Traffic-simulation-MAS

Intelligent systems engineering

2024-2025

Project goal: introduction

This project concerns the creation of a multi-agent system that manages the coordination and modeling of autonomous vehicles.

Its distinctive feature is the approach to communication and coordination between agents, which is entirely based on the concept of cognitive stigmergy. Coordination between agents does not take place through direct communication, but through the use of a shared space, in which each agent leaves its own cognitive feedback.

In cognitive stigmergy, therefore, communication does not take place through simple indirect feedback, as in the case of ant swarms, but rather through the sharing of structured information that also includes the agent's intentionality

Project goal: features

- creation of an environment in which agents can co-exist and roads are added. Each road has two lanes with a predetermined direction to emulate real behavior;

- creation of reactive agents capable of communicating through the environment, with a stigmergic and cognitive approach; in particular, an agent's intentionality allows it to intelligently calculate its next step;

- integration of a system of timed traffic lights (cartago artifact), each with its own internal timer, which facilitates coordination;

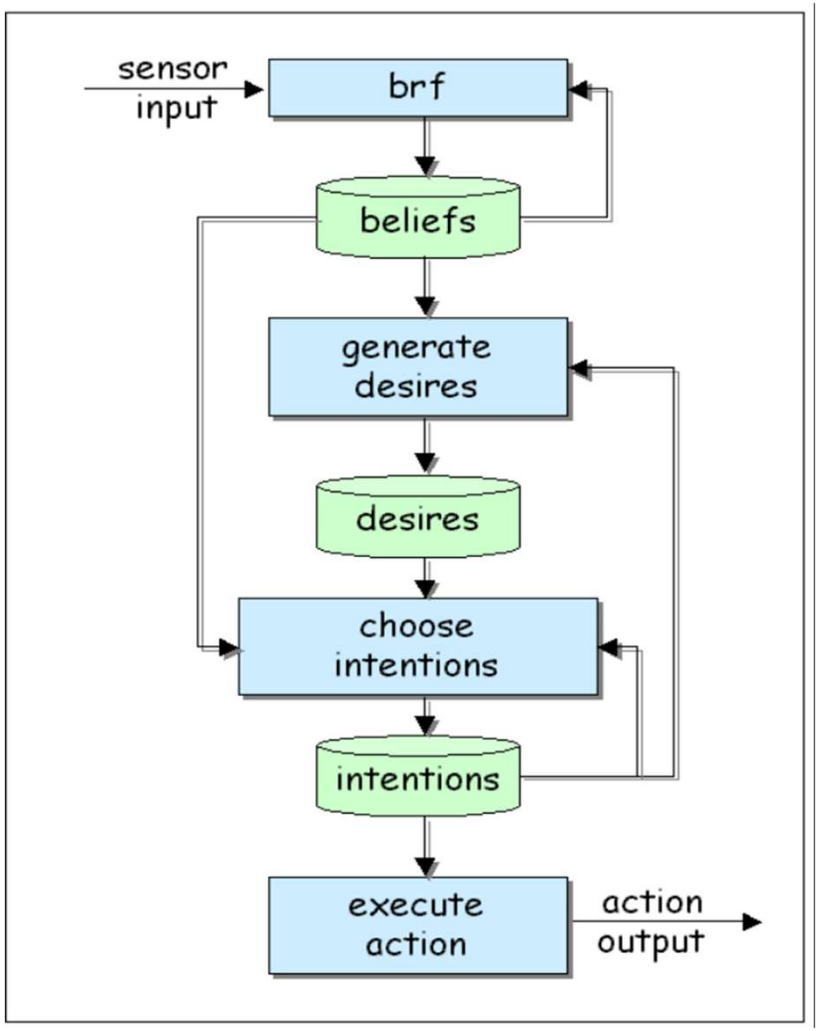
- possibility of adding intersections to the map, which are by definition the meeting point of four different roads, and through intentionality, the mechanism governing right of way is defined;

- possibility of adding simple turns to the map;

- creation of a rather complex mechanism to randomly generate all MAS components. The generator opens up the possibility of creating potentially infinite scenarios, which can then be exploited, through metrics (currently not present), for performance calculation.

Context: BDI architecture

In this project, agents are built with a BDI architecture, since Jason, the main technology is built with it. The BDI model is very efficient for balancing a deliberative vs reactive approach. Its structure can be described by the schema displayed on the right.



