum"]]}}}}
```

Source Code Organization

All the discussed code is contained in the */src/myWorld* folder.

The following files have been implemented to handle the house definition in an easier and more organized way.

- **House** (*House.js*): contains the description of the house scenario, including *People*, *Rooms*, *Devices*, and *Utilities*. The utilities only model the electricity consumption, which is simply defined as an **Observable**.
- **House devices** (*HouseDevices.js*): this file contains the class definition for each device in the house.
- **Person** (*Person.js*): this class is used to model the resident. The *in_room* property is used to model the room in which the resident is at a given time. The only method in this class is the *moveTo(to_room)*, which allows the resident to move around the house and also updated the house agents belief set.
- **Scenario** (*myScenario.js*: this file contains the simulation of the smart house.

  Note that, in this scenario, in order to condense in a single log file the complete simulation of the smart house, all house devices were used. This simulation may result not very realistic. For example, since the resident is at work for most of the day, the house should be quite clean. Thus, it does not make much sense to schedule the vacuum agent to run every day. It may be more realistic to schedule the vacuum cleaner to run once per week (i.e. on Thursday). This may also help to save electricity. The same thing may be true also for the other devices, such as the washing machine or the dishwasher. For this reason, a more realistic (fixed) schedule can be found in the *weeklyScenario.js* file.

Each device in the house has its own sensor.

- **Sensors**:
  - **LightSensor.js**: updates the beliefs of the house agent about each light in the house, also updates the electricity consumption when a light is turned on or off.
  - **BlindsSensor.js**: updates the beliefs of the security agent about each blinds in the house.
  - **DoorLockSensor.js**: detects when the status of the front door lock changes and informs the security agent.
  - **DishWasherSensor.js**: informs the house agent when the dishwasher is turned on or off. It is also used by the house agent to keep track of the dishwasher electricity consumption.
  - **FridgeSensor.js**: keeps track of the supplies shortages, it is also used by the house agent to keep track of the fridge electricity consumption.
  - **HeaterSensor.js**: similar functioning as the light sensor, informs the house agent when a heater has been turned on and updates its electricity consumption.
  - **PersonSensor.js**: detects when the resident is moving in the house. Using this information the house agent is able to automatically turn on or off the room light following the resident movements.
  - **WashingMachineSensor.js**: as for the dishwasher sensor, is used to inform the house agent if the the washing machine has been turned on, it updates the house agent belief set and it is also used by the house agent for keeping track of the washing machine electricity consumption.

Some agents have some specific goals, modeled using *intentions*.

- **Intentions**:
  - **BlindsManager.js**: intention used by the security agent to open all the blinds in the morning and close all the blinds at night.
  - **DoorLockManager.js**: intention used by the security agent to lock the front door at night and to unlock the front door in the morning when the resident leaves for work.
  - **HeatersManager.js**: intention used by the house agent to turn on and off the house heaters according to the resident schedule.
  - **vacuumCleanerScenario.js**: implements the planning scenario for the vacuum agent.

All the files related to the pddl part can be found in the **/myWorld/pddl** folder, while all the files defining the agent BDI architecture are in the **/myWorld/bdi** folder.