Announcement Finder User Guide

Eric Tsang

August 30, 2016

1 Table of Contents

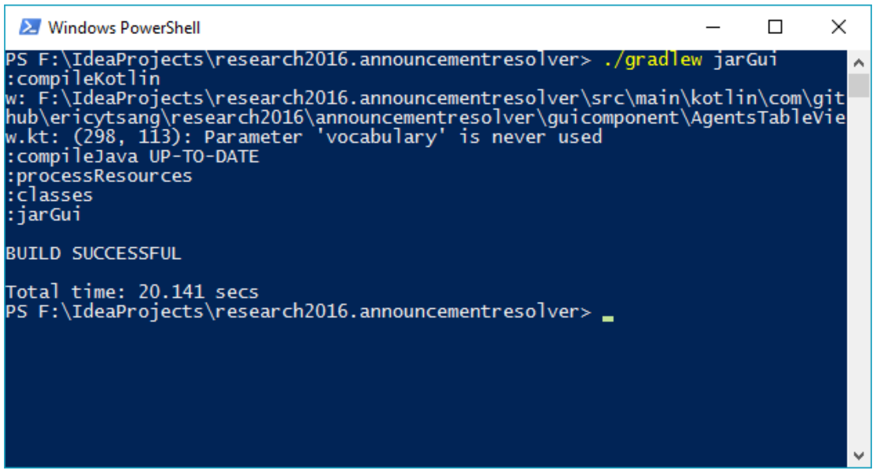
1 Table of Contents 2
2 Procedures 3
2.1 Obtain, Compile & Run the Project 3
2.2 Example 1: Finding an Announcement 4
2.3 Example 2: Changing Agent Behaviors with Announcements 5
3 Demonstrations 8
3.1 Hamming Distance 8
3.2 Weighted Hamming Distance 9
3.3 Patrol 10
3.4 Protect the VIP 11
4 Objects 12
4.1 Obstacle 12
4.2 Definition 12
4.3 Agent 12
4.3.1 Agent Behavior Selection 13
5 Editable Table View 14
6 Windows 15
6.1 Announcement Finder Window 16
6.1.1 File Menu 16
6.1.2 Window Menu 16
6.1.3 Agents Table 17
6.1.4 Display Mode Drop-Down Menu 20
6.1.5 Cancel, Find Announcement and Commit Buttons 21
6.2 Definitions Window 22
6.2.1 File Menu 22
6.2.2 Definitions Table 23
6.3 Obstacles Window 25
6.3.1 File Menu 26
6.3.2 Obstacles Table 26
6.3.3 Obstacles Table Input Dialog 26
6.4 Simulation Window 27
6.4.1 File Menu 27
6.4.2 Simulation 27
7 Propositional Logic Sentence Input Syntax 28
7.1 Operands 28
7.2 Operators 28
7.3 Examples 28

2 Procedures

This section contains a few tutorials for obtaining, running and familiarizing yourself with the announcement finder program.

2.1 Obtain, Compile & Run the Project

1. Download a `.zip` of the project from the link below and extract it:
   - https://github.com/ericytsang/research2016.announcementresolver/archive/master.zip
2. Open a terminal in the root directory of the project.
3. Make sure that both the `java` and `javac` commands are version `1.8.0_91` or greater. The latest versions of the JDK can be found at the following link:
   - http://www.oracle.com/technetwork/java/javase/downloads/index.html
4. To compile an executable JAR file, enter `./gradlew jarGui` into the terminal:

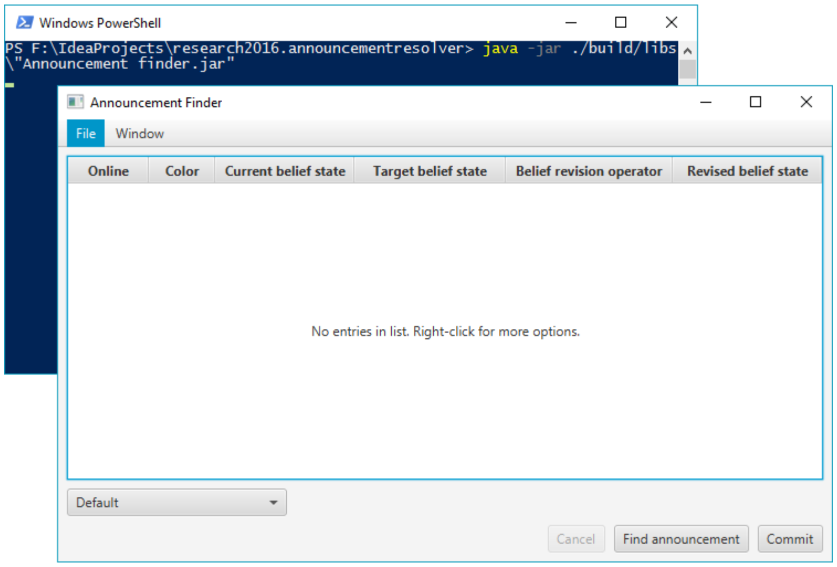


```
PS F:\IdeaProjects\research2016.announcementresolver> ./gradlew jarGui
:compileKotlin
w: F:\IdeaProjects\research2016.announcementresolver\src\main\kotlin\com\git
hub\ericytsang\research2016\announcementresolver\guicomponent\AgentsTableVie
w.kt: (298, 113): Parameter 'vocabulary' is never used
:compileJava UP-TO-DATE
:processResources
:classes
:jarGui

BUILD SUCCESSFUL

Total time: 20.141 secs
PS F:\IdeaProjects\research2016.announcementresolver>
```

5. To execute the compiled JAR file, enter `java -jar ./build/libs/"Announcement finder.jar"` into the terminal:

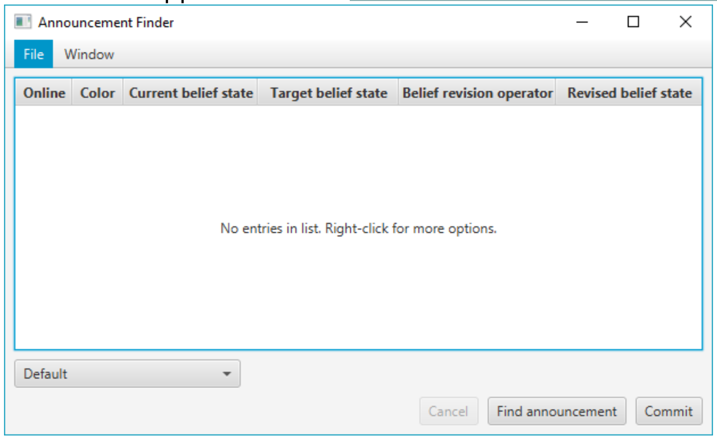


```
PS F:\IdeaProjects\research2016.announcementresolver> java -jar ./build/libs
\"Announcement finder.jar"
```

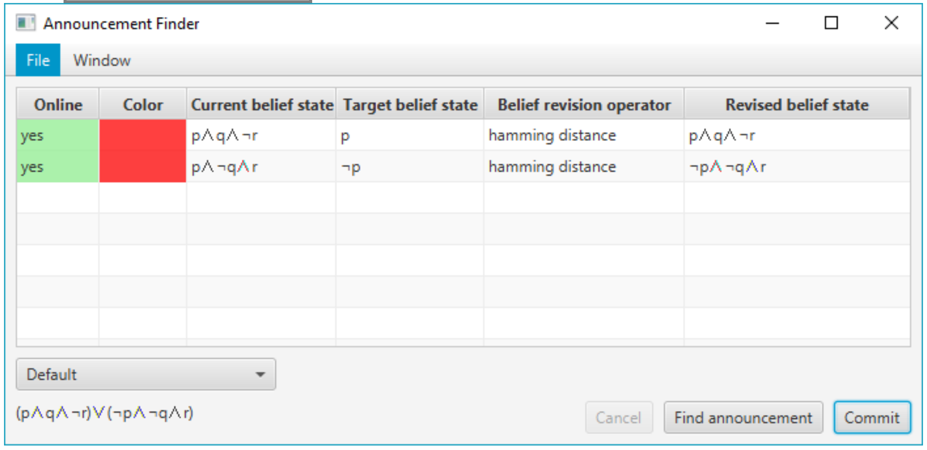
No entries in list. Right-click for more options.

2.2 Example 1: Finding an Announcement

1. Execute the application. The Announcement Finder Window (6.1) should appear:



2. Right click the Agents Table (6.1.3) then press *Add*. An Agents Table Input Dialog (6.1.3.1) should appear:
3. In the Agents Table Input Dialog (6.1.3.1):
    a. Enter `p and q and -r` into the *Current Belief State* field.
    b. Enter `p` into the *Target Belief State* field.
    c. Leave everything else as their default values.
    d. Click OK.
4. Right click the Agents Table (6.1.3) then press *Add* again. An Agents Table Input Dialog (6.1.3.1) should appear:
5. In the Agents Table Input Dialog (6.1.3.1):
    a. Enter `p and -q and r` into the *Current Belief State* field.
    b. Enter `-p` into the *Target Belief State* field.
    c. Leave everything else as their default values.
    d. Click OK.
6. Click Find announcement in the Announcement Finder Window (6.1).



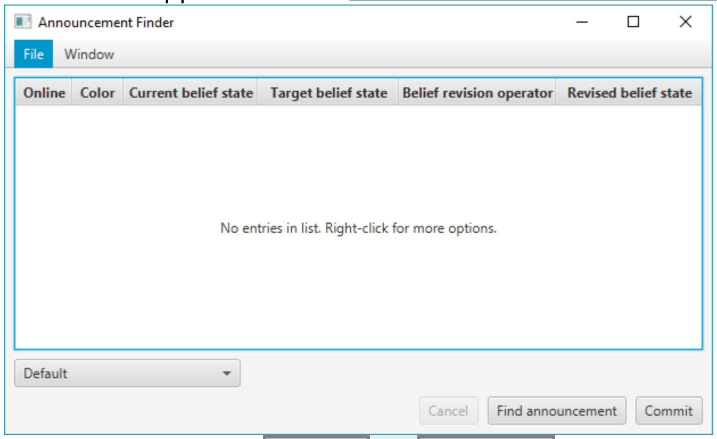
Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟥	p∧q∧¬r	p	hamming distance	p∧q∧¬r
yes	🟥	p∧¬q∧r	¬p	hamming distance	¬p∧¬q∧r

(p∧q∧¬r)∨(¬p∧¬q∧r)

    a. An announcement should appear in the bottom left corner of the Announcement Finder Window (6.1)
    b. A preview of the revised belief states of each agent will appear in the *Revised belief state* column.
7. Click Commit to have each agent adopt its revised belief state as its current belief state.

