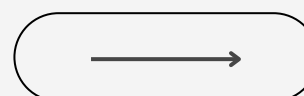


Jason Multi-Agent System for City Safety: Agents Cooperating to Arrest Criminals

Multi-Agent Systems Course Final Project
Creative Project Classified as Hard

Giulia Benvenuto - 4678610



INTRODUCTION

Project Overview

The project utilizes Jason, an interpreter for AgentSpeak, to develop a Multi-Agent System that simulates a city environment.

In this environment, four types of agents are implemented, each exhibiting a different behavior. Police, civilian, and clue agents cooperate to detect and arrest criminal agents to ensure safety in the city.

PROBLEM STATEMENT

City Safety Problem

Safety is a key issue in city areas, affecting the happiness of the residents. Thus, it's reasonable to think that civilians living in the city will collaborate with police to enhance their personal safety and contribute to the overall security of the environment.

Why a multi-agent system is suited for this problem

MAS aligns well with the City Safety Problem due to its ability to manage complex interactions among multiple agents who cooperatively work toward the common goal of ensuring city safety by arresting criminals.

DEVELOPMENT FRAMEWORK

Jason

Jason is an interpreter for an extended version of AgentSpeak. It is crucial in the project for defining and managing the behavior of the agents (police, civilian, clue, and criminals) making them able to dynamically interact basing on their beliefs, desires and intentions.

Java

Enhances Jason capabilities with the possibility of integrate and extend functionalities of the MAS. Allows for the dynamic addition of percepts to agents as they navigate the city environment, and supports the development of core system components (CityEnvironment, CityModel and CityView java classes), of internal actions and visual representation of the environment.

ENVIRONMENT (1)

Simulated City Environment

The City Environment in the MAS serves as the dynamic stage where agents perform their roles and interact. It is implemented through a structured architecture that involves the use of three Java classes that extend Jason classes:

- CityEnvironment.java → extends Jason.environment.Environment
- CityModel.java → extends GridWorldModel
- CityView.java → extends GridWorldView

ENVIRONMENT (2)

CityEnvironment.java

Initialize the city simulation environment where agents interact and navigate. Handles the movement of police agents icon step after step.

CityModel.java

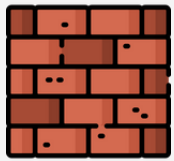
Responsible for creating and managing the state of the 40x40 city grid, it handles both dynamic and static entities within the environment.

CityView.java

Responsible for the visual representation of the city environment, including drawing icons of obstacles and agents. Also, handles updates in response to changes within the model.

ENVIRONMENT (3)

