|  |
| --- |
| Announcement Finder User Guide  Eric Tsang  August 30, 2016 |

# Table of Contents

[1 Table of Contents 2](#_Toc460558798)

[2 Procedures 3](#_Toc460558799)

[2.1 Obtain, Compile & Run the Project 3](#_Toc460558800)

[2.2 Example 1: Finding an Announcement 4](#_Toc460558801)

[2.3 Example 2: Changing Agent Behaviors with Announcements 5](#_Toc460558802)

[3 Demonstrations 8](#_Toc460558803)

[3.1 Hamming Distance 8](#_Toc460558804)

[3.2 Weighted Hamming Distance 9](#_Toc460558805)

[3.3 Patrol 10](#_Toc460558806)

[3.4 Protect the VIP 11](#_Toc460558807)

[4 Objects 12](#_Toc460558808)

[4.1 Obstacle 12](#_Toc460558809)

[4.2 Definition 12](#_Toc460558810)

[4.3 Agent 12](#_Toc460558811)

[4.3.1 Agent Behavior Selection 13](#_Toc460558812)

[5 Editable Table View 14](#_Toc460558813)

[6 Windows 15](#_Toc460558814)

[6.1 Announcement Finder Window 16](#_Toc460558815)

[6.1.1 File Menu 16](#_Toc460558816)

[6.1.2 Window Menu 16](#_Toc460558817)

[6.1.3 Agents Table 17](#_Toc460558818)

[6.1.4 Display Mode Drop-Down Menu 20](#_Toc460558819)

[6.1.5 Cancel, Find Announcement and Commit Buttons 21](#_Toc460558820)

[6.2 Definitions Window 22](#_Toc460558821)

[6.2.1 File Menu 22](#_Toc460558822)

[6.2.2 Definitions Table 23](#_Toc460558823)

[6.3 Obstacles Window 25](#_Toc460558824)

[6.3.1 File Menu 26](#_Toc460558825)

[6.3.2 Obstacles Table 26](#_Toc460558826)

[6.3.3 Obstacles Table Input Dialog 26](#_Toc460558827)

[6.4 Simulation Window 27](#_Toc460558828)

[6.4.1 File Menu 27](#_Toc460558829)

[6.4.2 Simulation 27](#_Toc460558830)

[7 Propositional Logic Sentence Input Syntax 28](#_Toc460558831)

[7.1 Operands 28](#_Toc460558832)

[7.2 Operators 28](#_Toc460558833)

[7.3 Examples 28](#_Toc460558834)

# Procedures

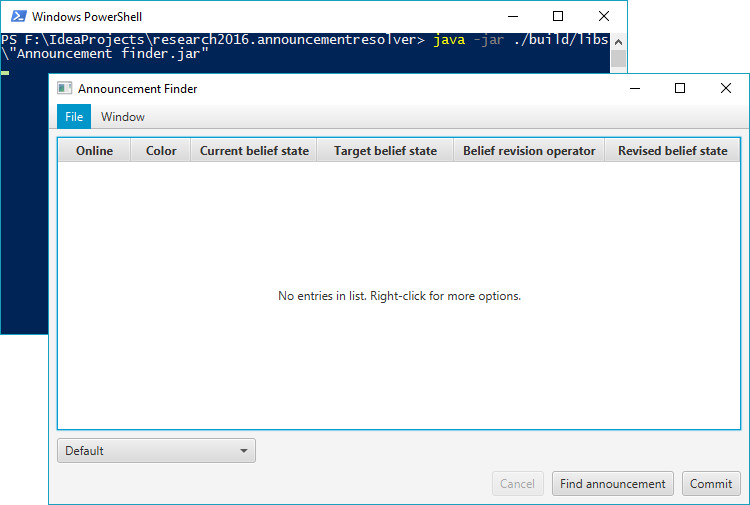
This section contains a few tutorials for obtaining, running and familiarizing yourself with the announcement finder program.