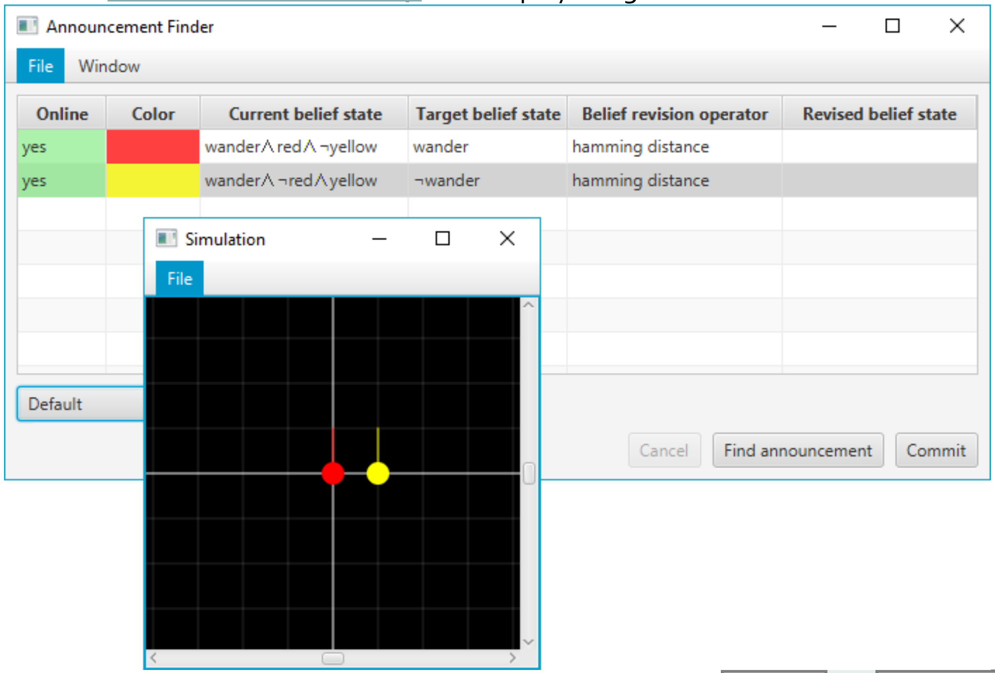
2.3 Example 2: Changing Agent Behaviors with Announcements

1. Execute the application. The Announcement Finder Window (6.1) should appear:



2. In the menu bar, click Window > Simulation to open the Simulation Window (6.4).
3. Go back to the Announcement Finder Window (6.1) and right click the Agents Table (6.1.3) then press *Add*. An Agents Table Input Dialog (6.1.3.1) should appear:
4. In the Agents Table Input Dialog (6.1.3.1):
   a. Enter `wander and red and -yellow` into the *Current Belief State* field.
   b. Enter `wander` into the *Target Belief State* field.
   c. Leave everything else as their default values.
   d. Click OK.
5. Right click the Agents Table (6.1.3) then press *Add* again. An Agents Table Input Dialog (6.1.3.1) should appear:
6. In the Agents Table Input Dialog (6.1.3.1):
   a. Enter `wander and -red and yellow` into the *Current Belief State* field.
   b. Enter `-wander` into the *Target Belief State* field.
   c. Set the *Agent color* to *yellow*.
   d. Set *X Position* to `1`.
   e. Leave everything else as their default values.
   f. Click OK.

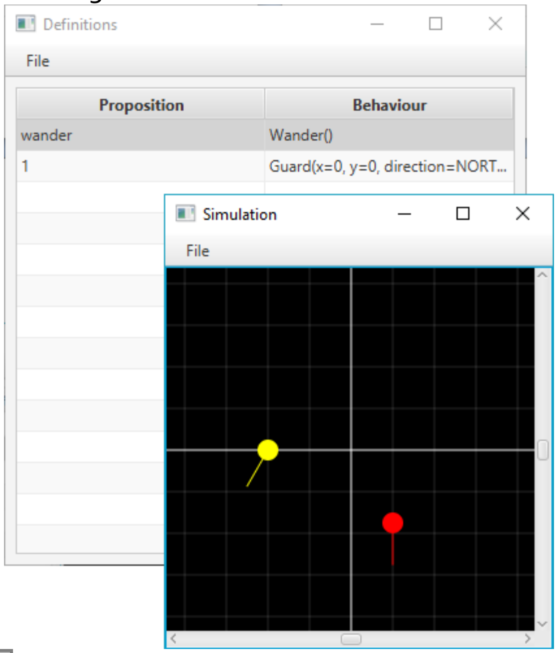
7. Notice that the Simulation Window (6.4) now displays 2 agents:



Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟥	wander∧red∧¬yellow	wander	hamming distance	
yes	🟨	wander∧¬red∧yellow	¬wander	hamming distance	

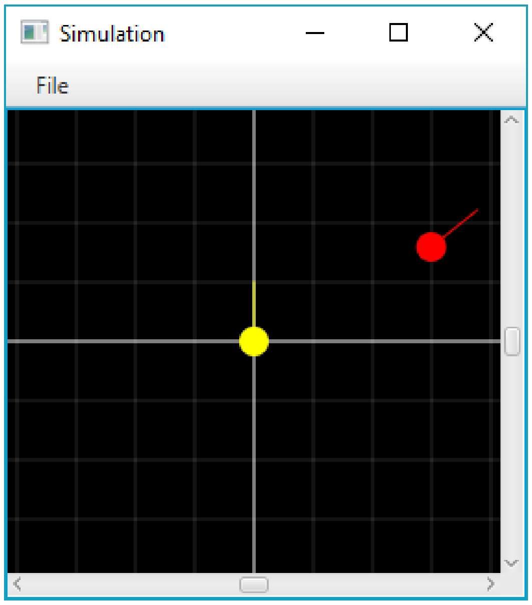
8. In the menu bar of the Announcement Finder Window (6.1), click Window > Definitions to open the Definitions Window (6.2).
9. Right click the Definitions Table (6.2.2) then click *Add*. A Definitions Table Input Dialog (6.2.2.1) should appear.
10. In the Definitions Table Input Dialog (6.2.2.1):
    a. Enter `wander` in the *Proposition* field.
    b. Set the *Behavior drop-down menu* to `Wander`.
    c. Click OK.
11. Right click the first item in the Definitions Table (6.2.2) again then click *Add*. A Definitions Table Input Dialog (6.2.2.1) should appear.
12. In the Definitions Table Input Dialog (6.2.2.1):
    a. Enter `1` in the *Proposition* field.
    b. Set the *Behavior drop-down menu* to `Guard`.
    c. Set *X position* to `0`.
    d. Set *Y position* to `0`.
    e. Set *Direction* to `NORTH`.
    f. Click OK.

13. Now, agents whose belief state satisfies the proposition wander will wander around in the simulation. The rest of the agents will stand guard in the center of the simulation. At this point, both agents should be wandering around.



Proposition	Behaviour
wander	Wander()
1	Guard(x=0, y=0, direction=NORT...

14. Click Find announcement in the Announcement Finder Window (6.1).
    a. An announcement should appear in the bottom left corner of the Announcement Finder Window (6.1)
    b. A preview of the revised belief states of each agent will appear in the *Revised belief state* column.
15. Click Commit to have each agent adopt its revised belief state as its current belief state. Now the yellow agent should return to the center of the simulation.



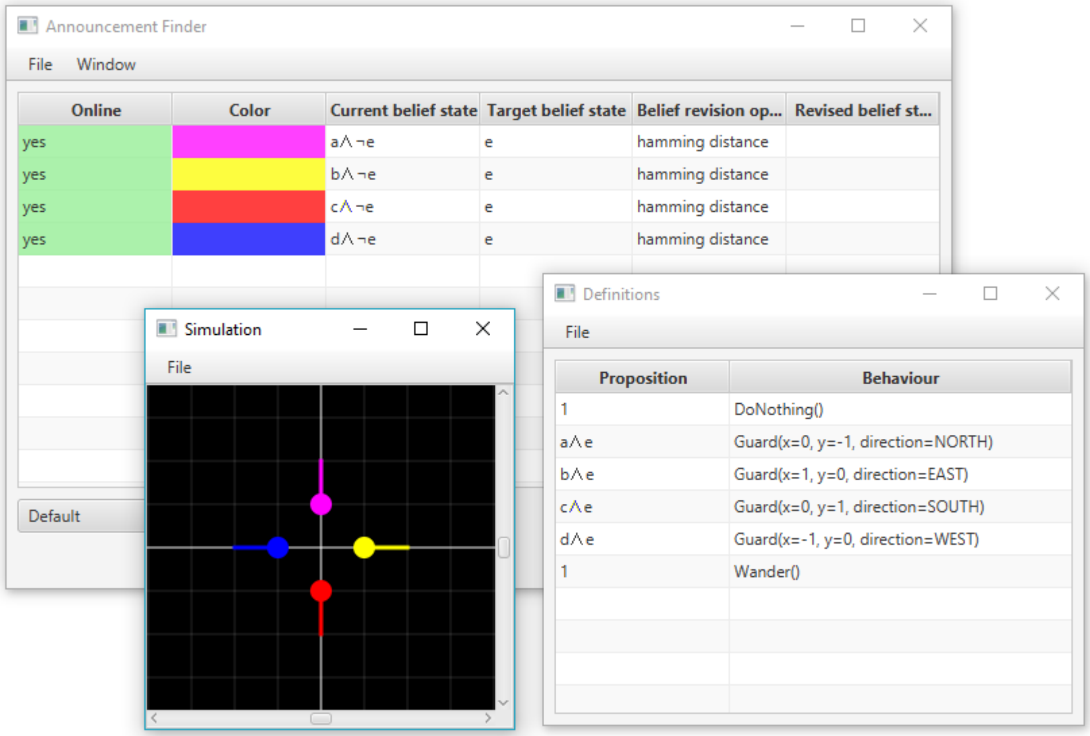
3 Demonstrations

This section contains examples for demonstration purposes. All save files mentioned in the examples are located in `[rootProjectDirectory]/demo`.

3.1 Hamming Distance

This is a simple example that uses hamming distance as the belief revision operator for all agents. This example is best viewed with the Display Mode Drop-Down Menu (6.1.4) set to *Disjunctive Normal Form*.

1. Open the Definitions Window (6.2) and Simulation Window (6.4) via the Window Menu (6.1.2) in the Announcement Finder Window (6.1) .
2. In the Definitions Window (6.2), select File > Load. Then select the `[rootProjectDirectory]/demo/hammingDistance.definitions` save file to load.
3. In the Announcement Finder Window (6.1), select File > Load. Then select the `[rootProjectDirectory]/demo/hammingDistance.agents` save file to load.



