جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN UNIVERSITY

SCHOOL OF SCIENCE & ENGINEERING

FALL 2023

CSC 4301 Introduction to Artificial Intelligence

Term Project: 2

Wumpus World

November 2nd, 2023

Realized by:

Khadija Salih Alj

Noura Ogbi

Younes Bouziane

Supervised by:

Prof. Tajjedine Rachidi

Table of Content

Introduction 3
I. Wumpus Game Environment 4
II. JESS Environment 6
III. Task 1: Improve the Hunter's Reasoning Ability 7
IV. Task 2: New Top-Level Goal – Killing the Wumpus 15
V. Task 3: Improve the Hunter's Ability to Go to a Given Location 22
VI. Task 4: Extra Credit: Allow the Hunter to Choose the Best Action 24
VII. Task 5: Extra Credit: Evaluate the Precision of the Agent 27
Conclusion 45
References 47

Introduction

In Artificial Intelligence, we often turn to captivating challenges to push the boundaries of what is possible. One challenge represented through the classic game of "Wumpus," where the primary mission is to locate and eliminate the Wumpus creature with an arrow, all while sidestepping the risky traps and pitfalls that threaten the progress.

Our project takes this timeless game and transforms it into a platform to explore and expand upon key AI principles, more specifically gaming agents. We employ the capabilities of JESS, a robust rule-based expert system shell, to craft a sophisticated framework that mirrors the intricate mechanics of the game. These rules guide the decision-making process, enhancing the player's ability to strategize and adapt.

In this project, we embark on five central tasks that aim to elevate the Hunter's abilities and the overall gameplay experience.

In task 1, our initial focus is to augment the Hunter's deductive capabilities. We aim to go beyond simply marking caves as "possibly/maybe" containing the Wumpus. Instead, we seek to empower the system to deduce the exact location of the Wumpus when sufficient information is available. This task extends to pit detection as well, enhancing the Hunter's precision in navigating the Wumpus World.

In task 2, our Hunter's objective is no longer limited to gold retrieval. We introduce an additional goal: eliminating the Wumpus. To achieve this, we equip the Hunter with a new action, 'shoot,' which consumes an arrow. If the Hunter finds itself adjacent to a cave with a living Wumpus, it can take action to shoot it! Dead Wumpus may still stink, but they pose no danger.

In task 3, we observe the Hunter's performance in different cave scenarios and identify areas where it may get stuck. This task involves enhancing its ability to set goals and navigate to distant caves more effectively. The goal is to make the Hunter a more dynamic and adaptive explorer.

In task 4, an extra challenge, we delve into the world of decision-making. By modifying rules that allow the Hunter to prioritize an action when faced with multiple possibilities.

In task 5, to provide a comprehensive assessment of our modified agent's performance, we develop test our system in various cave configurations. We take note of key outcomes such as picking up gold and Wumpus elimination success, offering valuable insights into our agent's precision.

With these tasks, we venture into the heart of the "Wumpus" game, employing AI principles to enrich the experience. Each task represents a distinct facet of our project, collectively forming an adventure of exploration, deduction, and strategic decision-making. Together, they embody the essence of artificial intelligence, problem-solving, and the thrill of gaming.

I. Wumpus Game Environment

The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game Hunt the Wumpus by Gregory Yob in 1973.[1]

The Wumpus world comprises a cave with rooms interconnected by passageways. Our intelligent agent ventures into this labyrinthine world armed with knowledge (which are the rules of the game and observations) and one arrow. Within this cave, there resides a menacing creature known as the Wumpus, capable of devouring anyone who enters its domain. The agent possesses a single arrow to potentially eliminate the Wumpus. Additionally, there are some pits that lead to a fatal "Game Over".

The agent's mission is clear: navigate the cave, locate the gold, kill the Wumpus, and exit without falling in a pit or becoming the Wumpus' prey. Successfully retrieving the gold rewards the agent, while falling into a Pit or being devoured by the Wumpus incurs penalties (even ending the game).