## Obtain, Compile & Run the Project

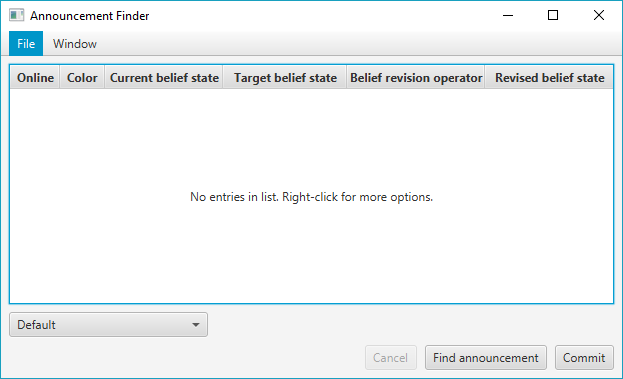
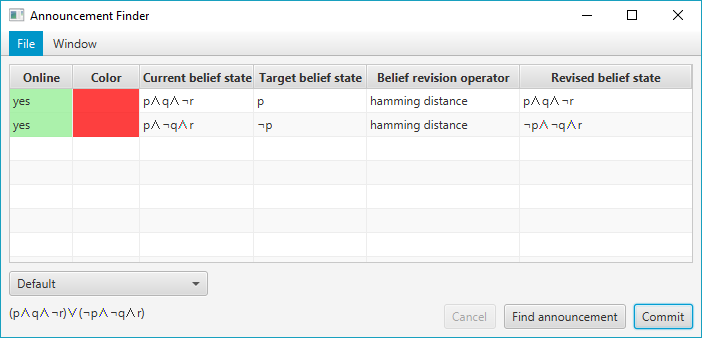
1. Download a .zip of the project from the link below and extract it:
   * <https://github.com/ericytsang/research2016.announcementresolver/archive/master.zip>
2. Open a terminal in the root directory of the project.
3. Make sure that both the java and javac commands are version 1.8.0\_91 or greater. 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:



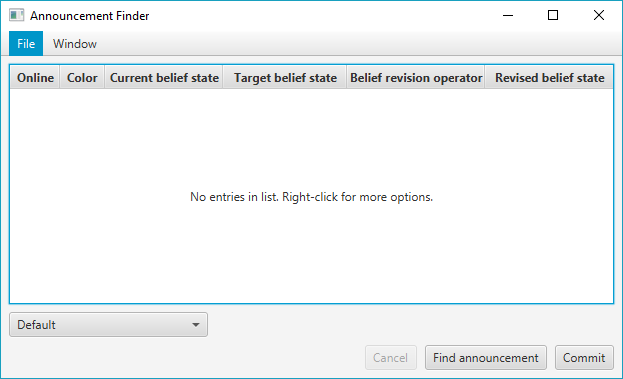
1. To execute the compiled JAR file, enter  
   java -jar ./build/libs/"Announcement finder.jar" into the terminal:

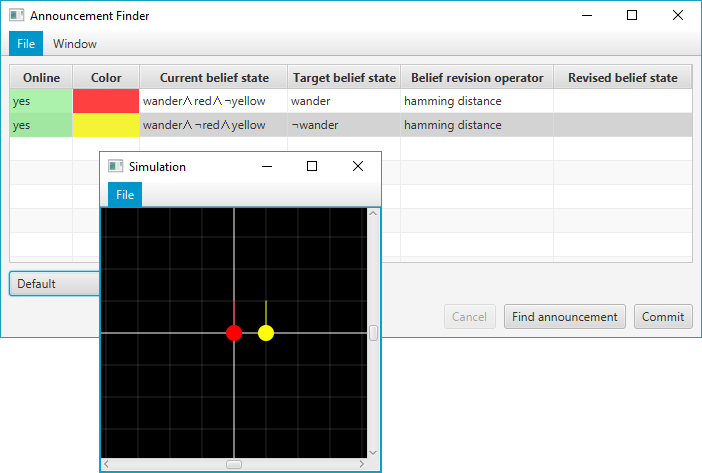


## Example 1: Finding an Announcement

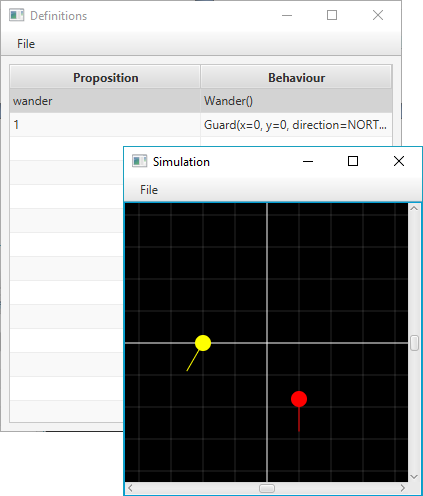
1. Execute the application. The () should appear:  
   