Online	Color	Current belief state	Target belief state	Belief revision op...	Revised belief st...
yes	🟪	a∧¬e	e	hamming distance	
yes	🟨	b∧¬e	e	hamming distance	
yes	🟥	c∧¬e	e	hamming distance	
yes	🟦	d∧¬e	e	hamming distance	

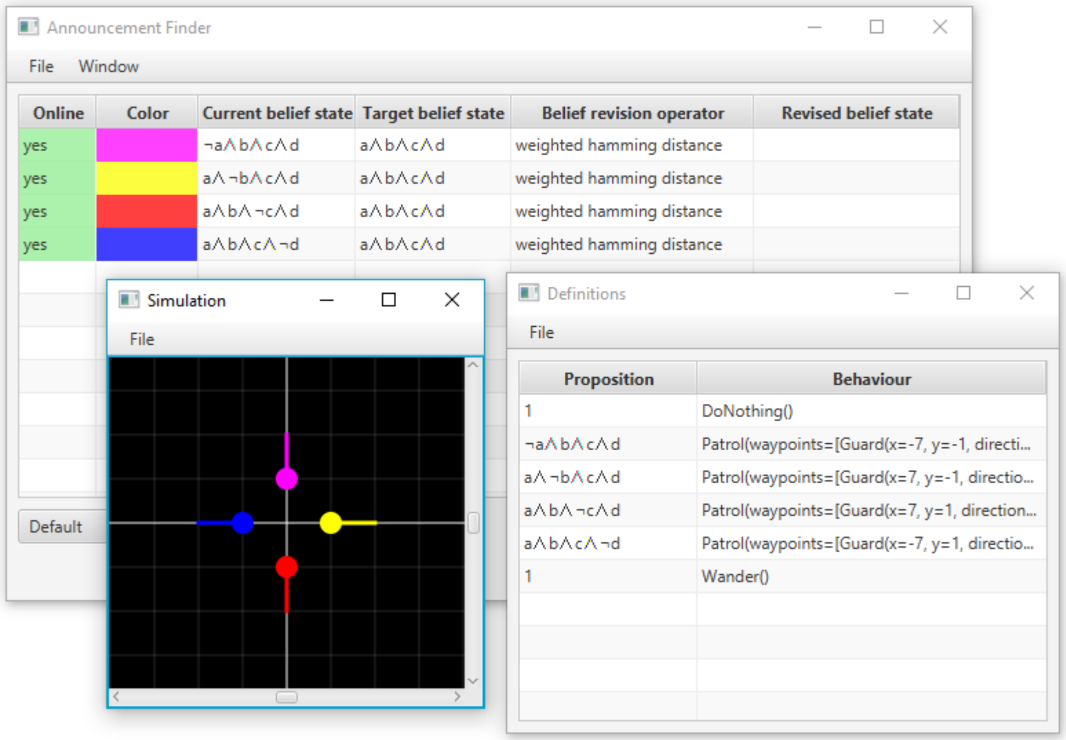
Proposition	Behaviour
1	DoNothing()
a∧e	Guard(x=0, y=-1, direction=NORTH)
b∧e	Guard(x=1, y=0, direction=EAST)
c∧e	Guard(x=0, y=1, direction=SOUTH)
d∧e	Guard(x=-1, y=0, direction=WEST)
1	Wander()

4. Select and delete the first item in the Definitions Table (6.2.2). The agents in the simulator will begin wandering around.
5. In the Announcement Finder Window (6.1), click Find announcement then Commit. The agents will return to the center of the map.
6. You can modify the target belief states of every agent to ¬e, then repeat step 5 to have the agents begin wandering about again.

3.2 Weighted Hamming Distance

This is a simple example that uses hamming distance as the belief revision operator for all agents.

1. Open the Definitions Window (6.2) and Simulation Window (6.4) via the Window Menu (6.1.2) in the Announcement Finder Window (6.1) .
2. In the Definitions Window (6.2), select File > Load. Then select the `[rootProjectDirectory]/demo/weightedHammingDistance.definitions` save file to load.
3. In the Announcement Finder Window (6.1), select File > Load. Then select the `[rootProjectDirectory]/demo/weightedHammingDistance.agents` save file to load.



Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟪	¬a∧b∧c∧d	a∧b∧c∧d	weighted hamming distance	
yes	🟨	a∧¬b∧c∧d	a∧b∧c∧d	weighted hamming distance	
yes	🟥	a∧b∧¬c∧d	a∧b∧c∧d	weighted hamming distance	
yes	🟦	a∧b∧c∧¬d	a∧b∧c∧d	weighted hamming distance	

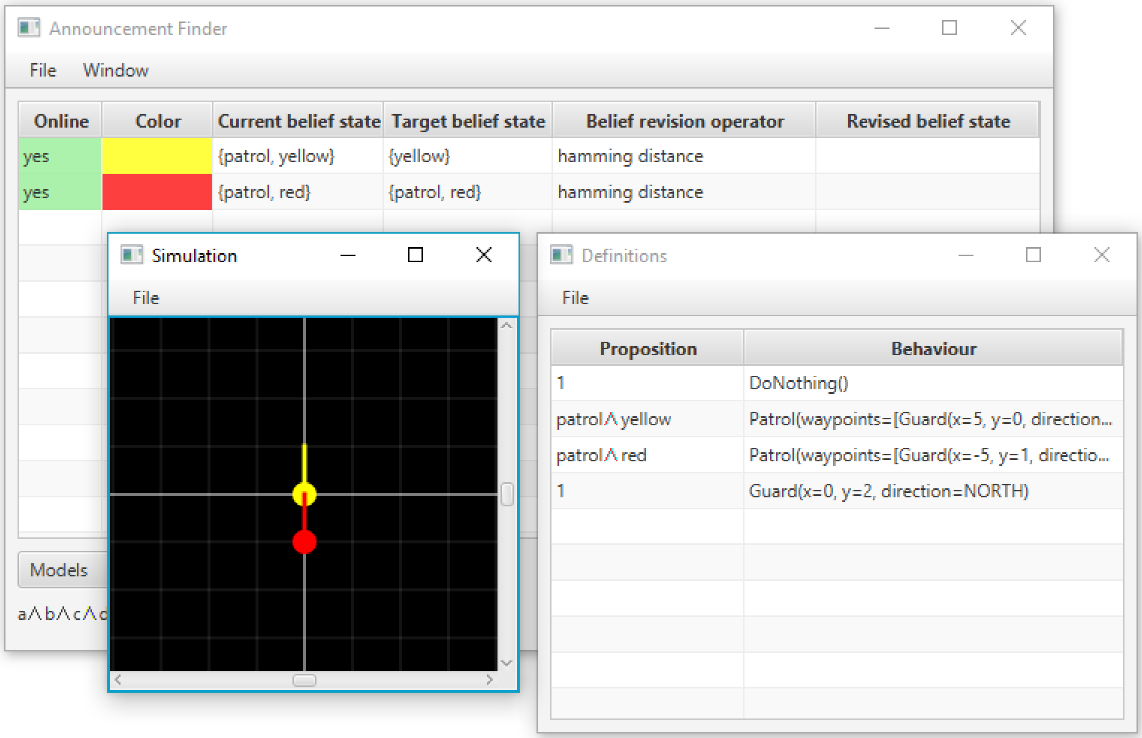
Proposition	Behaviour
1	DoNothing()
¬a∧b∧c∧d	Patrol(waypoints=[Guard(x=-7, y=-1, directi...
a∧¬b∧c∧d	Patrol(waypoints=[Guard(x=7, y=-1, directio...
a∧b∧¬c∧d	Patrol(waypoints=[Guard(x=7, y=1, direction...
a∧b∧c∧¬d	Patrol(waypoints=[Guard(x=-7, y=1, directio...
1	Wander()

4. Select and delete the first item in the Definitions Table (6.2.2). The agents in the simulator will begin patrolling.
5. In the Announcement Finder Window (6.1), click Find announcement then Commit. The agents start wandering.
6. At this point, you can replace the target belief states of the agents with their original belief states prior to the announcement (`-a and b and c and d`, `a and -b and c and d`, `a and b and -c and d`, `a and b and c and -d` respectively) and repeat step 5 to have the agents start patrolling again.

3.3 Patrol

This is a simple example that uses hamming distance as the belief revision operator for all agents. This example is best viewed with the Display Mode Drop-Down Menu (6.1.4) set to *Models*.

1. Open the Definitions Window (6.2) and Simulation Window (6.4) via the Window Menu (6.1.2) in the Announcement Finder Window (6.1) .
2. In the Definitions Window (6.2), select File > Load. Then select the `[rootProjectDirectory]/demo/patrol.definitions` save file to load.
3. In the Announcement Finder Window (6.1), select File > Load. Then select the `[rootProjectDirectory]/demo/patrol.agents` save file to load.



Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟨	{patrol, yellow}	{yellow}	hamming distance	
yes	🟥	{patrol, red}	{patrol, red}	hamming distance	

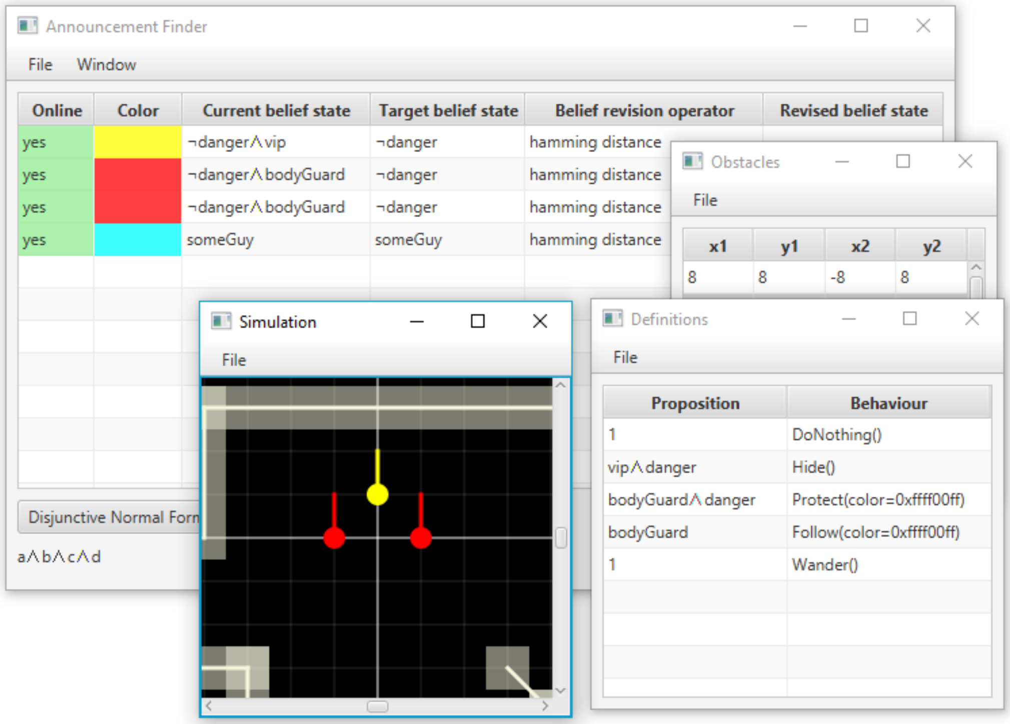
Proposition	Behaviour
1	DoNothing()
patrol∧yellow	Patrol(waypoints=[Guard(x=5, y=0, direction...
patrol∧red	Patrol(waypoints=[Guard(x=-5, y=1, directio...
1	Guard(x=0, y=2, direction=NORTH)

4. Select and delete the first item in the Definitions Table (6.2.2). The agents in the simulator will begin patrolling.
5. In the Announcement Finder Window (6.1), click Find announcement then Commit. The yellow agent will stop patrolling and stand guard at `[0,2]`.
6. You can change the target belief state of both agents and set both to patrol to true or both to false or any combination of truth values. The announcement finder will be able to find an announcement in any of these cases.