City Environment Result



Wall Obstacle



House Obstacle



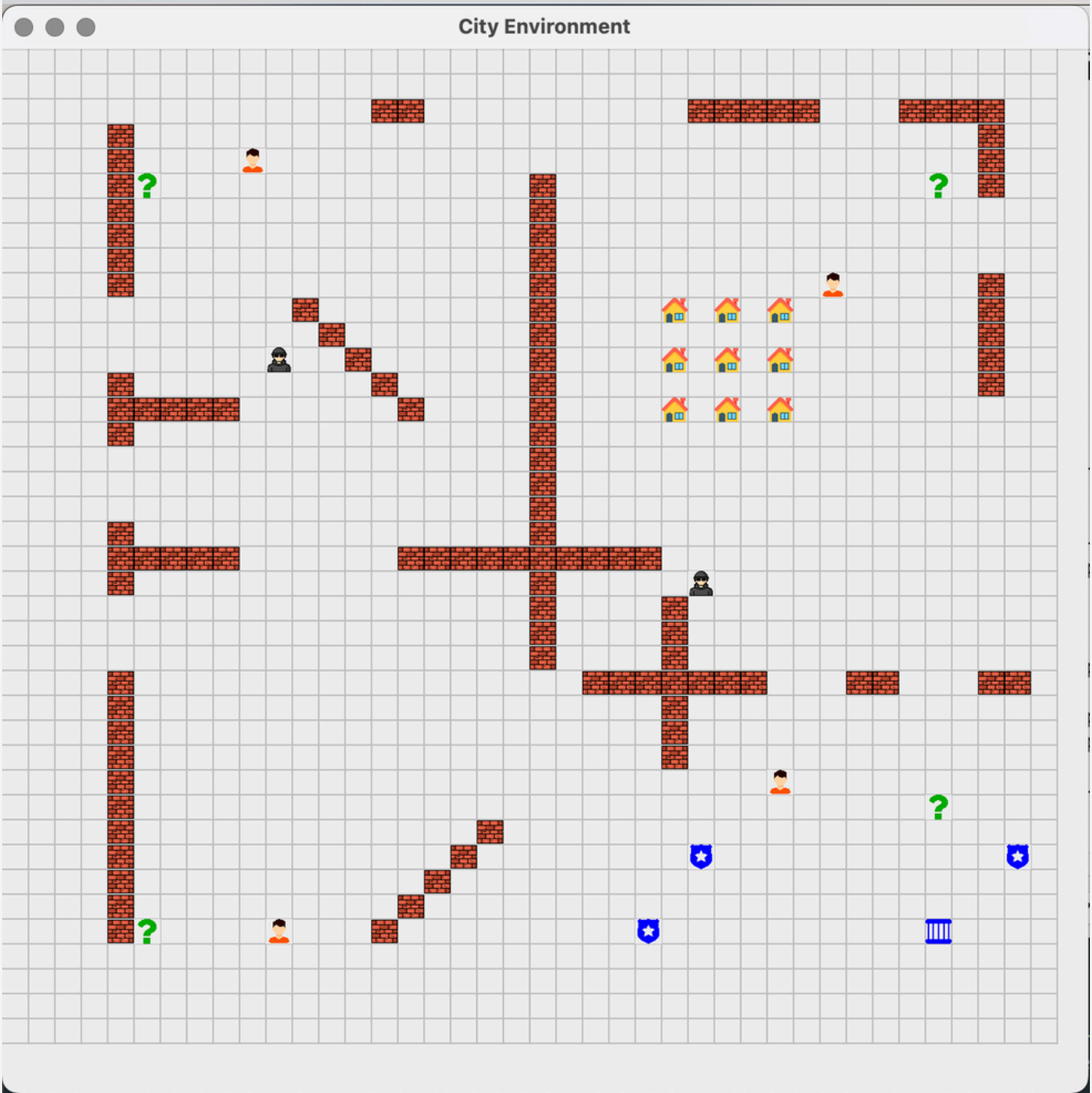
Jail



One criminal in jail



Two criminals in jail



Police Agent



Police Agent Escorting



Civilian Agent



Clue Agent



Criminal Agent

AGENTS (1)

Behavior of Police Agents

- Three police agents.
- Complex and dynamic position.
- Navigate the city following the shortest path computed with the A^* algorithm.
- At each step, look around the 8 neighboring cells.
- If they find a civilian agent take the clue position (X, Y) from it and next move there.
- If they find a clue agent collect the X or Y of a criminal.
- If both X and Y of a criminal has been found, they move there.
- If they find a criminal, arrests him, escort him to the jail and broadcast other agents that a criminal has been arrested.



Police Agent Icon



Police Agent Icon while Escorting

AGENTS (2)

Behavior of Civilian Agents

- Four civilian agents.
- Static position.
- If they are found by a police agent, communicate to him the position (X, Y) of their closest clue.



- If they receive a broadcast message from a police agent about the arrest of a criminal they respond with a message thanking him.