Context: self-organizing systems & cognitive stigmergy

- This project aims to build a self-organized system, which is a process in which a pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interaction among the system's components are executed using only local information, without reference to the global pattern. Self organization is really common in biology.
- Nowadays, stigmergy refers to a set of coordination mechanism mediated by the environment, in which agents release feedbacks indirectly, in the environment, such that they can be used by other agents in order to choose the better path or action (e.g in ant colonies chemical substances, namely pheromones, act as markers for specific activities).
- An interesting extension of the stigmergy concept is cognitive stigmergy that was introduced in 2007 by prof. Ricci and prof. Omicini. Agents can perceive, share and use artifacts to support individual goals without central control. The designed system (mine), leverages the presented principle, agents coordinate indirectly via shared environmental artifacts.

Technologies

- Jacamo;
  - As stated and explained in the JaCaMo documentation, the platform covers three different main dimensions, such as:
    - agents → autonomous entities (jason);
    - environment → shared space in which agents interact;
    - organization → rules, roles and structures that facilitate coordination.
  - The architecture is structured in three different levels, jason, cartago and moise that can also be used separately.
- Jason
  - Jason is an interpreter for an extended version of AgentSpeak. It implements the operational semantics of that language, and provides a platform for the development of multi-agent systems, with many user-customisable features. One of the best known approaches to the development of cognitive agents is the BDI (Beliefs-Desires-Intentions) architecture.
- CartAgO;
  - CArtAgO is based on the Agents & Artifacts (A&A) meta-model for modelling and designing multi-agent systems. A&A introduces high-level metaphors taken from human cooperative working environments: agents as computational entities performing some kind of task/goal-oriented activity (in analogy with human workers), and artifacts as resources and tools dynamically constructed;
- Java.

Design: Architecture

The architecture has been divided into two macro components, agent layer and environment layer.

Agent layer

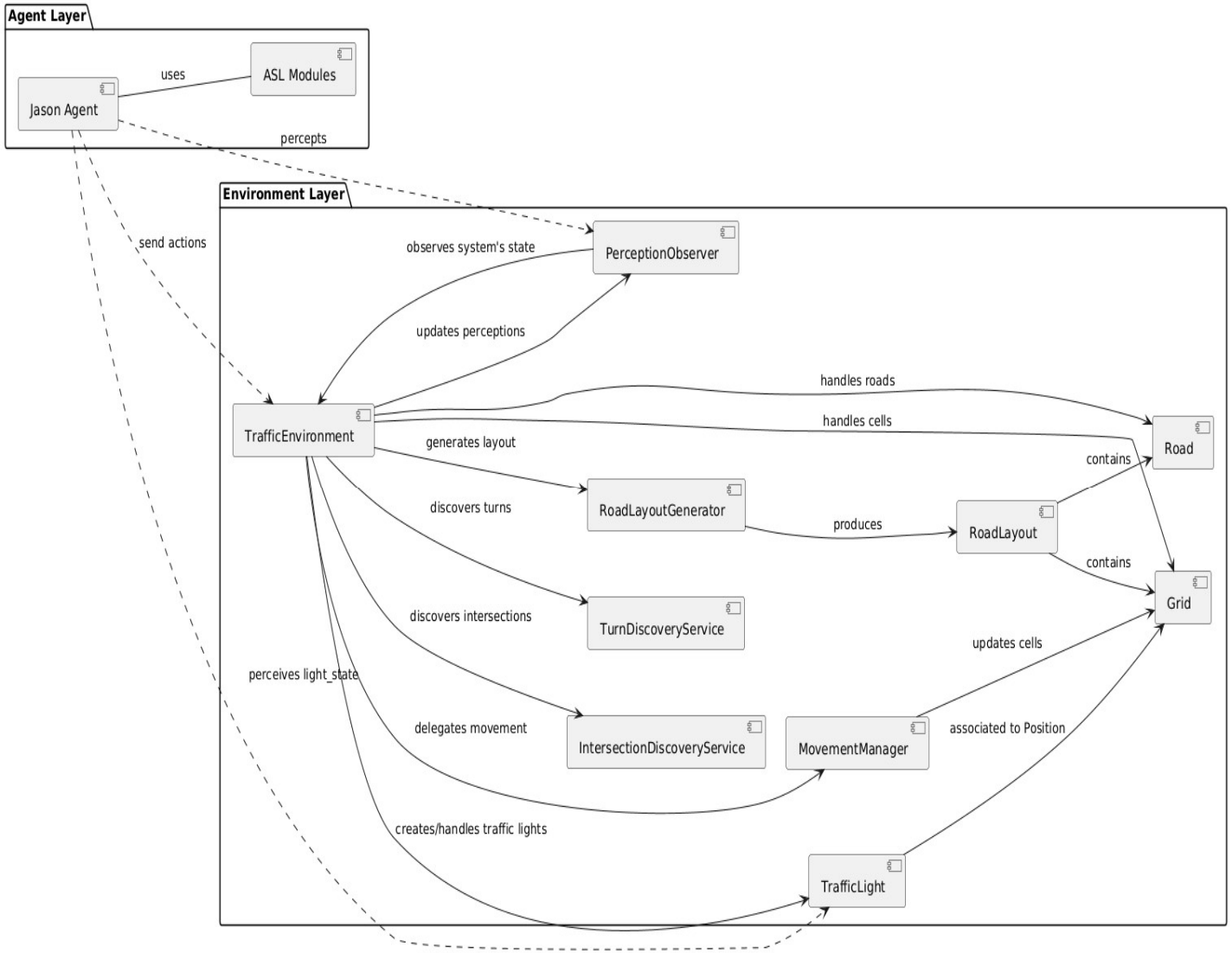
This layer involves the jason agents, so practically, it includes the .asl modules that implement the perception, decision and coordination logic. The agents interact with the environment, and percept the system's state.