The Wumpus world Properties:

- ✓ **Partially observable**: The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
- ✓ **Deterministic**: It is deterministic, as the result and outcome of the world are already known.
- ✓ **Sequential**: The order is important, so it is sequential.
- ✓ **Static**: It is static as Wumpus and Pits are not moving.
- ✓ **Discrete**: The environment is discrete.
- ✓ **One agent**: The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

Sensors:

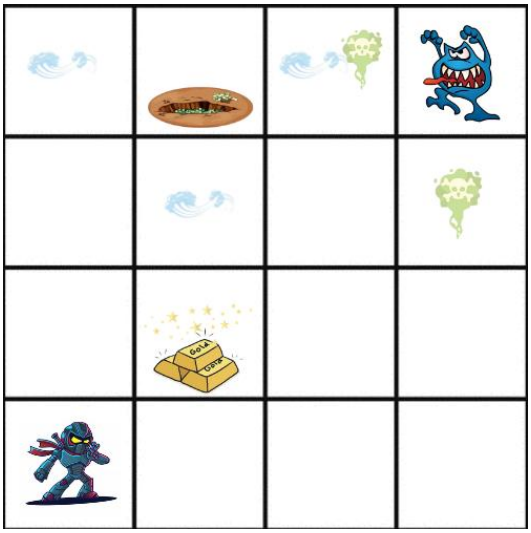
- ✓ The agent will perceive the **stench** if it is in an adjacent cave to the Wumpus (not diagonally).
- ✓ The agent will perceive **breeze** if it is in a cave directly adjacent to the pit.
- ✓ The agent will perceive the **glitter** in the cave where the gold is present.
- ✓ The agent will perceive the **bump** if he walks into a wall.
- ✓ When the Wumpus is shot, it emits a horrible **scream** which can be perceived anywhere in the cave.
- ✓ These percepts can be represented as five element list, in which we will have different indicators for each sensor. Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as: **[Stench, Breeze, None, None, None].**

[1] (The Wumpus World in Artificial intelligence, n.d.)

Caves in this project:

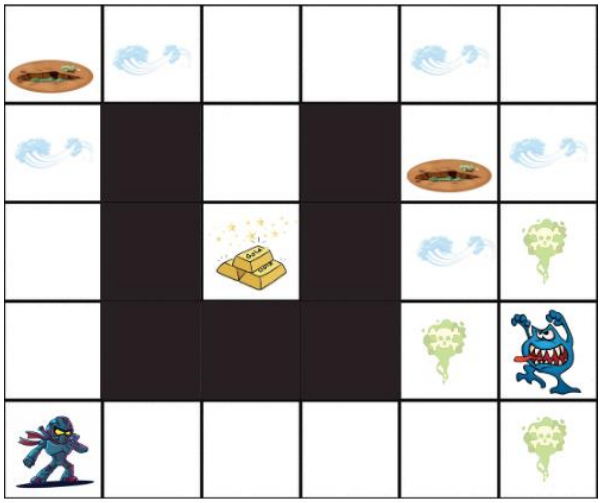
Cave0:

Cave1:



Cave2:

Cave3:



II. JESS Environment

JESS is an acronym for Java Expert System Shell, serving as both an expert system shell and a scripting language developed entirely in the Java programming language. It is commonly employed for constructing rule-based expert systems closely integrated with Java-based systems. These expert systems created with JESS can be executed through a command-line interface or utilized as an Applet.[2]

Benefits of JESS:

- Better to be applied in bigger problems where performance is dominated by algorithm quality.
- Include backward chaining and working memory queries.
- Faster than the expert systems developed using expert system shells written in C language.
- Contains some commands which allow less memory usage when executing the system.[2]

Indeed, as we delve into this project, it becomes evident that the JESS engine is an invaluable asset, as it serves as a potent tool for crafting intelligent systems capable of tackling intricate problems and making decisions grounded in logical rules. The speciality of JESS lies in its ability to perform inferencing, making it a suitable platform for constructing expert systems that exhibit profound reasoning capabilities.

At its core, the JESS engine operates by building a knowledge base comprising rules and facts, meticulously tailored to a specific domain. Rules in JESS are the guiding principles, consisting of a set of conditions and corresponding actions. These conditions act as the gatekeepers of logic, while the actions dictate what happens when specific conditions are met.

