# Multi-Agent System in a disaster scenario (AgentSpeak)

Vinícius Lafourcade

# Scenario

- Disaster scenario at PUCRS;
- In this scenario, some victims have been rescued and are receiving medical treatment in some buildings, but these buildings don’t have enough supplies (medicines, food, water, blankets and curatives). Also, some victims must be taken to the hospital;
- The collect agents are responsible for collecting and delivering these supplies to these buildings to help the medical rehabilitation of the victims;
- The ambulance agents are responsible to taking the victims to the hospital;

# Scenario

- There are 5 types of supplies
  - Medicines
  - Food
  - Water
  - Blankets
  - Curatives
- The volume of the supplies were not implemented, but when a new supply is requested, the rescue point inform the necessary quantity;

# Scenario

- Was developed AgentSpeak plans to cover 2 classes of agents
  - Building agents
    - Collect Point
    - Delivery Point
    - Hospital
  - Transportation agents
    - Delivery agent
    - Ambulance
- Each delivery agent has a pre-defined load capacity to transport supplies;

# AgentSpeak

- AgentSpeak is an agent-oriented programming language;
- It is based on logic programming and BDI (belief-desire-intention) architecture;
- AgentSpeak was created by Anand Rao in 1996 *
- Some conventions are important to define
  - Variables are represented with the first character uppercase;
  - Properties are represented with the first character lowercase;
  - All beliefs, desires and intentions are executed in a context, then we can define that some intention is valid only in a determined context, e.g., in a context where the door is opened does not make sense that the agent ask to open the again. This context can be defined in AgentSpeak as follows:
    - +!door(Status) : door(CurrentStatus) & Status \== CurrentStatus
      - This intention will be valid only when the current status of the door is different than the status passed by parameter;

* Anand S. Rao, 1996. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. Proceedings of Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-96).

# AgentSpeak - Belief

- The belief is something that the agent assumes as true, e.g., the agent has a belief that the door is open, so if another agent ask this agent about the door status, the agent will reply that the door is open. To represent it based in AgentSpeak, we can use the belief +door(open);
- AgentSpeak allows 4 operations in the agent belief base: addition (“+”), deletion (“-”), consult (“?”) and update (“-+”)
- Examples
    - Addition a new belief: +position(10, 11) - Will add a new position belief with the coordinates (10,11);
    - Deletion the belief: -position(10, 11) - Will remove the position belief with the coordinates (10,11);
    - Consult a belief: ?position(X, Y) - Will ask the position belief and return the values to the variables X and Y;
    - Update a belief: -+position(20,25) - Will update the position belief with the coordinates (20,25)

# AgentSpeak - Desire

- As the name suggests, the desire is something that the agent desires to achieve (goal), e.g., the agent knows that the door is open, but it desires the door to be closed, to achieve this goal in the AgentSpeak we can use a desire !door(close).
- The desires are represented by a “!” symbol;

# AgentSpeak - Intention

- The intention is the way that the agent can achieve its goals, e.g., every time the agent want to change the door state it demonstrate an intention to change that status, to represent this intention in AgentSpeak we can use +!door(Status), where the Status is a variable and can be close or open;

# AgentSpeak - Initial Setup

- To run the simulation on Jason or Pyson the building agents must receive a “pose(X,Y)” belief and the delivery agents must receive a “loadCapacity(Capacity)” belief;
- These configuration must be completed based on the AgentSpeak interpreter (Jason or Pyson);

# AgentSpeak - Building agent workflow

- The building agent start printing its initial position and initial status;
- After that, the building agent sends its position to the transportation agents;

# AgentSpeak - Rescue point agent workflow

- With this first steps completed, the rescue point agent request a new supply. To choose which item should be requested, an aleatory number from 0 to 1 is generated, as below:
  - 0 to 0.17: Medicines;
  - 0.17 to 0.34: Food;
  - 0.34 to 0.51: Water;
  - 0.51 to 0.68: Blankets;
  - 0.68 to 0.85: Curatives;
  - 0.85 to 1: Victim transportation;
- When a supply is chosen, a new aleatory number from 0 to 1 is generated and multiplied to 100, this value will be represent the quantity of supplies that the rescue point needs;
- When a supply is delivered, a new supply is requested;

# AgentSpeak - Delivery agent workflow

- The delivery agent starts the simulation printing its starting position and its starting status;
- After that the delivery agent will waits for a new supply request from the rescue points;
- When a new supply request is available, the agent validates if its load capacity is higher than the supply quantity needed;
    - If this validation fails, the agent discards the request;
    - If this validation pass, the agent notifies the other agents that it will attend that request;
- To define which agent will attend a request, the name the agents is used
    - If the agent name is higher than the other agent, it will discards the request, e.g., if the agent1 is competing with the agent2 for the request, the agent1 will wins because its name is lower than the name of the other agent;

# AgentSpeak - Delivery agent workflow

- When a request is confirmed, the agent changes its status to “GOING_TO_COLLECT” and call a custom action called go_to(X, Y) (that must be available on Jason and Pyson) where the X and Y will be the coordinates of the collect point. This custom action must notify the ROS to move the robot;
- Every time that ROS has a new position of the robot, Jason and Pyson must add a new belief called execute(X, Y) to the agent belief base, then the agent will process its plans and will continues its logic;
- When the agent arrives to the collect point, the agent changes its status to “GOING_TO_DELIVERY_POINT” and call the go_to(X,Y) custom action to the ROS move the robot to the delivery point;
- When the agent arrives to the delivery point, the supply is delivered and the workflow is restarted;

# AgentSpeak - Ambulance agent workflow

- The ambulance agent has a similar workflow than the delivery agent workflow, the difference is while the delivery agent goes to the collect point to get the supply and delivers to the rescue point, the ambulance start going to the rescue point to get the victim and after transports to the hospital;

# Limitations

- We found some limitations during our project;
  - We found a limitation on Pyson that we cannot update a belief using "-+" operator. For this reason we created simple plans only to remove the current belief and add a new one with an updated value.

```
+!updateStatus(Status) : true
    <-
    ?currentStatus(CurrentStatus);
    -currentStatus(CurrentStatus);
    +currentStatus(Status).

+!updateCurrentRequest(Request) : true
    <-
    ?currentRequest(CurrentRequest);
    -currentRequest(CurrentRequest);
    +currentRequest(Request).
```

  - The conditional on the AgentSpeak-py never satisfied our plans while in Pyson we could achieve the conditions properly.

# Limitations

- Unfortunately we could not implement the battery level logic, but we think this logic does not impacted the final result, once we were focused in provide plans to execute in both platforms (Jason and Pyson);

# DEMO

# Links

- GitHub
  - https://github.com/disaster-robotics-proalertas/pucrs_campus_gazebo
- Documentation
  - http://pucrs-campus-on-gazebo.readthedocs.io/en/latest/