Environment layer

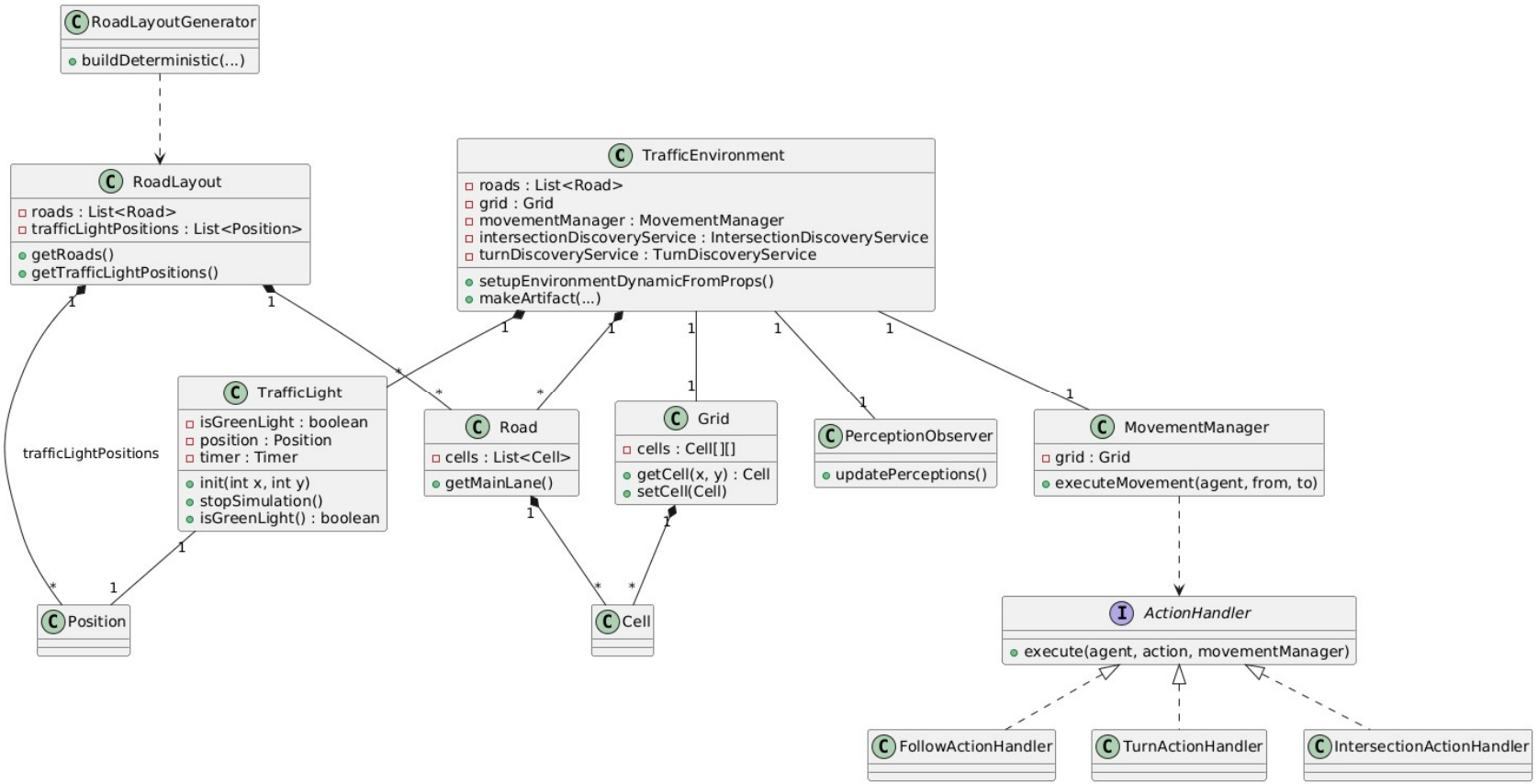
This layer comprehends java components that model the simulated reality. There are two main artifacts, which expose operations, observable properties and they have an internal space.