3.4 Protect the VIP

This is a simple example that uses hamming distance as the belief revision operator for all agents. This example is best viewed with the Display Mode Drop-Down Menu (6.1.4) set to *Disjunctive Normal Form*.

1. Open the Definitions Window (6.2), Obstacles Window (6.3) and Simulation Window (6.4) via the Window Menu (6.1.2) in the Announcement Finder Window (6.1) .
2. In the Definitions Window (6.2), select File > Load. Then select the `[rootProjectDirectory]/demo/protectVip.definitions` save file to load.
3. In the Announcement Finder Window (6.1), select File > Load. Then select the `[rootProjectDirectory]/demo/ protectVip.agents` save file to load.
4. In the Obstacles Window (6.3), select File > Load. Then select the `[rootProjectDirectory]/demo/ protectVip.obstacles` save file to load.



Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟨	¬danger∧vip	¬danger	hamming distance	
yes	🟥	¬danger∧bodyGuard	¬danger	hamming distance	
yes	🟥	¬danger∧bodyGuard	¬danger	hamming distance	
yes	🟦	someGuy	someGuy	hamming distance	

Disjunctive Normal Form

a∧b∧c∧d

x1	y1	x2	y2
8	8	-8	8

Proposition	Behaviour
1	DoNothing()
vip∧danger	Hide()
bodyGuard∧danger	Protect(color=0xffff00ff)
bodyGuard	Follow(color=0xffff00ff)
1	Wander()

5. Select and delete the first item in the Definitions Table (6.2.2). The yellow and cyan agents will begin wandering the map. The red agents will begin following the yellow agent around.
6. In the Announcement Finder Window (6.1), click Find announcement then Commit. The yellow agent will flee to a corner, and the red agents will stick close by it.
7. You can change the target belief states of the yellow and red agents to be either `danger` or `-danger`. The announcement finder will be able to find an announcement for these cases.

4 Objects

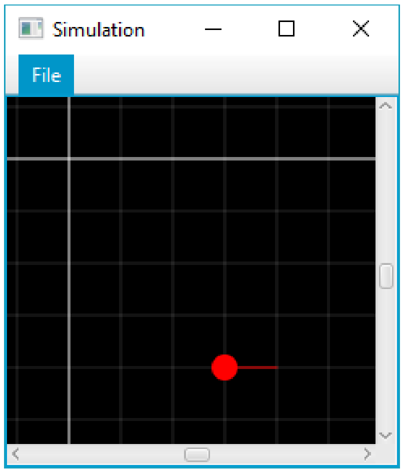
4.1 Obstacle

An Obstacle (4.1) consists of two points that are connected by a line. Agents (4.3) cannot pass through Obstacles (4.1) (i.e. the lines). They must go around them instead.

4.2 Definition

A Definition (4.2) is an association of a propositional logic sentence with an agent behavior.

4.3 Agent

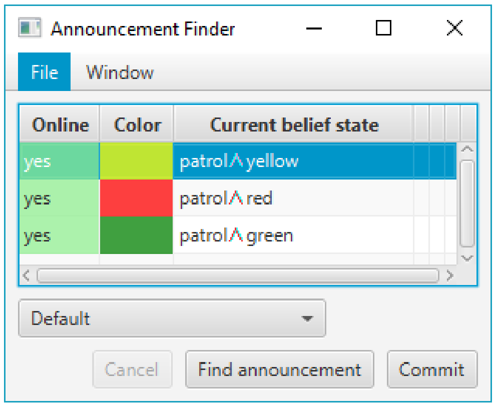


An Agent (4.3) has a position, direction, color, belief state and belief revision operator. The image to the left shows the Simulation Window (6.4) with a red agent at position 3, 4 facing east.

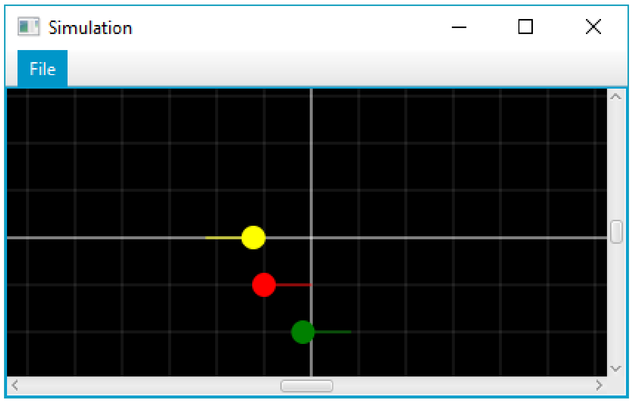
The Agent (4.3) will exhibit the behavior of the first Definition (4.2) in the Definitions Table (6.2.2) that its belief state satisfies. See Agent Behavior Selection (4.3.1) for more details.

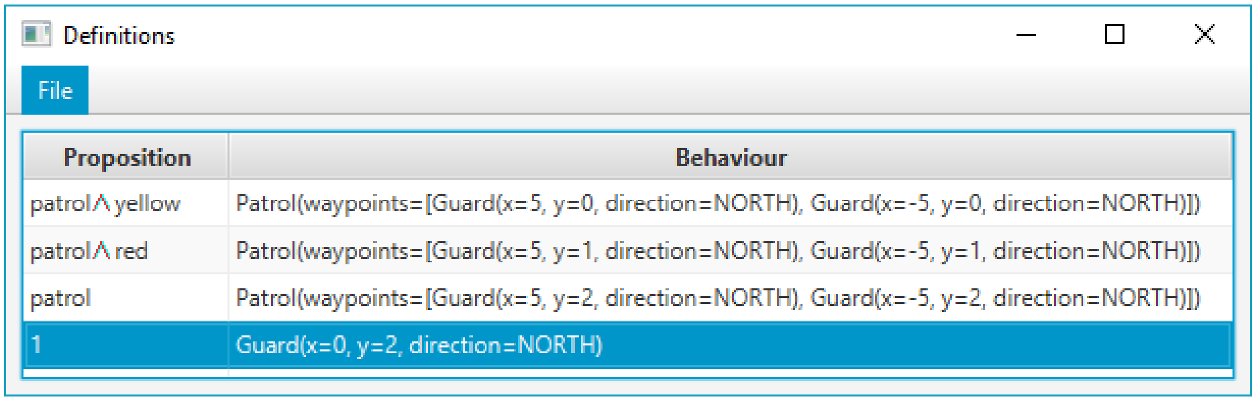
4.3.1 Agent Behavior Selection

When an agent's belief state changes, it looks through the Definitions (4.2) of the Definitions Table (6.2.2) to determine what behavior to perform. The Definition (4.2) the Agent (4.3) chooses to perform is the behavior of the first Definition (4.2) whose proposition is satisfied by the agent's belief state.



Online	Color	Current belief state
yes	🟨	patrol∧yellow
yes	🟥	patrol∧red
yes	🟩	patrol∧green



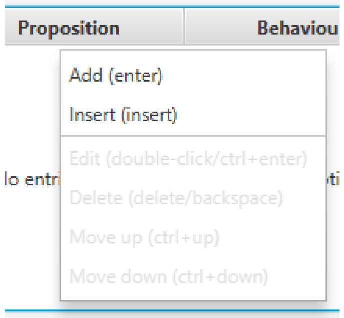


Proposition	Behaviour
patrol∧yellow	Patrol(waypoints=[Guard(x=5, y=0, direction=NORTH), Guard(x=-5, y=0, direction=NORTH)])
patrol∧red	Patrol(waypoints=[Guard(x=5, y=1, direction=NORTH), Guard(x=-5, y=1, direction=NORTH)])
patrol	Patrol(waypoints=[Guard(x=5, y=2, direction=NORTH), Guard(x=-5, y=2, direction=NORTH)])
1	Guard(x=0, y=2, direction=NORTH)

The screenshots above show three agents with different belief states in a single simulation performing different behaviors:

- The yellow Agent (4.3) is patrolling at `y=0` because its belief state satisfies the first Definition (4.2) in the Definitions Table (6.2.2) which maps `patrol ^ yellow` with patrolling along `y=0`.
- The red Agent (4.3) is patrolling at `y=1` because its belief state failed to satisfy the first Definition (4.2) in the Definitions Table (6.2.2), but it satisfies the second Definition (4.2) which maps `patrol ^ red` with patrolling along `y=1`.
- The green Agent (4.3) is patrolling at `y=2` because its belief state fails to satisfy the first two Definitions (4.2) in the Definitions Table (6.2.2); however, it satisfies the third Definition (4.2) which maps `patrol` to patrolling along `y=2`.

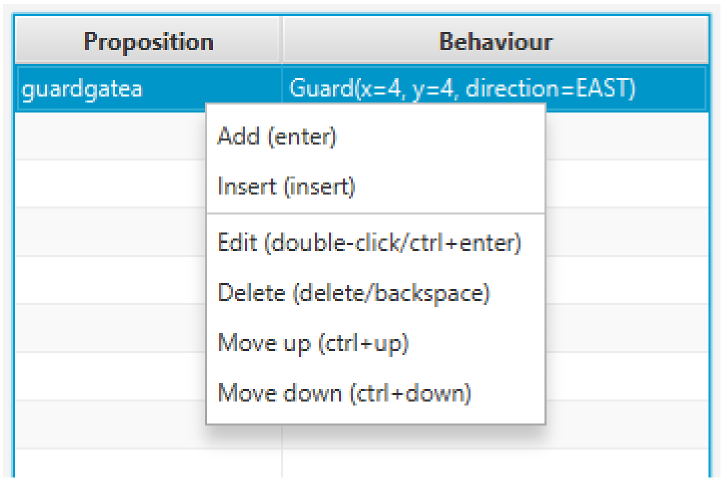
5 Editable Table View



The Editable Table View (5) is used in many places throughout the application, so it is important to understand it.

Each column of the table describes an attribute of some object and each row of the table describes a single instance of the object.

Right clicking anywhere on the table while no list items are selected will produce the context menu shown on the left.



Right clicking on the table while a list item selected will show context menu with item-specific options enabled as shown to the left.

Hotkeys for each context menu item are shown in parentheses to the right of each item.