In inferencing, the JESS engine undertakes the task of matching these conditions within each rule against the facts residing within the knowledge base. When a rule's conditions align with the facts, it triggers the execution of some predefined actions. The execution of actions can also lead to the generation of new facts, setting in motion a dynamic chain reaction of rules and conclusions. This iterative process persists until no more rules can be executed…

[2] Siriwardhane, P. (2022, February 2). Retrieved October 28, 2023, from Medium: https://medium.com/nerd-for-tech/an-introduction-to-expert-system-shells-530043914ec0

III. Task 1: Improve the Hunter's Reasoning Ability

Objective:

In this task, our primary objective is to enhance the reasoning ability of the Hunter within the Wumpus game. To accomplish this, we employ the capabilities of the JESS expert system shell, creating a set of rules that allow the Hunter to deduce the exact location of the Wumpus and potentially pits.

Implementation for Wumpus:

```
;; Start Change Task 1

(defquery find-wumpus-location
  (declare (variables ?x ?y))
  (adj ?x ?y ?x2 ?y2)
  (and
    (not (cave (x ?x2) (y ?y2) (has-pit TRUE)))
    (not (cave (x ?x2) (y ?y2) (has-gold TRUE)))
    (not (cave (x ?x2) (y ?y2) (visited TRUE)))))

(defrule final-wumpus-location
  (task think)
  (cave (x ?x)(y ?y)(stench TRUE))
  (hunter (agent ?a)(x ?x1)(y ?y1))
  (adj ?x ?y ?x2 ?y2)
  ?f <- (cave (x ?x2)(y ?y2)(has-wumpus MAYBE))
  (test (= 1 (count-query-results find-wumpus-location ?x ?y)))
  (wumpus (x ?x2) (y ?y2) (alive TRUE))
  =>
  (printout t "The wumpus is in (" ?x2 "," ?y2 ")." crlf)
  (modify ?f (has-wumpus TRUE) (safe FALSE))
  )

;; End Change Task 1
```

- **Defining the Query find-wumpus-location:**

This query is created to search for potential locations of the Wumpus in adjacent caves when certain conditions are met. It uses the following variables:

- **?x and ?y**: These are variables declared for the coordinates of the current cave.
- **?x2 and ?y2**: These are variables representing the coordinates of adjacent caves.

The query consists of the following conditions:

- **(adj ?x ?y ?x2 ?y2)**: It checks whether the current cave is adjacent to another cave, with coordinates ?x2 and ?y2.
- **(and ...)**: This is a logical "and" operator that combines several conditions. It checks three things for the adjacent cave:

- **(not (cave (x ?x2) (y ?y2) (has-pit TRUE)))**: Ensures that the adjacent cave does not have a pit.
- **(not (cave (x ?x2) (y ?y2) (has-gold TRUE)))**: Ensures that the adjacent cave does not have gold.
- **(not (cave (x ?x2) (y ?y2) (visited TRUE)))**: Ensures that the adjacent cave has not been visited before.
- **We did not do (cave (x ?x2) (y ?y2) (visited FALSE)) because we need to keep track on anything besides TRUE (i.e., MAYBE case 😉)**

➢ **Defining the Rule - final-wumpus-location:**

This rule is created to make inferences about the potential location of the Wumpus when the Hunter detects a stench in the current cave.

- **(task think)**: The rule activates when the Hunter is in the "think" task, indicating that it is making decisions and deductions.
- **(cave (x ?x)(y ?y)(stench TRUE))**: This condition checks whether the current cave has a stench, indicating the possible presence of the Wumpus.
- **(hunter (agent ?a)(x ?x1)(y ?y1))**: It captures the Hunter's details, including its agent name and coordinates.
- **(adj ?x ?y ?x2 ?y2)**: Checks if the current cave is adjacent to another cave with coordinates ?x2 and ?y2.
- **?f <- (cave (x ?x2)(y ?y2)(has-wumpus MAYBE))**: The rule introduces a pattern to create a variable ?f that matches caves that might have a Wumpus ("MAYBE").
- **(test (= 1 (count-query-results find-wumpus ?x ?y)))**: This is a test condition. It checks if there is only one adjacent cave where the find-wumpus-location query conditions are met, indicating a potential location for the Wumpus.
- **(wumpus (x ?x2) (y ?y2) (alive TRUE))**: This condition ensures that the Wumpus in the adjacent cave is still alive.

The => symbol marks the actions to be taken if all the conditions are met.