TrafficEnvironment → main artifact, works as a shared space for the multi-agent system;

TrafficLight → artifact that handles and coordinates traffic lights, each traffic light has its own timer.



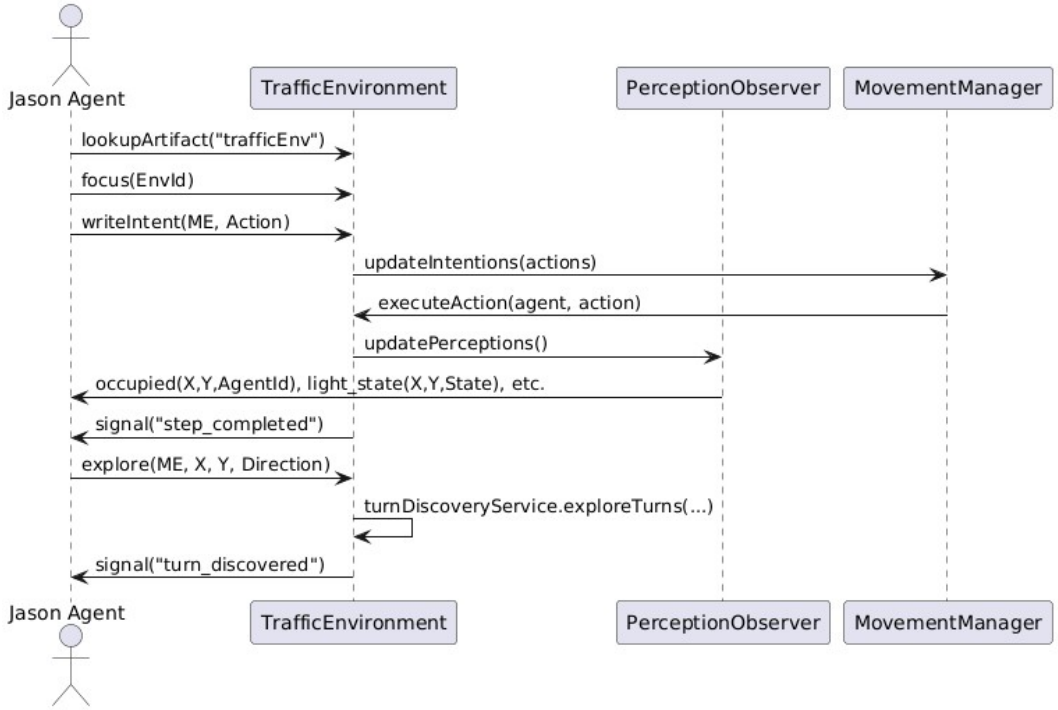
Design: Modelling



In-depth: agent behaviour

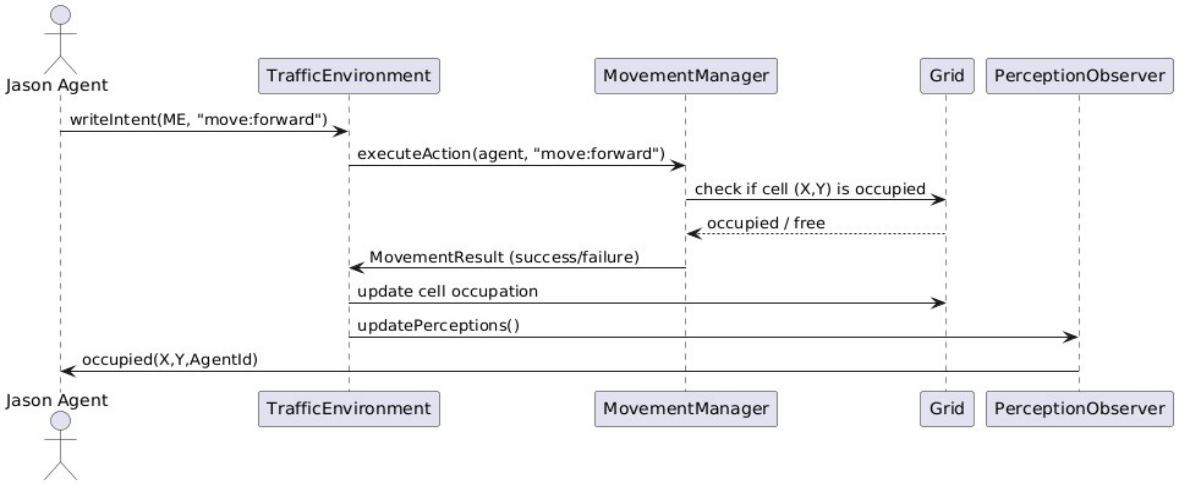
agents receive feedback from the environment, update beliefs and choose their strategy accordingly. Each time an agent (vehicle) wants to move, it writes its intention that will be elaborated from the environment with the writeIntent method, defined in the artifact.
