Mars Exploration Environment:
A Logic-Based Autonomous Agent System

Mahla Entezari

Fall 2024

Abstract

This report presents the development of a Mars exploration simulation using a grid-based environment and a knowledge-based agent implemented via first-order logic (FOL). The environment includes hazards (holes), resources (goods), and a partially observable state space. The agent utilizes a deterministic depth-first search (DFS) strategy informed by logical inference rules to safely navigate and collect all resources. Visualization is implemented using Pygame, and agent decisions are grounded in formal logical expressions.

1 Introduction

Exploring hazardous and uncertain terrains such as those found on Mars requires agents capable of logical reasoning, partial observability handling, and safe decision-making. In this project, we simulate such an environment and propose a logical agent that models its behavior using a formal predicate logic system. The aim is to collect all resources (goods) while avoiding hazards (holes) in a grid environment.

2 System Overview

2.1 Environment Structure

The environment is a $H \times W$ grid initialized with:

- A single agent at position $(0, 0)$ or randomized.
- Randomly placed **holes** (hazards).
- Randomly placed **goods** (collectible resources).

Each cell in the grid can be in one and only one of the following states:

- Empty
- Contains a Hole
- Contains a Good

2.2 Core Logical Constraints

- **State Exclusivity:**

$$\forall c \in \mathrm{Grid}, (\mathrm{Hole}(c) \to \neg\mathrm{Good}(c) \wedge \neg\mathrm{Empty}(c) \wedge \mathrm{Good}(c) \to \neg\mathrm{Hole}(c) \wedge \neg\mathrm{Empty}(c))$$

- **Safe Movement:**

$$\mathrm{SafeMove}(d) \equiv \exists c \in \mathrm{Adjacent}(d), \neg\mathrm{Hole}(c)$$

- **Collect Good:**

$$\mathrm{CollectGood}(d) \equiv \exists c \in \mathrm{Adjacent}(d), \mathrm{Good}(c)$$

- **Termination:**

$$\mathrm{GameOver} \equiv \exists c, (\mathrm{AtAgent}(c) \wedge \mathrm{Hole}(c)) \vee (\forall c, \neg\mathrm{Good}(c))$$

3 Agent Design

3.1 Logic and Policy

The agent is governed by first-order logic predicates:

- $A_1(b)$: Is there a hole in block b?
- $A_2(b)$: Is there a good in block b?

Suitability Rule:

$$A_2(b) \vee (\neg A_1(b) \wedge \forall r \in \mathrm{Neighbors}(b), \neg A_2(r))$$

This rule ensures the agent either:

1. Moves toward goods directly.
2. Chooses a safe unexplored cell when no immediate goods are detected.

3.2 DFS Strategy

A recursive Depth-First Search is used. The agent:

- Marks the current cell as seen.
- Evaluates all valid adjacent cells based on logical suitability.
- Recursively visits suitable and unseen neighbors.
- Backtracks if no progress is possible.

4 Implementation Details

4.1 Environment Class

Key methods of the `Mars_Exploration_ENV` class include:

- `init_grid()`: Randomly populates the grid.
- `get_adjacent_blocks()`: Returns adjacent states.
- `take_action(action)`: Updates the agent's position.
- `update_env()`: Pygame-based rendering.

4.2 Agent Class

From `agent.py`:

Listing 1: Suitability Evaluation

```
selected = block.isGood() or (
    not block.isHole() and
    not self._disjuntion(call_method_on_objects(r_neig, "isGood"))
)
```

4.3 Execution Entry

Listing 2: Main Agent Execution

```
if __name__ == "__main__":
    env = Mars_Exploration_ENV(grid_h=15, grid_w=15, num_hol=20,
       num_good=20)
    agent = FOL_Agent(env)
    agent.dfs(agent.env.get_current_position(), agent.env.
       get_adjacent_blocks())
```

5 Visualization

Real-time interaction is visualized via `pygame`. The agent is a green circle, holes are brown, and goods are blue crystals.

6 Conclusion

This project demonstrates the application of first-order logic for autonomous exploration in a simulated Mars terrain. The agent's reasoning process ensures safe movement and efficient resource collection without prior map knowledge, relying entirely on local logical inference and recursive search.

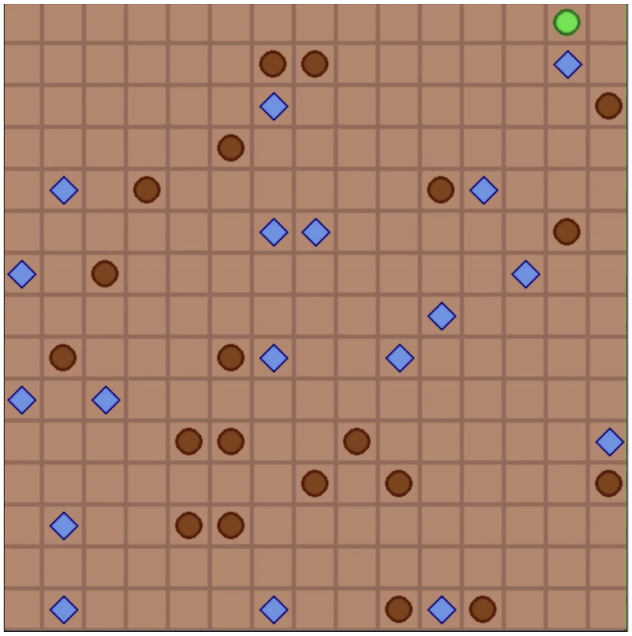

Figure 1: Start Simulation Snapshot

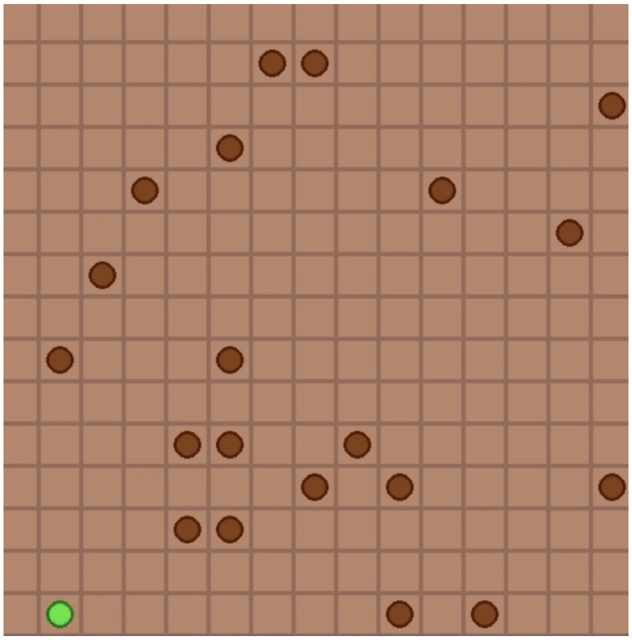

Figure 2: End Simulation Snapshot