➢ **(printout t "The wumpus is in (" ?x2 "," ?y2 ")." crlf)**: If the conditions are satisfied, the rule prints out the exact location of the Wumpus.
➢ **(modify ?f (has-wumpus TRUE) (safe FALSE))**: Finally, the rule modifies the information associated with the adjacent cave, confirming that it indeed has the Wumpus (TRUE) and marking it as unsafe (FALSE).

In summary, this rule enhances the Hunter's reasoning ability by allowing it to deduce the exact location of the Wumpus when possible. It does so by examining the stench in the current cave, checking adjacent caves for conditions that exclude pits, gold, and previous visits, and confirming the presence of a live Wumpus in a cave where no other adjacent cave meets these conditions. When such a scenario is identified, the rule provides the precise location of the Wumpus and updates the information accordingly.

Implementation for Pit:

```
;; Start Change Task 1

(defquery find-pit-location
  (declare (variables ?x ?y))
  (adj ?x ?y ?x2 ?y2)
  (and (not (cave (x ?x2) (y ?y2) (has-wumpus TRUE)))
       (not (cave (x ?x2) (y ?y2) (has-gold TRUE)))
       (not (cave (x ?x2) (y ?y2) (visited TRUE)))))

(defrule final-pit-location
  (task think)
  (cave (x ?x)(y ?y)(breeze TRUE))
  (adj ?x ?y ?x2 ?y2)
  ?f <- (cave (x ?x2)(y ?y2)(has-pit MAYBE))
  (test (= 1 (count-query-results find-pit-location ?x ?y)))
  =>
  (printout t "The pit is in (" ?x2 "," ?y2 ")." crlf)
  (modify ?f (has-pit TRUE) (safe FALSE)))

;; End Change Task 1
```

The above piece of code is quite similar to the previous one (finding wumpus), with the main difference being that it is designed to locate pits in the game environment.

➢ Defining the Query find-pit-location:

This query is created to search for potential locations of pits in adjacent caves when certain conditions are met. It uses the same variables as before…

- **?x and ?y**: Variables for the coordinates of the current cave.
- **?x2 and ?y2**: Variables representing the coordinates of adjacent caves.

The query consists of the following conditions:

- **(adj ?x ?y ?x2 ?y2)**: It checks whether the current cave is adjacent to another cave, with coordinates ?x2 and ?y2.
- **(and ...)**: This is a logical “and” operator that combines several conditions. It checks three things for the adjacent cave:
  - **(not (cave (x ?x2) (y ?y2) (has-wumpus TRUE)))**: Ensures that the adjacent cave does not have a Wumpus.
  - **(not (cave (x ?x2) (y ?y2) (has-gold TRUE)))**: Ensures that the adjacent cave does not have gold.
  - **(not (cave (x ?x2) (y ?y2) (visited TRUE)))**: Ensures that the adjacent cave has not been visited before.

- **Defining the Rule - final-pit-location:**

This rule is created to make inferences about the potential location of pits when the Hunter detects a breeze in the current cave. It works in a similar way to the previous rule:

- **(task think)**: The rule activates when the Hunter is in the "think" task, indicating decision-making and reasoning.
- **(cave (x ?x)(y ?y)(breeze TRUE))**: This condition checks whether the current cave has a breeze, indicating the possible presence of a pit.
- **(adj ?x ?y ?x2 ?y2)**: Checks if the current cave is adjacent to another cave with coordinates ?x2 and ?y2.
- **?f <- (cave (x ?x2)(y ?y2)(has-pit MAYBE))**: The rule introduces a pattern to create a variable ?f that matches caves that might have a pit ("MAYBE").
- **(test (= 1 (count-query-results find-pit-location ?x ?y)))**: This is a test condition. It checks if there is only one adjacent cave where the find-pit-location query conditions are met, indicating a potential location for a pit.