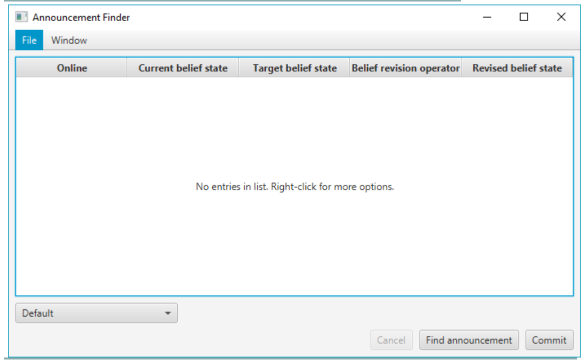
The items in the context menu each have different functions:

- *Add* – Opens an *input dialog* for you to enter details about the new row
  - If you fill out the dialog and press *OK*, a new row will be added to the table after the currently selected row.
  - If you press *Cancel*, the list shall remain unchanged.
- *Insert* – Like *Add* except this will create the new row before the selected row.
- *Edit* – Opens an *input dialog* for you to modify the row data. The input dialog is prefilled with the row's current data.
  - If you fill out the input dialog and press *OK*, your changes will be applied to the selected row.
  - If you press *Cancel*, the list will remain unchanged.
- *Delete* – removes the selected row from the table.
- *Move up* – moves the selected row above the row above it.
- *Move down* – moves the selected row below the row below it.

6 Windows

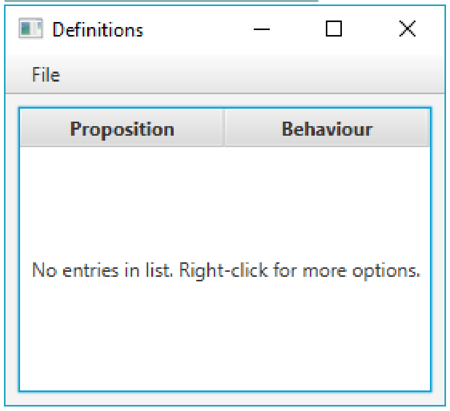
This section describes the all windows and dialog boxes that exist in the application. There are 4 main windows:

- Announcement Finder Window (6.1)



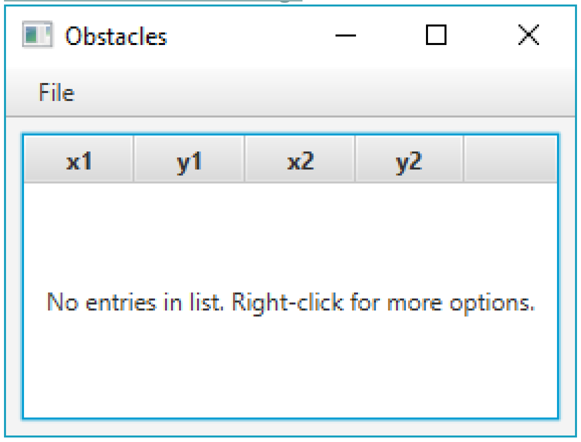
No entries in list. Right-click for more options.

- Definitions Window (6.2)



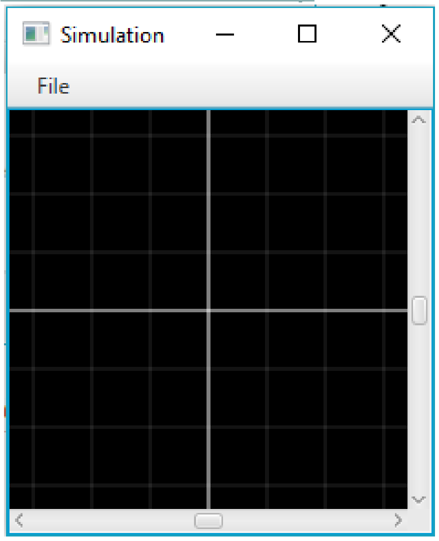
No entries in list. Right-click for more options.

- Obstacles Window (6.3)

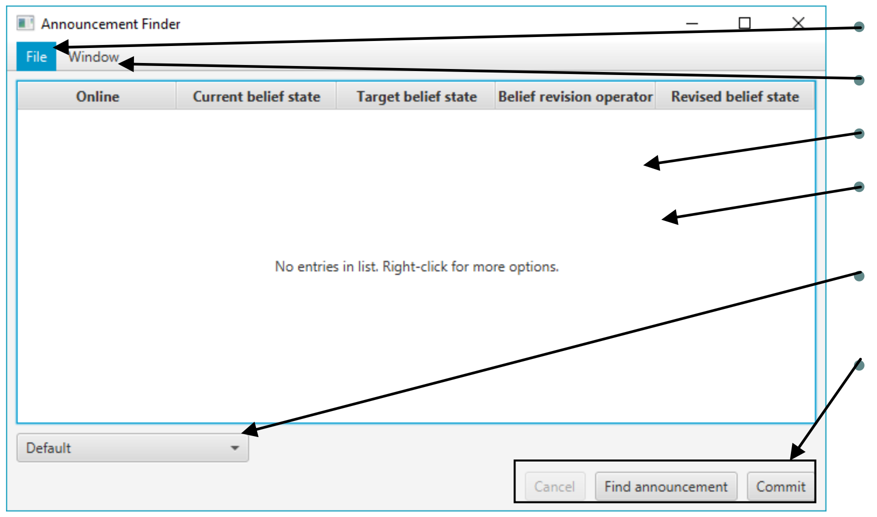


No entries in list. Right-click for more options.

- Simulation Window (6.4)



6.1 Announcement Finder Window



File Menu (6.1.1)

Window Menu (6.1.2)

Agents Table (6.1.3)

Agents Table Input Dialog (6.1.3.1)

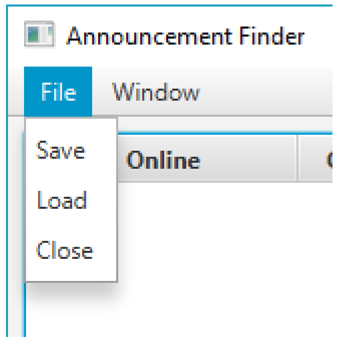
Display Mode Drop-Down Menu (6.1.4)

Cancel, Find Announcement and Commit Buttons (6.1.5)

The Announcement Finder Window (6.1) is shown above. It is the first window that appears when starting the application. The main activities for this window include:

- Viewing and modifying the list of agents
- Finding an announcement for the list of agents
- Opening and closing peripheral windows

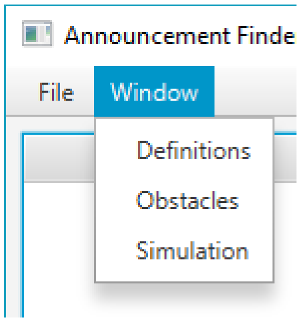
6.1.1 File Menu



The File Menu (6.1.1) lets you:

- *Save* the current list of Agents (4.3) to a text file.
- *Load* a list of Agents (4.3) from a save file into the program. This will overwrite all existing agents.
- *Close* the window. This will also close all peripheral windows and terminate the application.

6.1.2 Window Menu



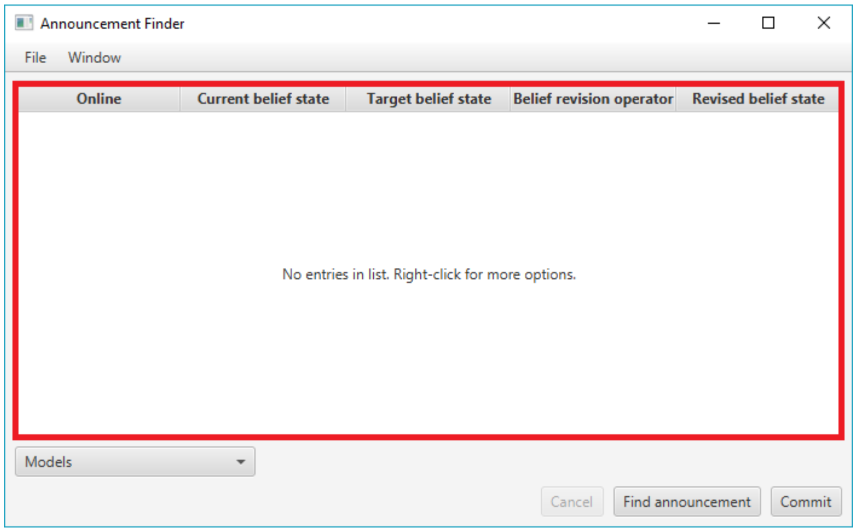
The Window Menu (6.1.2) can open and close peripheral windows:

- Definitions Window (6.2)
- Obstacles Window (6.3)
- Simulation Window (6.4)

When a peripheral window is open, a check mark will appear beside the corresponding menu item in the Window Menu (6.1.2).

Multiple peripheral windows may be open at once.

6.1.3 Agents Table

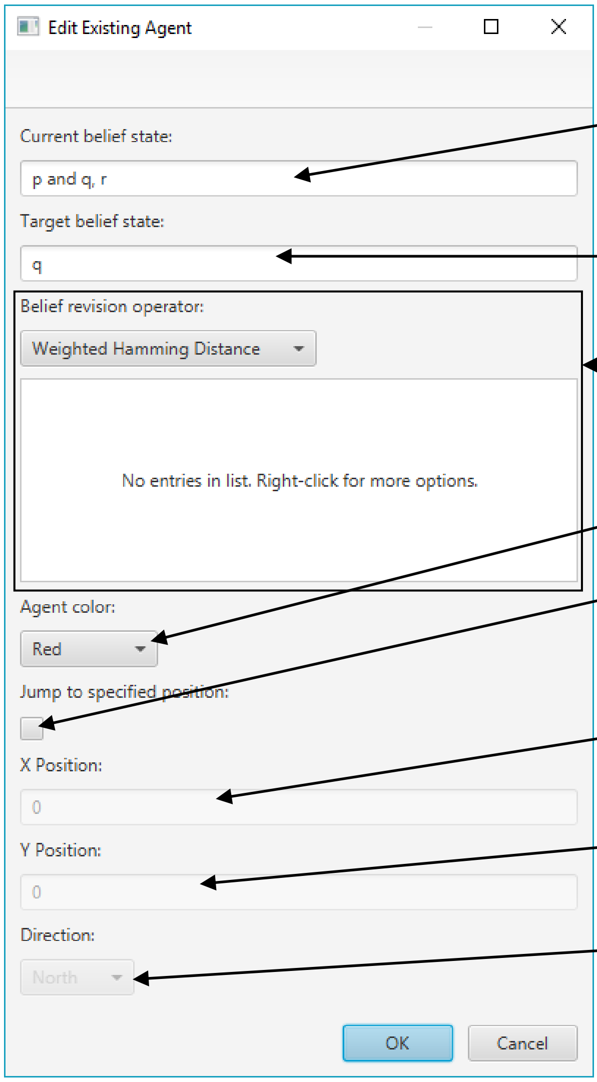


The Agents Table (6.1.3) is an Editable Table View (5) which allows you to view and modify the list of Agents (4.3). The input dialog used by this table is the Agents Table Input Dialog (6.1.3.1).