2. Right click the () then press Add. An () should appear:
3. In the ():
   1. Enter p and q and -r into the Current Belief State field.
   2. Enter p into the Target Belief State field.
   3. Leave everything else as their default values.
   4. Click OK.
4. Right click the () then press Add again. An () should appear:
5. In the ():
   1. Enter p and -q and r into the Current Belief State field.
   2. Enter -p into the Target Belief State field.
   3. Leave everything else as their default values.
   4. Click OK.
6. Click Find announcement in the ().  
   
   1. An announcement should appear in the bottom left corner of the ()
   2. 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.

## Example 2: Changing Agent Behaviors with Announcements

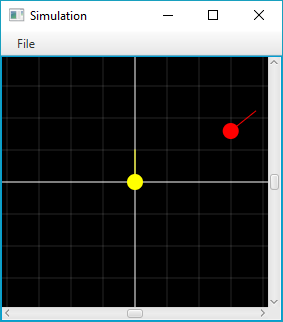
1. Execute the application. The () should appear:  
   
2. In the menu bar, click Window > Simulation to open the ().
3. Go back to the () and right click the () then press Add. An () should appear:
4. In the ():
   1. Enter wander and red and -yellow into the Current Belief State field.
   2. Enter wander into the Target Belief State field.
   3. Leave everything else as their default values.
   4. Click OK.
5. Right click the () then press Add again. An () should appear:
6. In the ():
   1. Enter wander and -red and yellow into the Current Belief State field.
   2. Enter -wander into the Target Belief State field.
   3. Set the Agent color to yellow.
   4. Set X Position to 1.
   5. Leave everything else as their default values.
   6. Click OK.
7. Notice that the () now displays 2 agents:



1. In the menu bar of the (), click Window > Definitions to open the ().
2. Right click the () then click Add. A () should appear.
3. In the ():
   1. Enter wander in the Proposition field.
   2. Set the Behavior drop-down menu to Wander.
   3. Click OK.
4. Right click the first item in the () again then click Add. A () should appear.
5. In the ():
   1. Enter 1 in the Proposition field.
   2. Set the Behavior drop-down menu to Guard.
   3. Set X position to 0.
   4. Set Y position to 0.
   5. Set Direction to NORTH.
   6. Click OK.
6. 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.



1. Click Find announcement in the ().
   1. An announcement should appear in the bottom left corner of the ()
   2. A preview of the revised belief states of each agent will appear in the Revised belief state column.
2. 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.



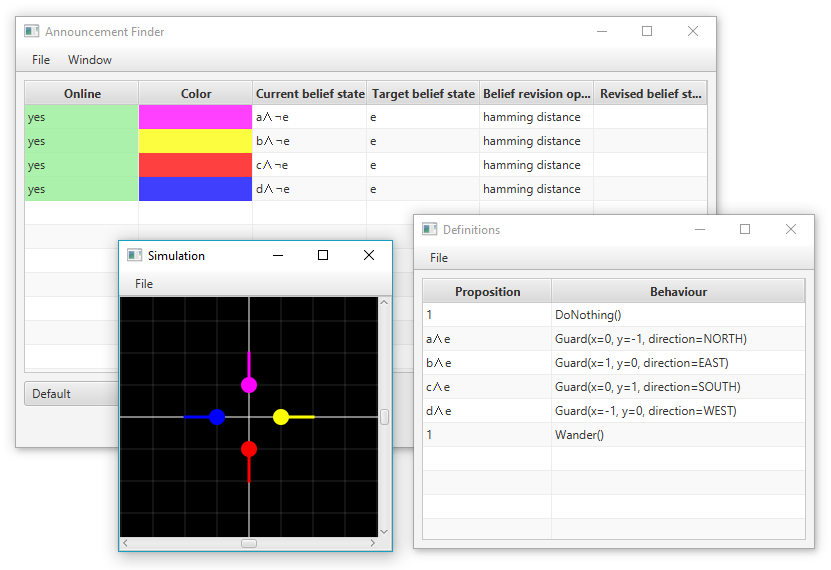
# Demonstrations

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

## 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 () set to Disjunctive Normal Form.

1. Open the () and () via the () in the () .
2. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/hammingDistance.definitions save file to load.
3. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/hammingDistance.agents save file to load.