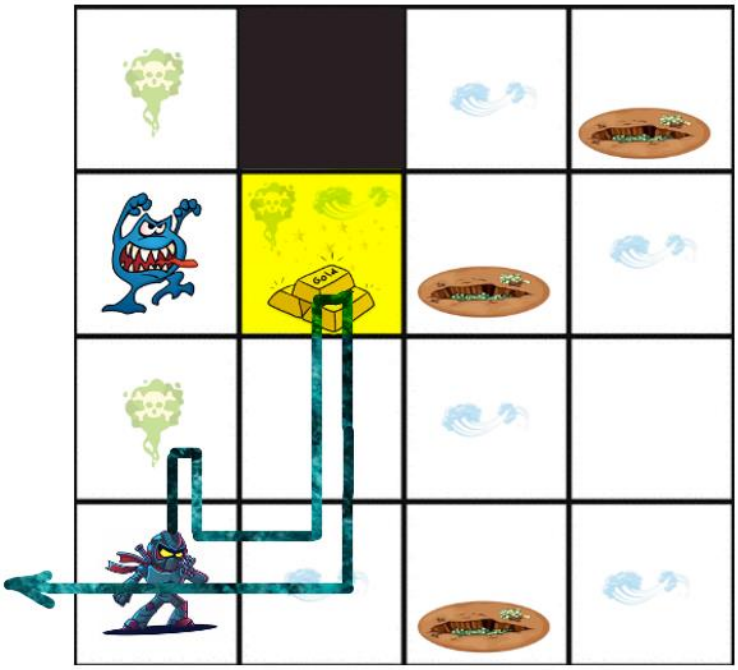
The => symbol marks the actions to be taken if all the conditions are met.

- **(printout t "The pit is in (" ?x2 "," ?y2 ")." crlf)**: If the conditions are satisfied, the rule prints out the exact location of the pit.
- **(modify ?f (has-pit TRUE) (safe FALSE))**: Finally, the rule modifies the information associated with the adjacent cave, confirming that it indeed has a pit (TRUE) and marking it as unsafe (FALSE).

This rule enhances the Hunter's reasoning ability by allowing it to deduce the exact location of pits when possible. It does so by examining the breeze in the current cave, checking adjacent caves for conditions that exclude Wumpus, gold, and previous visits, and confirming the presence of a pit in a cave where no other adjacent cave meets these conditions. When such a scenario is identified, the rule provides the precise location of the pit and updates the information accordingly.

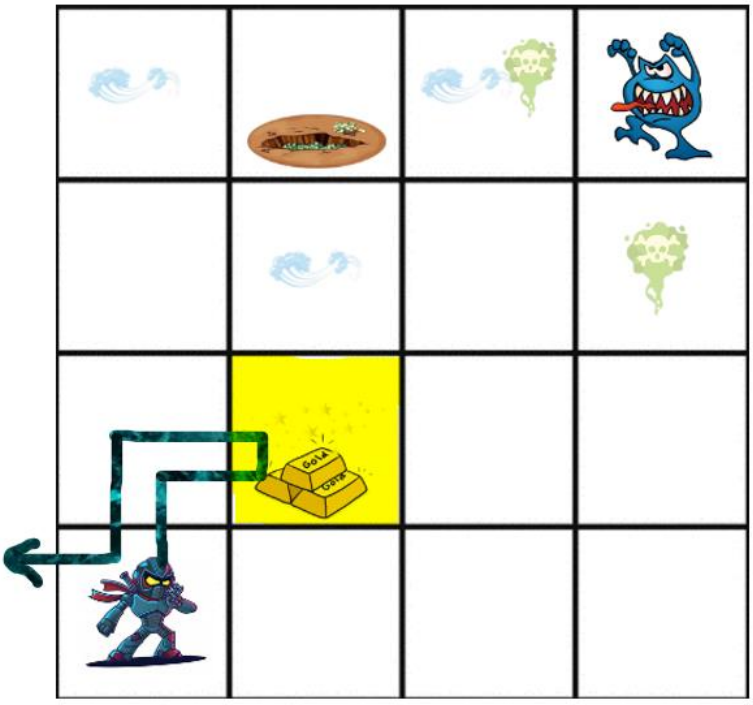
Showing it works:

Cave0: (Score=162)



```
The wumpus is in (1,3).  The pit is in (3,1).  The pit is in (3,3).
```