The columns in the agents table are:

- *Online* – whether or not the agent instance is connected to the application. Virtual agents are always connected, but connections could be lost with physical robot agents (e.g. bad Bluetooth connection).
- *Current belief state* – current belief state of the agent. This combined with the Definitions Table (6.2.2) determines what behavior the agent shall exhibit. See Agent Behavior Selection (4.3.1).
- *Target belief state* – the belief state you want the agent's belief state to satisfy after it is revised by the announcement.
- *Belief revision operator* – belief revision operator the agent uses when doing belief revision.
- *Revised belief state* – the belief state the agent would adopt if it were to revise by the announcement.

6.1.3.1 Agents Table Input Dialog



An edit agent input dialog is shown on the left. It has many interactive components:

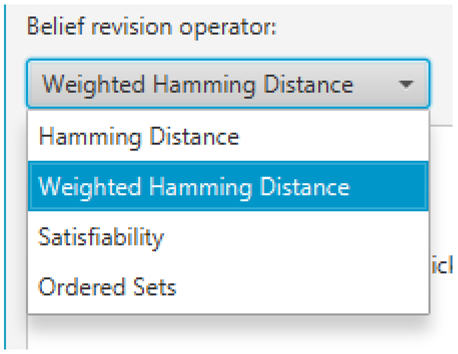
- *Current belief state* – define the agent's current belief state here by entering a comma separated list of propositional logic sentences.
- *Target belief state* – define the agent's target belief state by entering a single propositional logic sentence.
- *Belief revision operator* – choose the agent's belief revision operator. See Choosing the Agent's Belief Revision Operator (6.1.3.1.1).
- *Agent Color* – Choose what color the agent appears in on the simulator.
- *Jump to specified position* – check this box to have the agent jump to the x y position and face a direction specified in the fields below the checkbox.
- *X Position* – the x position the agent should jump to if *jump to specified position* is checked. Takes a signed integer as input.
- *Y Position* – the y position the agent should jump to if *jump to specified position* is checked. Takes a signed integer as input.
- *Direction* – the direction that the agent should face if *jump to specified position* is checked.

All propositional logic sentences must be entered using the Propositional Logic Sentence Input Syntax (7).

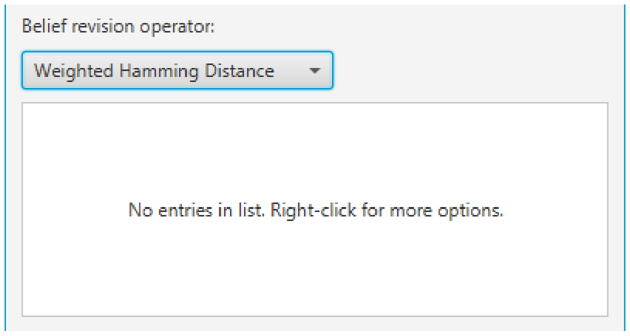
6.1.3.1.1 Choosing the Agent’s Belief Revision Operator

You can choose a belief revision operator for an Agent (4.3) when you are adding a new agent or editing an existing one through the Agents Table Input Dialog (6.1.3.1).

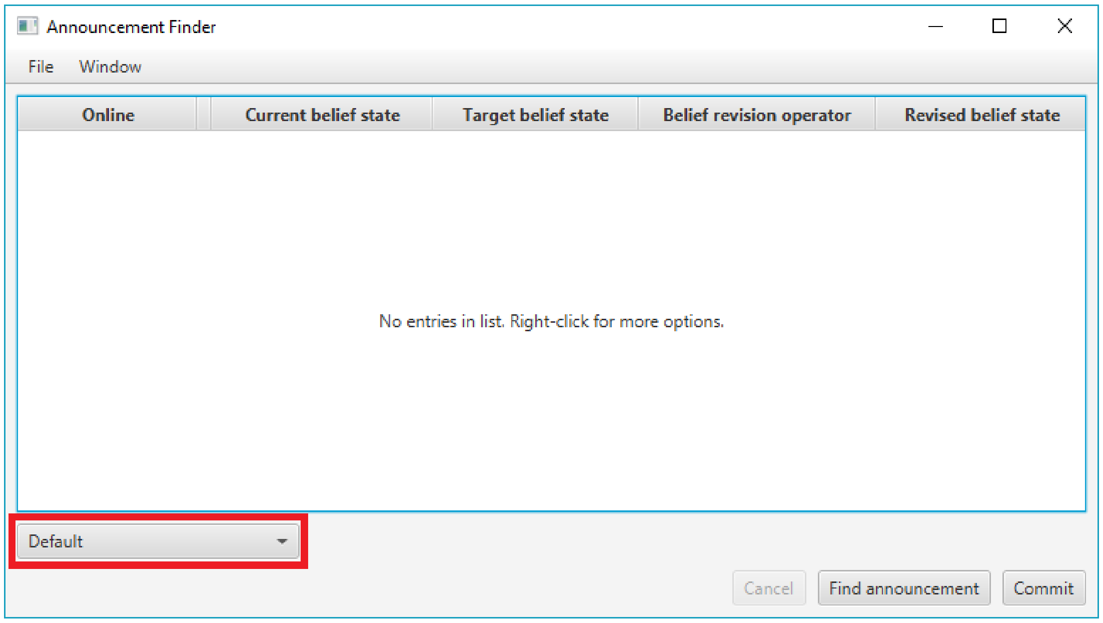
The interactive component used to specify the belief revision operator is a drop down menu as shown below. Using the drop down menu you can select which belief revision operator you would like to use for the Agent (4.3).


Belief revision operator:
Weighted Hamming Distance
Hamming Distance
Weighted Hamming Distance
Satisfiability
Ordered Sets
icl

Some belief revision operators require additional input to work properly so when they are selected, some more input controls will appear below the drop down menu as shown below. Follow the on-screen instructions to configure these belief revision operators.


Belief revision operator:
Weighted Hamming Distance
No entries in list. Right-click for more options.

6.1.4 Display Mode Drop-Down Menu



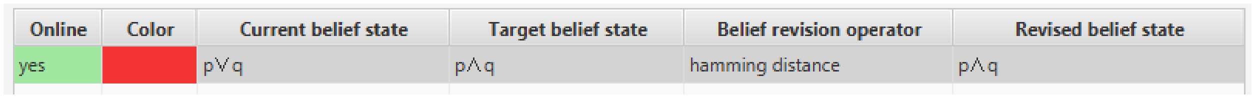
Online	Current belief state	Target belief state	Belief revision operator	Revised belief state

No entries in list. Right-click for more options.

The Display Mode Drop-Down Menu (6.1.4) is located near the bottom left of the Announcement Finder Window (6.1) as shown above.

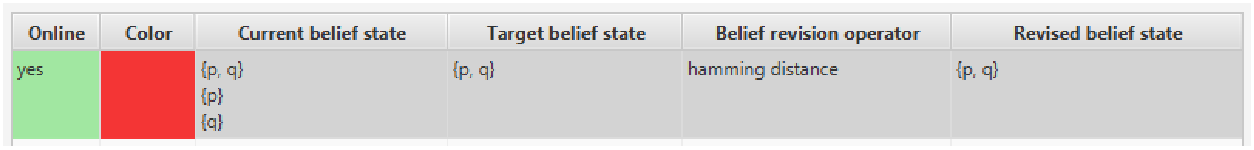
Changing the value in this control changes the way belief states for all agents are displayed:

- *Default* – belief states are displayed the way they were inputted into the application:



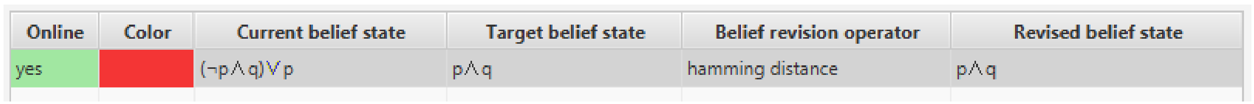
Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟥	p∨q	p∧q	hamming distance	p∧q

- *Models* – belief states are displayed as a set of states where each state in the set is displayed on its own line:



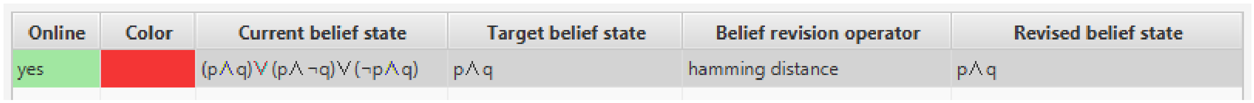
Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟥	{p, q} {p} {q}	{p, q}	hamming distance	{p, q}

- *Disjunctive Normal Form* – The application attempts to express each belief state as a disjunction of conjunctions in their most simple form:



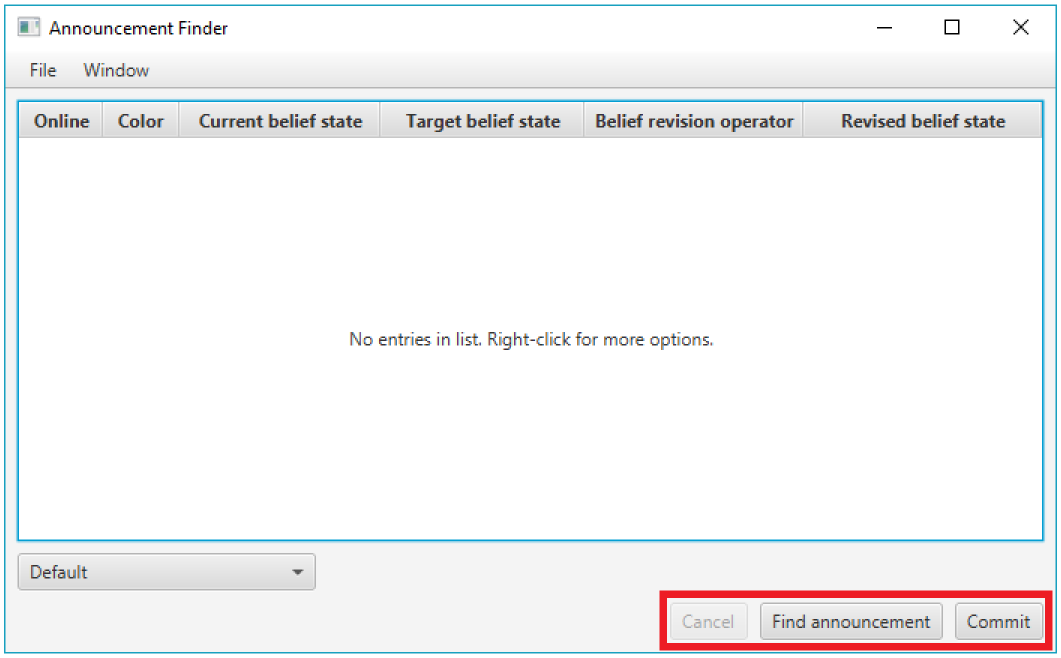
Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟥	(¬p∧q)∨p	p∧q	hamming distance	p∧q