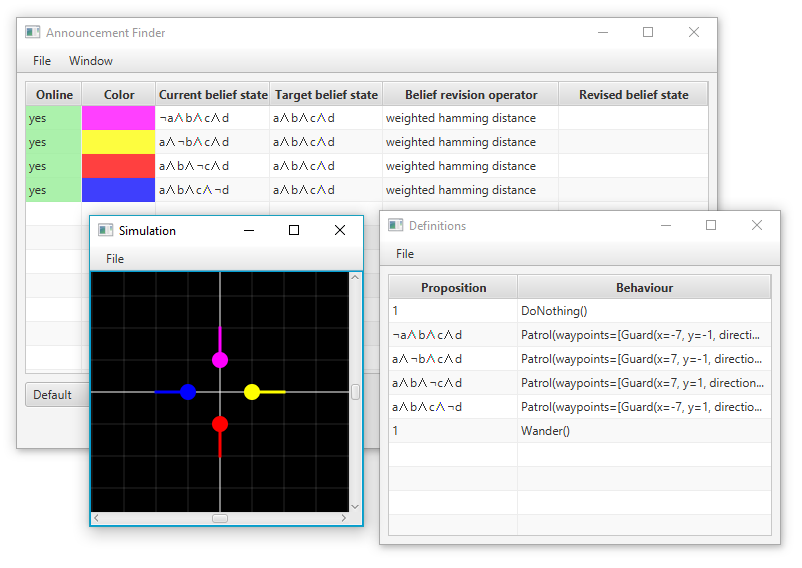


1. Select and delete the first item in the (). The agents in the simulator will begin wandering around.
2. In the (), click Find announcement then Commit. The agents will return to the center of the map.
3. You can modify the target belief states of every agent to –e, then repeat step 5 to have the agents begin wandering about again.

## Weighted Hamming Distance

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

1. Open the () and () via the () in the () .
2. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/weightedHammingDistance.definitions save file to load.
3. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/weightedHammingDistance.agents save file to load.

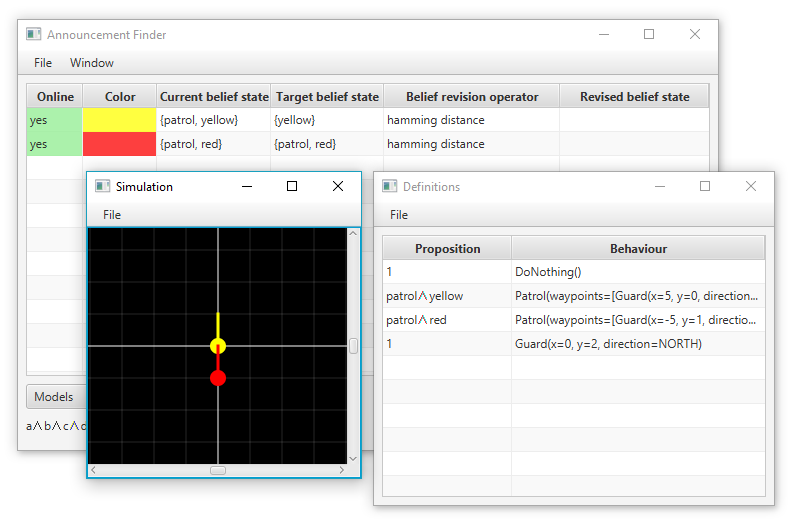


1. Select and delete the first item in the (). The agents in the simulator will begin patrolling.
2. In the (), click Find announcement then Commit. The agents start wandering.
3. At this point, you can replace the target belief states of the agents with their original belief states prior to the announcement (-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.

## 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 () set to Models.

1. Open the () and () via the () in the () .
2. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/patrol.definitions save file to load.
3. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/patrol.agents save file to load.