The following image, shows a possible interaction flow, in which PerceptionObserver and Movement manager are cited:

- movementManager: centralizes and validates agent movements, ensuring safe and coordinated updates to the grid state
- perceptionObserver: aggregates and propagates environment state changes, enabling agents to perceive and react to the dynamic simulation.



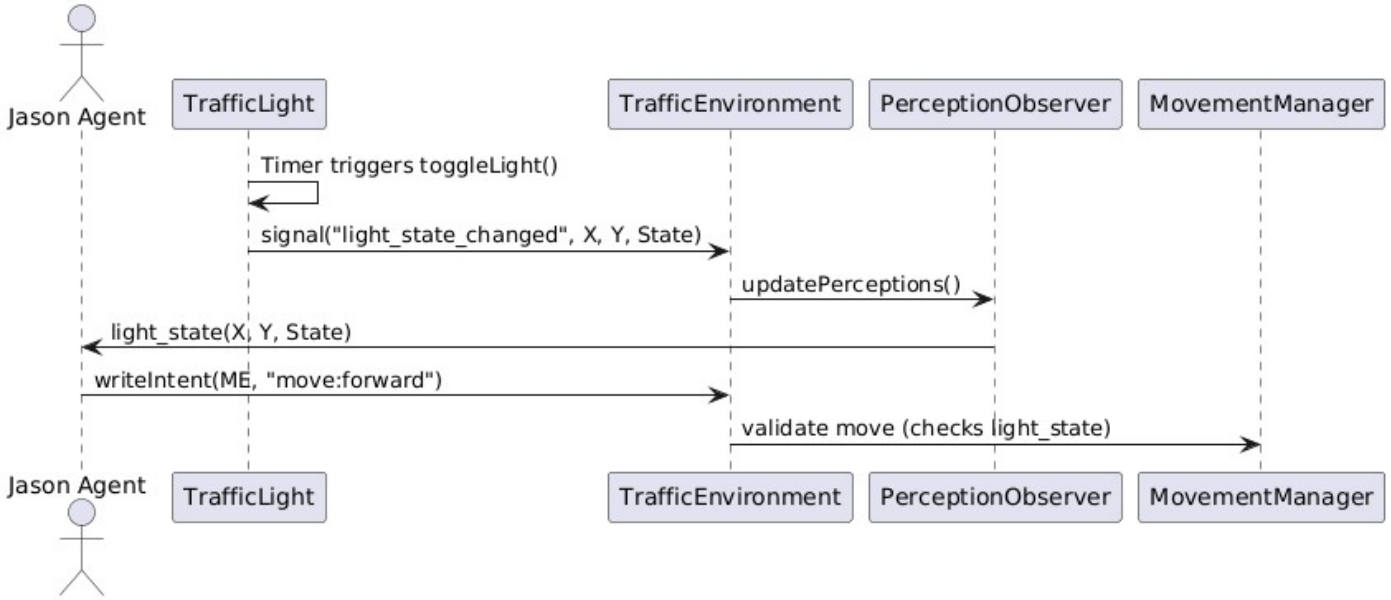
In-depth: how agents deal with an occupied cell