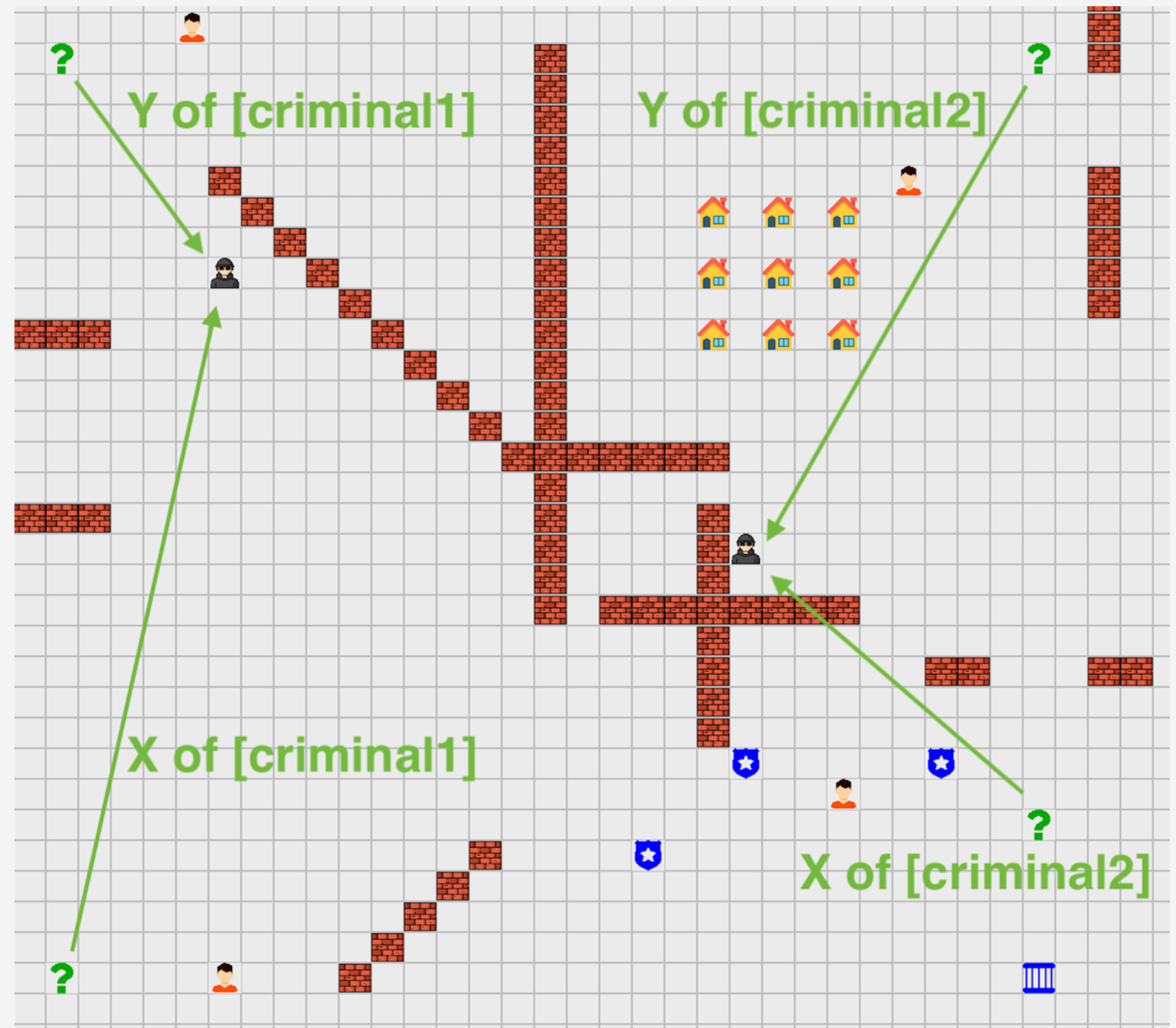


Civilian Agent Icon

AGENTS (3)

Behavior of Clue Agents

- Four clue agents.
- Static position.
- If they are found by a police agent, communicate to him the known X or Y coordinate and the ID of the criminal.



Clue Agent Icon

AGENTS (4)

Behavior of Criminal Agents

- Two criminal agents.
- Static position.
- They hide in the city and upon being discovered, they receive a message from the police agent declaring that they are under arrest.
- When arrested by a police agent, they are removed from the grid as they are escorted to jail.



Criminal Agent Icon

INTERNAL ACTIONS (1)

What are Internal Actions?

Internal Actions are used in the project as a building block for agents to perform operations that go beyond basic logical reasoning.

To develop an Internal Action, one must extend a java class with the "DefaultInternalAction" class provided by Jason.

This approach allows the implementation of specific agent behaviors.

INTERNAL ACTIONS

1) **FindPath.java**

Handles the logic of finding a path between two locations (x, y) in the city grid. Used by police agents in the “explore” goal.

2) **Escorting.java**

Handles the escorting logic for the police agents. It activates the escorting state to true when a police agent arrest a criminal, updating to the police agent's icon.

3) **EnterJail.java**

Manages a police agent's entry into jail with a criminal, including removing the agent's icon from the grid to indicate they are inside.

4) **Arrested.java**

Handles the logic to arrest a criminal which involves removing the criminal's icon from the grid when a police agent finds and arrest him.

UTILITY AND HELPER CLASSES

1) **AgentIdMapper.java**

Maps agent IDs from Java to Jason since they have a different numbering systems: Java IDs range from 0 to 12, while Jason IDs span from 0 to one less than the total agent instances.

2) **AgentPercept.java**

Adds and updates the agents' beliefs, so it acts as the intermediary layer between the agent representation in Java and Jason.

3) **LookAround.java**

Used by police agents at each step to detect the presence of other agents, obstacles or jail in the eight neighboring cells around their current position.

Thank You