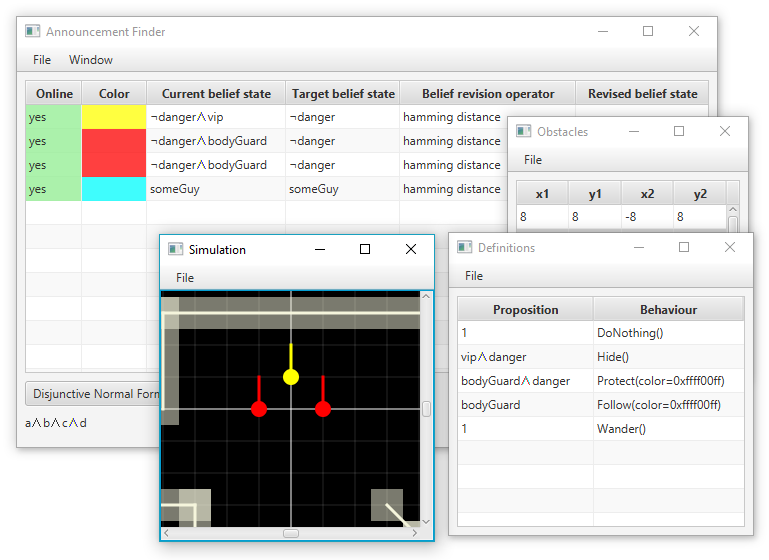


1. Select and delete the first item in the (). The agents in the simulator will begin patrolling.
2. In the (), click Find announcement then Commit. The yellow agent will stop patrolling and stand guard at [0,2].
3. You can change the target belief state of both agents and set both to patrol to true 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.

## 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 () set to Disjunctive Normal Form.

1. Open the (), () and () via the () in the () .
2. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/protectVip.definitions save file to load.
3. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/ protectVip.agents save file to load.
4. In the (), select File > Load. Then select the [rootProjectDirectory]/demo/ protectVip.obstacles save file to load.



1. Select and delete the first item in the (). The yellow and cyan agents will begin wandering the map. The red agents will begin following the yellow agent around.
2. In the (), click Find announcement then Commit. The yellow agent will flee to a corner, and the red agents will stick close by it.
3. 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.

# Objects

## Obstacle

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

## Definition

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

## Agent

|  |  |
| --- | --- |
|  | An (4.3) has a position, direction, color, belief state and belief revision operator. The image to the left shows the () with a red agent at position 3, 4 facing east.  The (4.3) will exhibit the behavior of the first (4.2) in the () that its belief state satisfies. See () for more details. |

### Agent Behavior Selection

When an agent’s belief state changes, it looks through the s () of the () to determine what behavior to perform. The () the (4.3) chooses to perform is the behavior of the first () whose proposition is satisfied by the agent’s belief state.

|  |  |
| --- | --- |
|  |  |
|  | |

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

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

# Editable Table View

|  |  |
| --- | --- |
|  | The () 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.

# Windows

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

|  |  |
| --- | --- |
| * () | * () |
| * () | * () |

## Announcement Finder Window

|  |  |
| --- | --- |
|  | * () * () * () * () * () * () |

The () 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

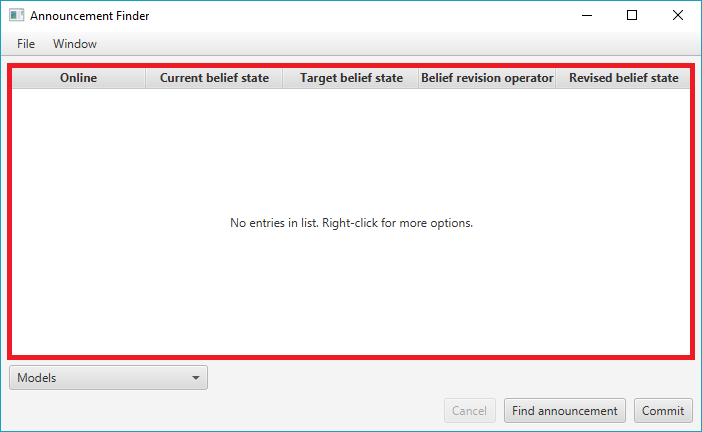
### File Menu

|  |  |
| --- | --- |
|  | The () lets you:   * Save the current list of s () to a text file. * Load a list of s () 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. |

### Window Menu

|  |  |
| --- | --- |
|  | The () can open and close peripheral windows:   * () * () * ()   When a peripheral window is open, a check mark will appear beside the corresponding menu item in the ().  Multiple peripheral windows may be open at once. |

### Agents Table



The () is an () which allows you to view and modify the list of s (4.3). The input dialog used by this table is the ().