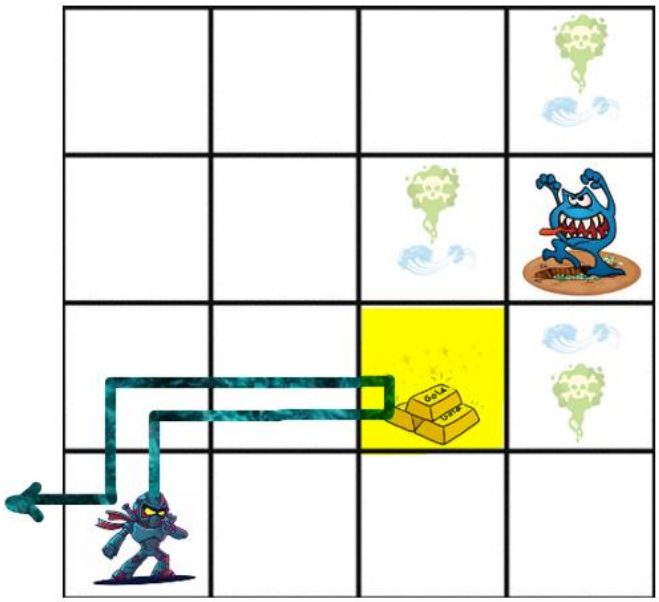
Cave1: (Score=118) **No sensed Stench/Breeze ⟹ No encountered Wumpus/Pit!**



```
GENESIS...
Actaeon enters the caves at (1,1).
Adding adj asserts for a 4 by 4  world.
SIMULATING...
SENSING...
Actaeon notices (1,2) nearby.
Actaeon notices (2,1) nearby.
Actaeon sees no glitter.
Actaeon smells nothing.
Actaeon feels no breeze in (1,1).
THINKING...
No stench in (1,1) means no wumpus in (2,1).
There's no breeze in (1,1) so there's no pit  in (2,1).
Actaeon somewhat wants to go to (2,1).
With neither wumpus nor pit, (2,1) is safe.
Actaeon strongly wants to go to (2,1).
There's no breeze in (1,1) so there's no pit  in (1,2).
No stench in (1,1) means no wumpus in (1,2).
Actaeon somewhat wants to go to (1,2).
With neither wumpus nor pit, (1,2) is safe.
Actaeon strongly wants to go to (1,2).
Since Actaeon is in (1,1) and not dead, it must be safe.
Seeing no glitter, Actaeon knows there is no gold in (1,1).
(1,1) is safe, so there's no pit or wumpus in it.
PLANNING...
ACTING...
Actaeon goes to (1,2).
SIMULATING...
SENSING...
Actaeon sees no glitter.
Actaeon notices (1,3) nearby.
Actaeon notices (2,2) nearby.
Actaeon smells nothing.
Actaeon feels no breeze in (1,2).
THINKING...
Actaeon moderately wants to go to (2,1).
No stench in (1,2) means no wumpus in (1,3).
There's no breeze in (1,2) so there's no pit  in (1,3).
Actaeon somewhat wants to go to (1,3).
With neither wumpus nor pit, (1,3) is safe.
Actaeon strongly wants to go to (1,3).
No stench in (1,2) means no wumpus in (2,2).
There's no breeze in (1,2) so there's no pit  in (2,2).
Actaeon somewhat wants to go to (2,2).
With neither wumpus nor pit, (2,2) is safe.
Actaeon strongly wants to go to (2,2).
Seeing no glitter, Actaeon knows there is no gold in (1,2).
PLANNING...
ACTING...
Actaeon goes to (2,2).
SIMULATING...
SENSING...
Actaeon sees glitter.
Actaeon notices (2,3) nearby.
Actaeon notices (3,2) nearby.
Actaeon smells nothing.
Actaeon feels no breeze in (2,2).
THINKING...
Actaeon moderately wants to go to (1,3).
No stench in (2,2) means no wumpus in (2,3).
There's no breeze in (2,2) so there's no pit  in (2,3).
Actaeon somewhat wants to go to (2,3).
With neither wumpus nor pit, (2,3) is safe.
Actaeon strongly wants to go to (2,3).
There's no breeze in (2,2) so there's no pit  in (3,2).
No stench in (2,2) means no wumpus in (3,2).
Actaeon somewhat wants to go to (3,2).
With neither wumpus nor pit, (3,2) is safe.
Actaeon strongly wants to go to (3,2).
Seeing glitter, Actaeon knows there is gold in (2,2).
```

```
Seeing glitter, Actaeon knows there is gold in (2,2).
Actaeon wants to pick up the gold in (2,2).
Actaeon strongly wants to go to (2,1).
PLANNING...
ACTING...
Actaeon picks up 100 pieces of gold!
SIMULATING...
SENSING...
THINKING...
Actaeon strongly wants to go to (1,2).
Actaeon moderately wants to go to (1,3).
Actaeon strongly wants to go to (3,2).
Actaeon strongly wants to go to (2,3).
Actaeon strongly wants to go to (2,1).
PLANNING...
ACTING...
Actaeon goes to (1,2).
SIMULATING...
SENSING...
THINKING...
Actaeon moderately wants to go to (3,2).
Actaeon moderately wants to go to (2,3).
Actaeon strongly wants to go to (1,1).
Actaeon moderately wants to go to (2,1).
Actaeon strongly wants to go to (1,3).
PLANNING...
ACTING...
Actaeon goes to (1,1).
Actaeon leaves the caves.
118
```

