BELIEFSET

RobotInfo: has the current robot data. This beliefset does not have key fields (only value fields) so it has only one entry. It contains the following fields:

* p, shorthand for position;
* ms, shorthand for movementState;
* lus, shorthand for loadUnloadState;
* lb, shorthand for loadedBox.

The position is a number from 0 to 15.

The movementState is the movement that the robot is doing. It can assume one of the following values:

* 0 = stop: the robot is not moving, its position doesn’t change
* 1 = running forward: the robot’s position is increasing
* 2 = running backward: the robot’s position is decreasing
* 3 = stopping: the robot is stopping

The loadUnloadState indicates if the robot has a box on board. It can be:

* 0 = unloaded: the robot has no box
* 1 = loaded: the robot has a box
* 2 = loading left: the robot is loading a box on his left
* 3 = loading right: the robot is loading a box on his right
* 4 = unloading left: the robot is unloading on his left
* 5 = unloading right: the robot is unloading on his right

The loadedBox reports the name of the box loaded on the robot. If there is no box, its value is null.

In the named data section of the JDE, robotInfo is a beliefset of type RobotInfo. robotInfo is used as private data by all RobotController agents and has to be used by all plans that need to read it.

RobotPerception: contains perception data of the robots being perceived. It contains position, name and movement. The key field is the position, the others are value fields.

position indicates the position of another robot perceived by the robot containing a RobotPerception. It can assume the following values:

* 0 = ahead: the other robot’s position is greater than mine
* 1 = behind: the other robot’s position is smaller than mine
* 2 = right: the other robot is not on my rail but it’s at my right
* 3 = left: the other robot is not on my rail but it’s at my left
* 4 = ahead right: the other robot isn’t on my rail but it’s ahead and at my right
* 5 = ahead left: the other robot isn’t on my rail but it’s ahead and at my left
* 6 = behind right: the other robot isn’t on my rail but it’s behind and at my right
* 7 = behind left: the other robot isn’t on my rail but it’s behind and at my right
* 8 = unknown: I don’t know where the other robot is

The robot can only know the position of another robot when it’s in the 8 areas around him.

name reports the name of the robot being perceived.

movement indicates the movement of the other robot compared to the perceiving robot. It doesn’t’ consider the relative velocity, only its direction. It can be:

* 0 = getting closer: the other robot is moving towards the perceiver (no matter if the perceiver is moving or stopped)
* 1 = going away: the other robot is moving in the opposite direction compared to me (no matter if the perceiver is moving or stopped)
* 2 = steady: the other robot’s position does not change
* 3 = unknown: I don’t know what the other robot is doing

In the named data section of the JDE, otherRobots is a beliefset of type RobotPerception. otherRobots is used as private data by all RobotController agents and has to be used by all plans that need to read it.

BoxesPerception: contains data on the boxes. It has position (key field) and name (value field).

position it can be 2 (right) or 3 (left, as in RobotPerception.position) because the boxes are detected on left or on right.

name is the name of the box.

In the named data section of the JDE, boxes is a beliefset of type BoxesPerception. boxes is used as private data by all RobotController agents and has to be used by all plans that need to read it.

VIEW

PerceptionInfo: connects the RobotPerception and the BoxesPerception.

It contains a complex query that returns the box name, the robot name and the robot movement given a position.

In the named data section of the JDE, perception is a view of type PerceptionInfo. perception is used as private data by all RobotController agents and has to be used by all plans that need to read it.

AGENT TYPES

RobotController: it’s the agent type that controls the robot.

It contains an updateTime method that updates the information about the robot.

RobotCoordinator: it’s the agent type that controls and coordinates all the robots in the environment. It sends a message to every single robot and when they have received it, they start their plan.

PLAN TYPES

CoordinationPlan: used by the RobotCoordinator. It sends the event to each robot (IRobotAgents). IRobotAgent is an interface that provides methods to control the agent.

After that, every robot starts their events.

OTHERS FILES

DataLog: contains log method that permits to simplify the print in java.

RobotData: contains all the robot information.

The most important thing in RobotData is the update() method. It checks all the fields and if there is a change, it notifies all the observers.

Do not use this actuators to see their value but the fields in the beliefset!