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 () determines what behavior the agent shall exhibit. See ().
* 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.

#### Agents Table Input Dialog

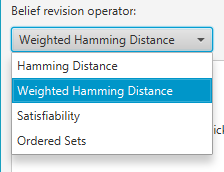
|  |  |
| --- | --- |
|  | An edit agent input dialog is shown on the left. It has many interactive components:   * Current belief state – define the agent’s current belief state here by entering a comma separated list of propositional logic sentences. * Target belief state – define the agent’s target belief state by entering a single propositional logic sentence. * Belief revision operator – choose the agent’s belief revision operator. See (). * 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 positionis 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 positionis checked*.* |

All propositional logic sentences must be entered using the ().

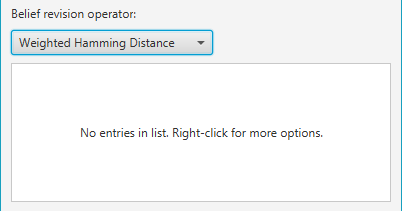
##### Choosing the Agent’s Belief Revision Operator

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

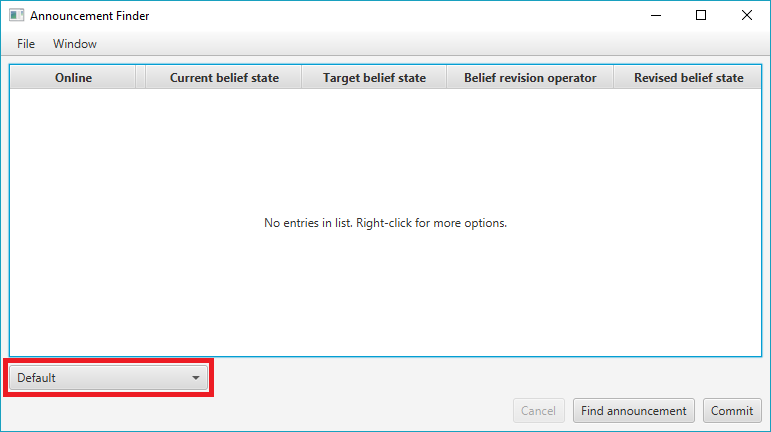
The interactive component used to specify the belief revision operator is a drop down menu as shown below. Using the drop down menu you can select which belief revision operator you would like to use for the (4.3).



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.

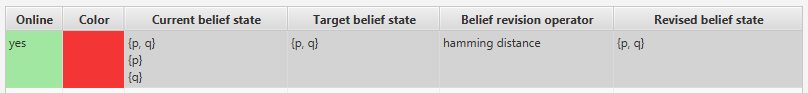


### Display Mode Drop-Down Menu

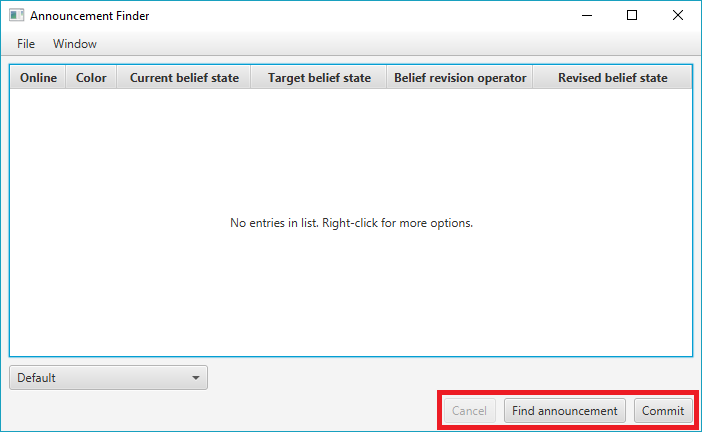


The () is located near the bottom left of the () as shown above.