When multiple agents navigate a shared environment, the management of occupied cells in the grid becomes crucial for collision avoidance and effective coordination.
The interaction between movement validation and perception ensures that agents are always aware of the current occupancy of the grid, allowing them to adapt their strategies dynamically as the simulation evolves. This mechanism is possible through the occupied percept, that contains a list of occupied positions at time t of the simulation. Agents can retrieve that information to check if they're able to reach a specific position.



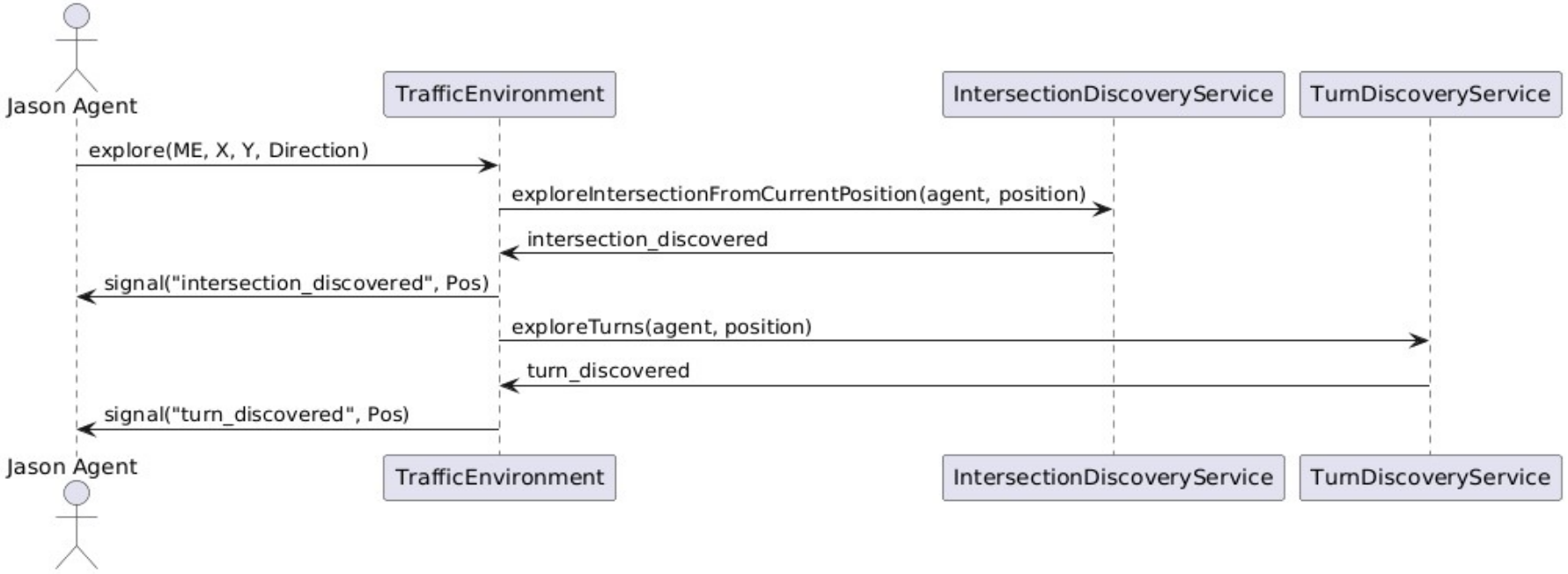
In-depth: how traffic lights work

TrafficLight is the artifact responsible for the semaphores active in the map, each one has its own timer and can be found in the grid. This sequence diagram shows how the interaction works between agent and artifact.