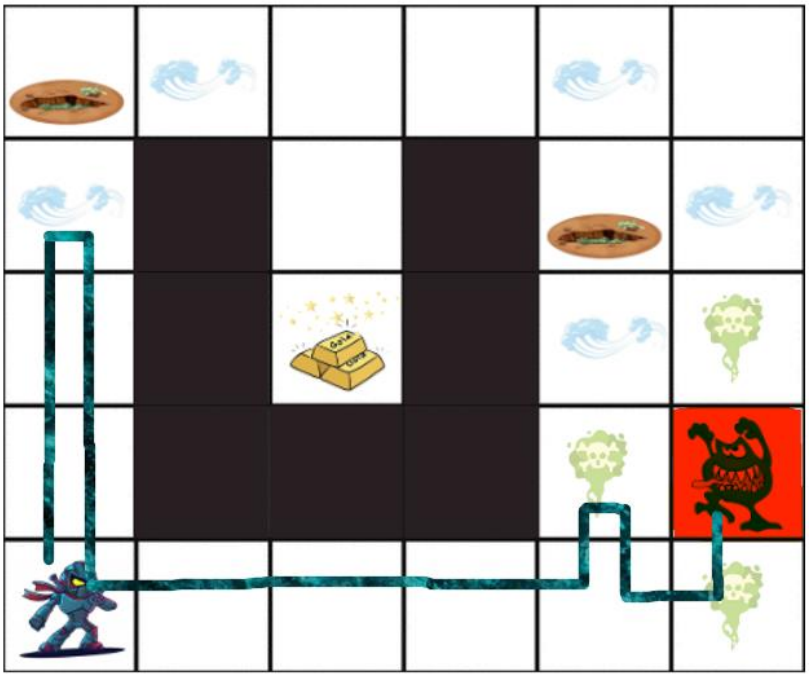
Cave2: (Score=184) **No Sensed Stench/Breeze ⟹ No encountered Wumpus/Pit!**



```
GENESIS...
Orion enters the caves at (1,1).
Adding adj asserts for a 4 by 4  world.
SIMULATING...
SENSING...
Orion notices (1,2) nearby.
Orion notices (2,1) nearby.
Orion sees no glitter.
Orion smells nothing.
Orion feels no breeze in (1,1).
THINKING...
No stench in (1,1) means no wumpus in (2,1).
There's no breeze in (1,1) so there's no pit  in (2,1).
Orion somewhat wants to go to (2,1).
With neither wumpus nor pit, (2,1) is safe.
Orion strongly wants to go to (2,1).
There's no breeze in (1,1) so there's no pit  in (1,2).
No stench in (1,1) means no wumpus in (1,2).
Orion somewhat wants to go to (1,2).
With neither wumpus nor pit, (1,2) is safe.
Orion strongly wants to go to (1,2).
Since Orion is in (1,1) and not dead, it must be safe.
Seeing no glitter, Orion knows there is no gold in (1,1).
(1,1) is safe, so there's no pit or wumpus in it.
PLANNING...
ACTING...
Orion goes to (1,2).
SIMULATING...
SENSING...
Orion sees no glitter.
Orion notices (1,3) nearby.
Orion notices (2,2) nearby.
Orion smells nothing.
Orion feels no breeze in (1,2).
THINKING...
Orion moderately wants to go to (2,1).
No stench in (1,2) means no wumpus in (1,3).
There's no breeze in (1,2) so there's no pit  in (1,3).
Orion somewhat wants to go to (1,3).
With neither wumpus nor pit, (1,3) is safe.
Orion strongly wants to go to (1,3).
No stench in (1,2) means no wumpus in (2,2).
There's no breeze in (1,2) so there's no pit  in (2,2).
Orion somewhat wants to go to (2,2).
With neither wumpus nor pit, (2,2) is safe.
Orion strongly wants to go to (2,2).
Seeing no glitter, Orion knows there is no gold in (1,2).
PLANNING...
ACTING...
Orion goes to (2,2).
SIMULATING...
SENSING...
Orion sees no glitter.
Orion notices (2,3) nearby.
Orion notices (3,2) nearby.
Orion smells nothing.
Orion feels no breeze in (2,2).
THINKING...
Orion moderately wants to go to (1,3).
No stench in (2,2) means no wumpus in (2,3).
There's no breeze in (2,2) so there's no pit  in (2,3).
```

```
There's no breeze in (2,2) so there's no pit  in (2,3).
Orion somewhat wants to go to (2,3).
With neither wumpus nor pit, (2,3) is safe.
Orion strongly wants to go to (2,3).
There's no breeze in (2,2) so there's no pit  in (3,2).
No stench in (2,2) means no wumpus in (3,2).
Orion somewhat wants to go to (3,2).