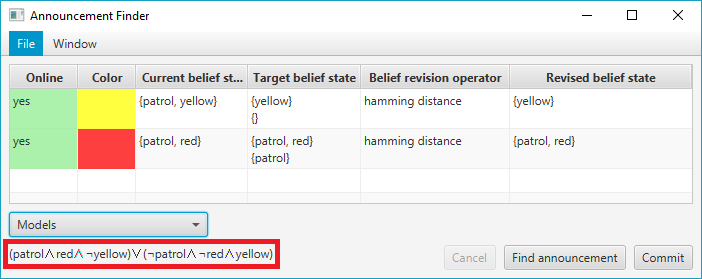
Changing the value in this control changes the way belief states for all agents are displayed:

* Default – belief states are displayed the way they were inputted into the application:  
  
* Models – belief states are displayed as a set of states where each state in the set is displayed on its own line:  
  
* Disjunctive Normal Form – The application attempts to express each belief state as a disjunction of conjunctions in their most simple form:  
  
* Full Disjunctive Normal Form – The application appends to express each belief state as a disjunction of conjunctions where each conjunction involves every known variable:  
  

### Cancel, Find Announcement and Commit Buttons



The Cancel, Find announcement and Commitbuttons are located near the bottom right of the ():

* 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 (), below the ():  
  
* 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.

## Definitions Window

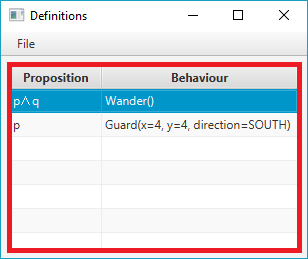
|  |  |
| --- | --- |
|  | The () can be opened from the () via Window > Definitions as shown on the left. |

|  |  |
| --- | --- |
|  | The () is shown on the left.  The () is used to define an ordered list of s ().   * () * () * () |

### File Menu

|  |  |
| --- | --- |
|  | The () lets you:   * Save the current list of s () to a text file. * Load a list of s () from a save file into the program. This will overwrite all existing definitions. * Close the window. |

### Definitions Table



The () is an () which allows you to view and modify the ordered list of (). The input dialog used by the () is the ().

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.

#### Definitions Table Input Dialog

|  |  |
| --- | --- |
|  | The () 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 ()). * 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 () 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 () used to view and modify a list of waypoints. The input dialog used by this table view is the (). |

##### Waypoints Table Input Dialog

|  |  |
| --- | --- |
|  | The () 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. |

## Obstacles Window

|  |  |
| --- | --- |
|  | The () can be opened from the () via Window > Obstacles as shown on the left. |

|  |  |
| --- | --- |
|  | The () is shown on the left.  The () is used to view and modify the s () that exist in the simulation.   * () * () * () |

### File Menu

|  |  |
| --- | --- |
|  | The () lets you:   * Save the current list of s () to a text file. * Load a list of s () from a save file into the program. This will overwrite all existing obstacles. * Close the window. |

### Obstacles Table

|  |  |
| --- | --- |
|  | The () is an () which allows you to view and modify the list of s (). The input dialog used by the () is the ().  The columns of this table include:   * x1 – x position of first point in the (). * y1 – y position of first point in the (). * x2 – x position of second point in the (). * y2 – y position of second point in the (). |

### Obstacles Table Input Dialog

|  |  |
| --- | --- |
|  | An Obstacle Table Input Dialog (3.3.3) is shown to the left. It is used by the () when creating a new or editing an existing () object. It has a few input fields:   * Position 1 x – Specifies the x position of the first point of the (). It takes a signed integer as input. * Position 1 y – Specifies the y position of the first point of the (). It takes a signed integer as input. * Position 2 x – Specifies the x position of the second point of the (). It takes a signed integer as input. * Position 2 y – Specifies the y position of the second point of the (). It takes a signed integer as input. |

## Simulation Window

|  |  |
| --- | --- |
|  | The () can be opened from the () via Window > Simulation as shown on the left. |

|  |  |
| --- | --- |
|  | The () is shown to the left.  The () is used to observe the behavior of agents and how they react to changes in their belief states.   * () * () |

### File Menu

|  |  |
| --- | --- |
|  | The () lets you:   * Close the window. |

### Simulation

|  |  |
| --- | --- |
|  | * The () 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. |

# Propositional Logic Sentence Input Syntax

There are two kinds of symbols in the ():

* Operands
* Operators

## 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 |

## 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 |

## Examples

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