In-depth: turn and intersection discovery

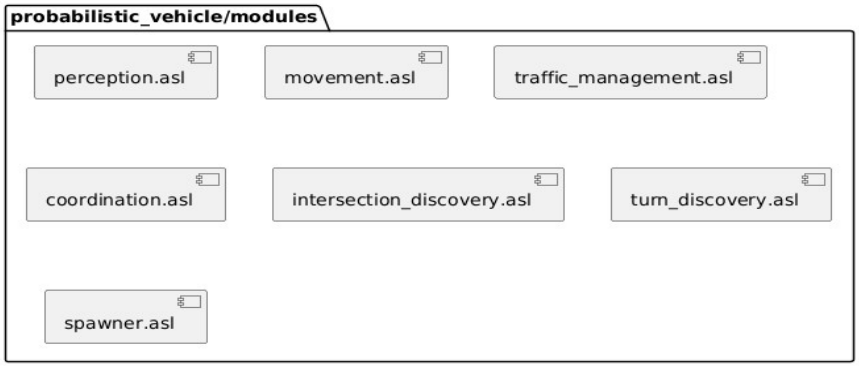
Turns and intersections are not known apriori, it's because, since it's a self-organizing system, agents should be able to move based on their own knowledge. The project's characteristics and specifications require a specific strategy in order to gain and share knowledge. Agents, as can be seen in the following sequence diagram, explore in order to find intersections, or turns, once they find them (when they are close to them), they share their knowledge to the shared space, defining a belief.



In-depth: dynamic map generation (steps)

- **Road bases selection;**
  - The roads starting coordinates (chosen either regularly or randomly) will be generated with the selectBases method, which will assure that all the roads starting points will be evenly spaced, based on a minimum Manhattan distance. This prevents overlaps and ensures a uniform or variable distribution of roads.
- **Road placement;**
  - Once the bases are defined, for each base a Road will be created with the custom Road factory method. Roads can be generated vertically or horizontally, and each one of them has two lanes, with opposite directions.
- **Intersection placement;**
  - Not only roads will be placed, so where a vertical and horizontal road overlap, an intersection is created, by adding 4 footprints in order to make agents cross the road in every possible direction;
- **Generation of standalone turns;**
  - with similar logic to intersection placement, even standalone turns will be added to the map;
- **connectivity check.**
  - As mentioned in the previous sub-section, it's verified that all the roads are connected to each other, to avoid unreachable cells. If necessary a central road is added to ensure connectivity.

Jason module architecture



- The movement module encodes the agent's reasoning for navigation and movement, inside the defined environment grid. It's a simple yet important module, and these are the main responsibilities:
  - compute and calculate next position → grid's dimensions are used to implement toroidal movement, to avoid agents getting stuck;
  - occupancy check.
- The perception module contains the code that processes percepts received from the environment artifact, in order to maintain the agent0s state up to date, its main features are the following:
  - Positional tracking, occupancy awareness, traffic light change detection, simulation percept;
- The coordination module contains methods that work toward the negotiation of movements, in order to avoid conflicts when moving in a shared space.
- The intersection discovery focuses on the exploration of intersections (same thing happens with turns), in order to gain knowledge to share with other agents. When an agent perceives a new intersection it adds the location to its beliefs, and as introduced before, it shares the knowledge via intersection_discovered belief.
- Spawner module helps the random generation, according to the roadLayoutGenerator, presented in the second chapter of the report.

Conclusion

- The project involves the creation of a modular and extensible multi-agent system, this demonstrates how distributed agents can achieve safe and efficient navigation through local perception, and the possible efficiency of cognitive stigmergy approach. This can offer a solid fundation for future works, in order to understand better how this can be useful in real-life (and possible real time) applications.

Future works

A lot of work can still be done, such as:

- adding a graphical interface;
- metrics;
- experiments and statistical significance between classical and cognitive stigmergy controllers;
- adding more features or agents;
  - pedestrians (that can also crossroads);
  - signals;
  - possible crashes;
  - speed.
- adding environmental factors, like weather, maybe based on real-life conditions;
- scalability tests;
- advanced algorithms;
- path calculation algorithms, like a* algorithm.