- *Full Disjunctive Normal Form* – The application appends to express each belief state as a disjunction of conjunctions where each conjunction involves every known variable:



Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state
yes	🟥	(p∧q)∨(p∧¬q)∨(¬p∧q)	p∧q	hamming distance	p∧q

6.1.5 Cancel, Find Announcement and Commit Buttons

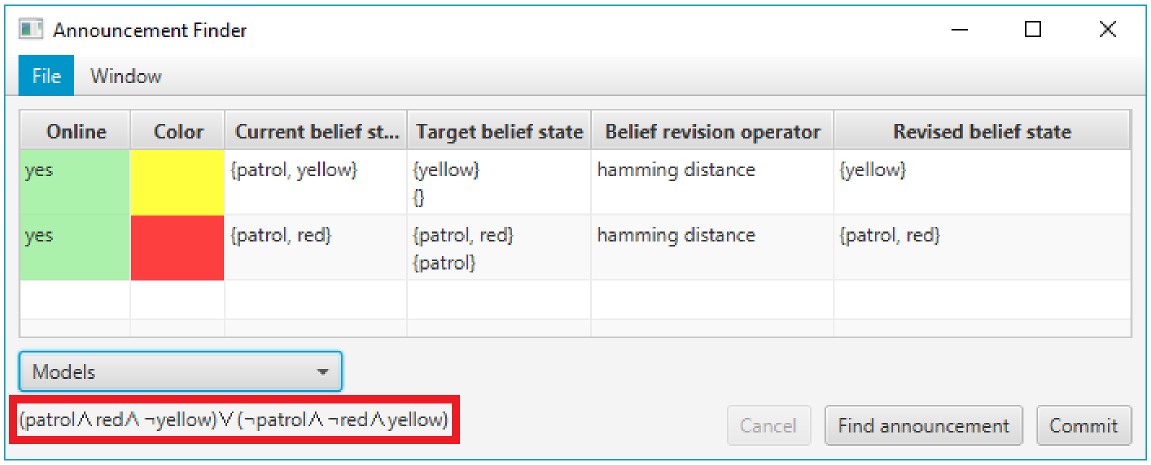


Online	Color	Current belief state	Target belief state	Belief revision operator	Revised belief state

No entries in list. Right-click for more options.

The *Cancel*, *Find announcement* and *Commit* buttons are located near the bottom right of the Announcement Finder Window (6.1):

- *Find Announcement* – begins computation of an announcement that when all agents revise their belief state by, their resulting belief state would satisfy their target belief state. Once an announcement has been computed, it will be displayed on the bottom left of the Announcement Finder Window (6.1), below the Display Mode Drop-Down Menu (6.1.4):



Online	Color	Current belief st...	Target belief state	Belief revision operator	Revised belief state

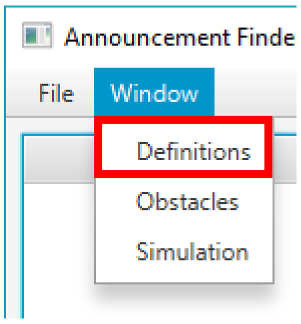
yes	🟨	{patrol, yellow}	{yellow}
			{}
				hamming distance	{yellow}

yes	🟥	{patrol, red}	{patrol, red} {patrol}	hamming distance	{patrol, red}

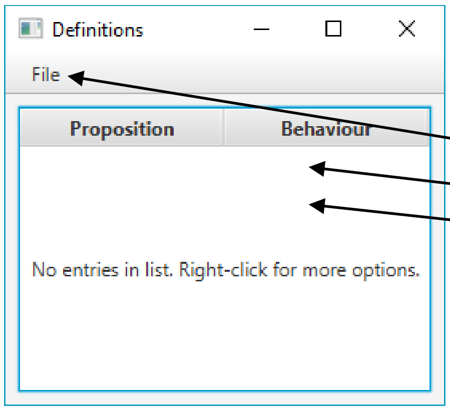
(patrol∧red∧¬yellow)∨(¬patrol∧¬red∧yellow)

- *Cancel* – stops the computing of an announcement. The button will be enabled while the application is computing an announcement. It is disabled otherwise.
- *Commit* – when an announcement is found, a preview of the revised belief states of each agent is displayed in the *Revised belief state* column. Pressing *Commit* will make the agents adopt the revised belief state as their current belief state.

6.2 Definitions Window



The Definitions Window (6.2) can be opened from the Announcement Finder Window (6.1) via Window > Definitions as shown on the left.

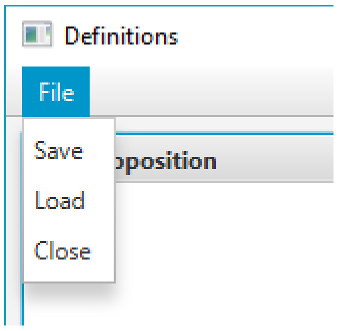


The Definitions Window (6.2) is shown on the left.

The Definitions Window (6.2) is used to define an ordered list of Definitions (4.2).

- File Menu (6.2.1)
- Definitions Table (6.2.2)
- Definitions Table Input Dialog (6.2.2.1)

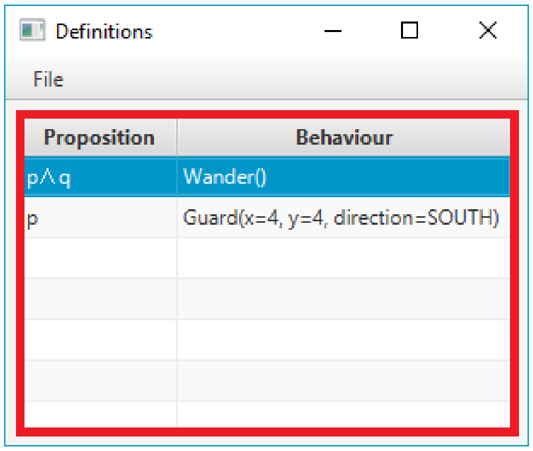
6.2.1 File Menu



The File Menu (6.2.1) lets you:

- *Save* the current list of Definitions (4.2) to a text file.
- *Load* a list of Definitions (4.2) from a save file into the program. This will overwrite all existing definitions.
- *Close* the window.

6.2.2 Definitions Table



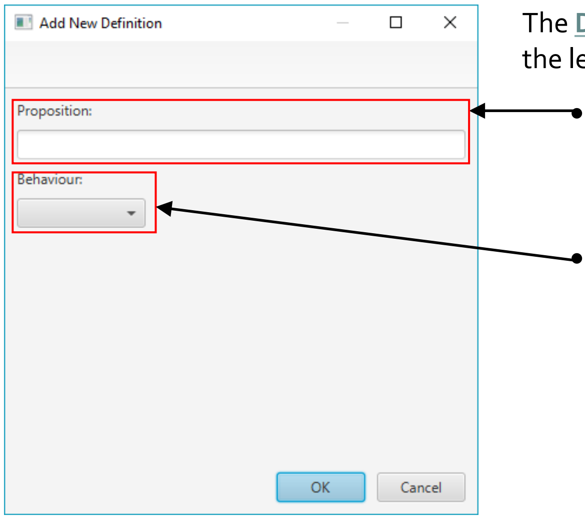
Proposition	Behaviour
p∧q	Wander()
p	Guard(x=4, y=4, direction=SOUTH)

The Definitions Table (6.2.2) is an Editable Table View (5) which allows you to view and modify the ordered list of Definition (4.2). The input dialog used by the Definitions Table (6.2.2) is the Definitions Table Input Dialog (6.2.2.1).

The columns of this table include:

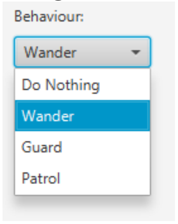
- *Proposition* – the propositional logic sentence that an agent's belief state must satisfy before performing the definition's associated *behavior*.
- *Behavior* – the agent behavior that an agent shall exhibit if its belief state satisfies the definition's *proposition*.

6.2.2.1 Definitions Table Input Dialog

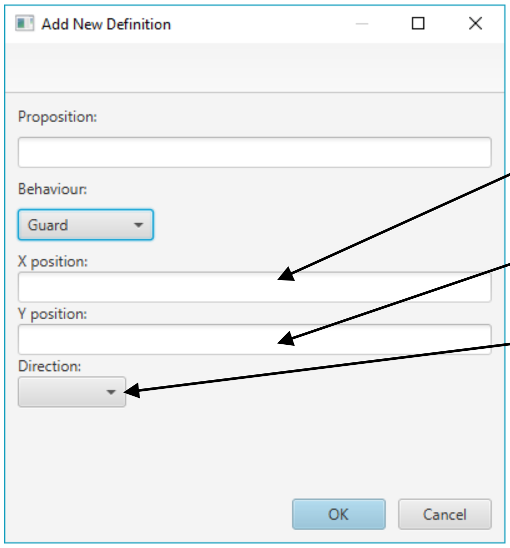


The Definitions Table Input Dialog (6.2.2.1) is shown on the left. It has a couple of interactive components:

- *Proposition* – enter the propositional sentence that an agent's belief state must satisfy before executing this behavior (see Propositional Logic Sentence Input Syntax (7)).
- *Behavior* – a drop-down menu where you can select various behaviors the agent should perform:

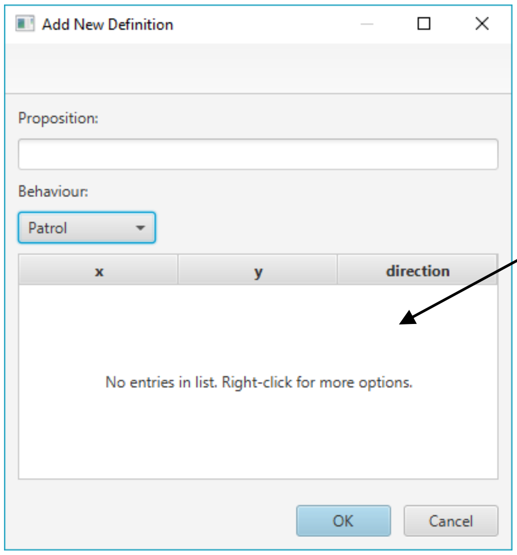


Some behavior options require additional inputs in order to work properly, so when they are selected, extra interactive components will appear below the *Behavior Drop-Down Menu*. Currently, only the *Guard* and *Patrol* behavior options require additional inputs.



