HLD Design Assignment 2 Advanced Topics

Idan Buskila & Ido Beerie

To ensure the algorithm makes accurate next-step decisions, we recognized the importance of having identical sensors in both the simulator and the algorithm. This led us to develop a class called `VacuumCleaner`, representing the robot itself. This class is responsible for tracking crucial data such as the robot's current position, the number of steps taken, and the battery level.

Additionally, we developed a `HouseManager` class to manage the house map that the robot builds as it explores and cleans. The `HouseManager` maintains a map and a set: the map stores only the cells visited by the robot, while the set tracks all accessible cells discovered using its wall sensor to identify neighbouring cells that are not walls. This method ensures that any cell in the set is reachable and permissible for the robot. The `HouseManager` data is utilised exclusively within the algorithm class.

The core of our solution lies in the `MyAlgorithm` class, which implements a depth-first search (DFS) method to explore the house. The algorithm uses an enum `AlgoState` to manage the different states of the vacuum cleaner:

- **INIT**: The initial state where the robot starts exploring and cleaning.

- **TO_DOCK**: The state where the robot is returning to the docking station to recharge.

- **CHARGING**: The state where the robot is recharging its battery.

- **TO_POS**: The state where the robot is moving to a new position to continue cleaning.

Algorithm Details

Initialization

The `MyAlgorithm` constructor initialises the starting position of the vacuum cleaner and marks the initial position as visited in the `HouseManager`.

Sensor Setup

The algorithm provides methods to set up various sensors, including the battery-meter, walls sensor, and dirt sensor. These sensors are essential for the robot to perceive its environment and make decisions accordingly.

Scanning and Moving

The `scanNeighbors` method checks the neighbouring cells using the wall sensor and updates the `HouseManager` with cells that need to be visited.

The `moveByState` method handles the robot's movement based on its current state (`AlgoState`). It ensures the robot follows the appropriate actions depending on whether it needs to dock, charge, or move to a new position.

Main Algorithm Logic

The `nextStep` method is the main logic of the algorithm. It manages the robot's decisions to move, clean, or return to the docking station based on the current state, battery level, and dirt levels.

Conclusion

The design ensures a clear separation of responsibilities and efficient operation of each component. The `VacuumCleaner` class tracks the robot's state, the `HouseManager` class manages the mapping and cleaning status of the house, and the `MyAlgorithm` class implements the logic for the robot's actions using an enum `AlgoState` to manage different states. This modular design allows for easy maintenance and scalability while ensuring that the robot can effectively navigate and clean the house.