When the *Guard* behavior is selected, a few extra controls appear below the *Behavior Drop-Down Menu*:

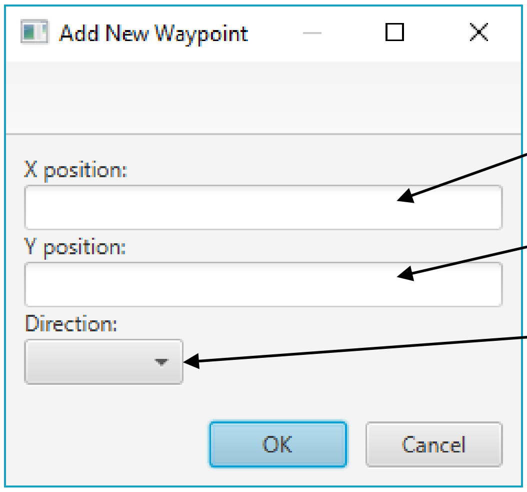
- *X position* – the x position that the agent should move to. This control takes a signed integer as input.
- *Y position* – the y position that the agent should move to. This control takes a signed integer as input.
- *Direction* – the direction that the agent should turn to face once it reaches the specified x y coordinates.



When the *Patrol* option is selected, an Editable Table View (5) appears below the *Behavior Drop-Down Menu*. The table view is used to edit a list of waypoints the agent should visit:

- *Waypoints table* – an Editable Table View (5) used to view and modify a list of waypoints. The input dialog used by this table view is the Waypoints Table Input Dialog (6.2.2.1.1).

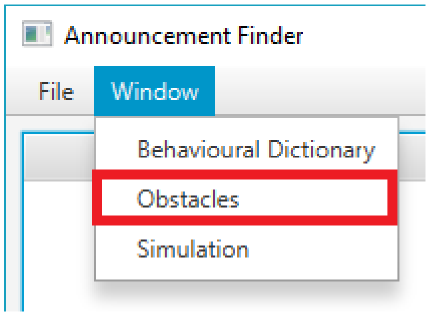
6.2.2.1.1 Waypoints Table Input Dialog



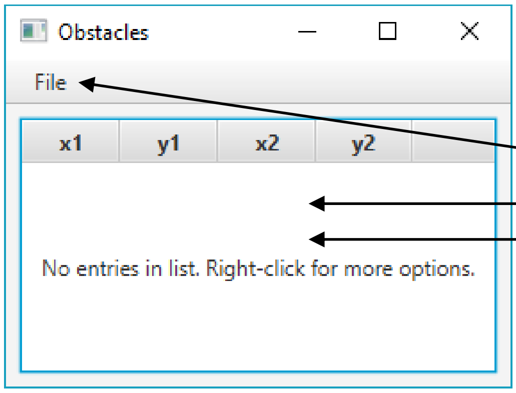
The Waypoints Table Input Dialog (6.2.2.1.1) has the following input controls:

- *X position* – the x position of the waypoint that the agent should visit. This control takes a signed integer as input.
- *Y position* – the y position of the waypoint that the agent should visit. This control takes a signed integer as input.
- *Direction* – the direction that the agent should turn to face once it reaches the waypoint's x y coordinates.

6.3 Obstacles Window



The Obstacles Window (6.3) can be opened from the Announcement Finder Window (6.1) via Window > Obstacles as shown on the left.

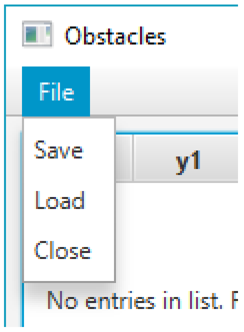


The Obstacles Window (6.3) is shown on the left.

The Obstacles Window (6.3) is used to view and modify the Obstacles (4.1) that exist in the simulation.

- File Menu (6.3.1)
- Obstacles Table (6.3.2)
- Obstacles Table Input Dialog (6.3.3)

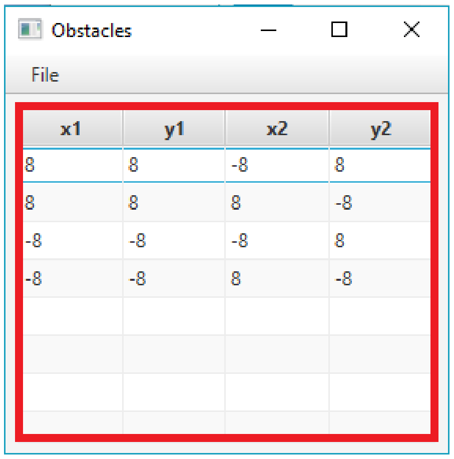
6.3.1 File Menu



The File Menu (6.3.1) lets you:

- *Save* the current list of Obstacles (4.1) to a text file.
- *Load* a list of Obstacles (4.1) from a save file into the program. This will overwrite all existing obstacles.
- *Close* the window.

6.3.2 Obstacles Table



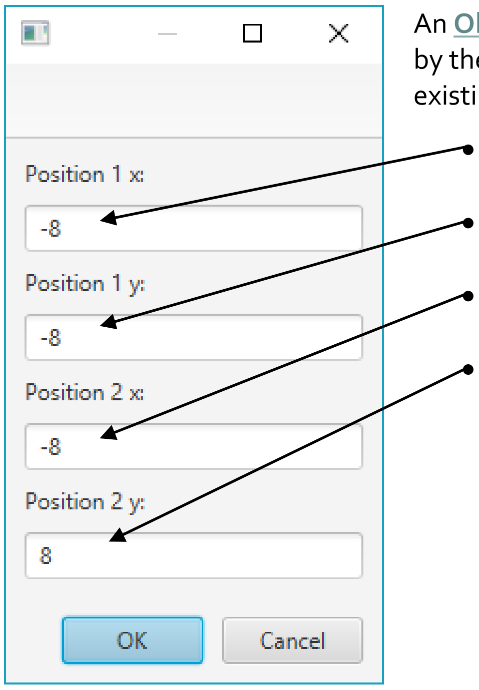
x1	y1	x2	y2
8	8	-8	8
8	8	8	-8
-8	-8	-8	8
-8	-8	8	-8

The Obstacles Table (6.3.2) is an Editable Table View (5) which allows you to view and modify the list of Obstacles (4.1). The input dialog used by the Obstacles Table (6.3.2) is the Obstacles Table Input Dialog (6.3.3).

The columns of this table include:

- *x1* – x position of first point in the Obstacle (4.1).
- *y1* – y position of first point in the Obstacle (4.1).
- *x2* – x position of second point in the Obstacle (4.1).
- *y2* – y position of second point in the Obstacle (4.1).

6.3.3 Obstacles Table Input Dialog



Position 1 x: -8

Position 1 y: -8

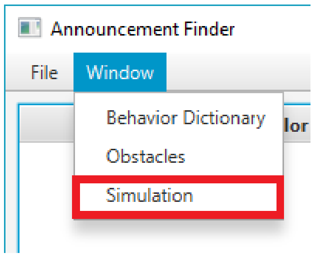
Position 2 x: -8

Position 2 y: 8

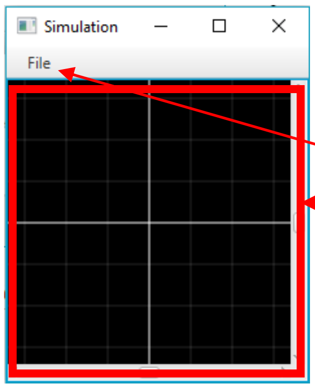
An Obstacle Table Input Dialog (3.3.3) is shown to the left. It is used by the Obstacles Table (6.3.2) when creating a new or editing an existing Obstacle (4.1) object. It has a few input fields:

- *Position 1 x* – Specifies the x position of the first point of the Obstacle (4.1). It takes a signed integer as input.
- *Position 1 y* – Specifies the y position of the first point of the Obstacle (4.1). It takes a signed integer as input.
- *Position 2 x* – Specifies the x position of the second point of the Obstacle (4.1). It takes a signed integer as input.
- *Position 2 y* – Specifies the y position of the second point of the Obstacle (4.1). It takes a signed integer as input.

6.4 Simulation Window



The Simulation Window (6.4) can be opened from the Announcement Finder Window (6.1) via Window > Simulation as shown on the left.

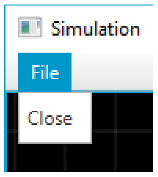


The Simulation Window (6.4) is shown to the left.

The Simulation Window (6.4) is used to observe the behavior of agents and how they react to changes in their belief states.

- File Menu (6.4.1)
- Simulation (6.4.2)

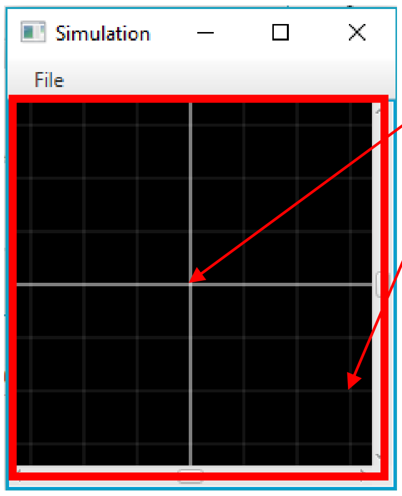
6.4.1 File Menu



The File Menu (6.4.1) lets you:

- *Close* the window.

6.4.2 Simulation



- The Simulation Window (6.4) is used to observe the behavior of agents and how they react to changes in their belief states.
- This is the origin; coordinate `x=0` and `y=0`.
- The length of every interval is exactly 1 unit.
- This is coordinate `x=3` and `y=2`.
- You can move the viewport by clicking and dragging in the simulation area.

7 Propositional Logic Sentence Input Syntax

There are two kinds of symbols in the Propositional Logic Sentence Input Syntax (7):

- Operands
- Operators

7.1 Operands

Name	Description	Input Symbol
Variable	An atomic proposition that may evaluate to true or false depending on the given state.	• Any combination of letters and numbers • Case insensitive • Must start with a letter • Examples: `p`, `patrol`, `abc123`, `guardGateA`, …
Tautology	An atomic variable that evaluates to true in every state.	• `1`
Contradiction	An atomic proposition that evaluates to false in every state.	• `0`

7.2 Operators

Name	Description	Input Symbol	Example
Implication (⟶)	`p→q` is true in all cases except for the case that `p` is true and `q` is false.	• `then`	• `p then q`
Equivalence (⟷)	`p⇔q` is true only when both `p` and `q` are true or both `p` and `q` are false.	• `iff`	• `p iff q`
Negation (¬)	`¬p` is true only when `p` is false.	• `-`	• `-q`
Conjunction (∧)	`p∧q` is true only when both `p` and `q` are true.	• `and`	• `p and q`
Disjunction (∨)	`p∨q` is true only when `p` is true or `q` is true or both `p` and `q` are true.	• `or`	• `p or q`
Exclusive Disjunction (⨁)	`p⨁q` is true only when either `p` or `q` is true, but not both.	• `xor`	• `p xor q`

7.3 Examples

- `variable`
- `-negated`
- `p iff -q xor r`
- `-abc123 or p90x`
- `-(p and q) then r`
- `guardGateA xor patrol`