With neither wumpus nor pit, (3,2) is safe.
Orion strongly wants to go to (3,2).
Seeing no glitter, Orion knows there is no gold in (2,2).
Orion strongly wants to go to (2,1).
PLANNING...
ACTING...
Orion goes to (3,2).
SIMULATING...
SENSING...
Orion sees glitter.
Orion notices (3,3) nearby.
Orion notices (4,2) nearby.
Orion notices (3,1) nearby.
Orion feels no breeze in (3,2).
Orion smells nothing.
THINKING...
Orion moderately wants to go to (2,3).
Orion moderately wants to go to (1,3).
Orion moderately wants to go to (2,1).
There's no breeze in (3,2) so there's no pit  in (3,1).
Orion somewhat wants to go to (3,1).
No stench in (3,2) means no wumpus in (3,1).
Orion somewhat wants to go to (3,1).
With neither wumpus nor pit, (3,1) is safe.
Orion strongly wants to go to (3,1).
There's no breeze in (3,2) so there's no pit  in (4,2).
No stench in (3,2) means no wumpus in (4,2).
Orion somewhat wants to go to (4,2).
With neither wumpus nor pit, (4,2) is safe.
Orion strongly wants to go to (4,2).
No stench in (3,2) means no wumpus in (3,3).
There's no breeze in (3,2) so there's no pit  in (3,3).
Orion somewhat wants to go to (3,3).
With neither wumpus nor pit, (3,3) is safe.
Orion strongly wants to go to (3,3).
Seeing glitter, Orion knows there is gold in (3,2).
Orion wants to pick up the gold in (3,2).
PLANNING...
ACTING...
Orion picks up 100 pieces of gold!
SIMULATING...
SENSING...
THINKING...
Orion strongly wants to go to (2,2).
Orion moderately wants to go to (2,3).
Orion moderately wants to go to (1,3).
Orion moderately wants to go to (2,1).
Orion strongly wants to go to (3,3).
Orion strongly wants to go to (4,2).
Orion strongly wants to go to (3,1).
PLANNING...
ACTING...
Orion goes to (2,2).
SIMULATING...
```

Cave3:(Score=273)



```
SENSING...
Seeker sees no glitter.
Seeker smells a stench.
Seeker feels no breeze in (6,1).
THINKING...
The wumpus is in (6,2).
Seeker somewhat wants to go to (5,3).
Seeing no glitter, Seeker knows there is no gold in (6,1).
PLANNING...
ACTING...
Seeker goes to (6,2).
SIMULATING...
Aaarrrggghhhhhh....munch...munch...munch 😕
261
```

IV. Task 2: New Top-Level Goal – Killing the Wumpus

Objective:

In this task, we aim to enhance the game by introducing a new top-level goal for the Hunter. This goal is to eliminate the Wumpus by adding a "shoot" action, while also retaining the original goal of finding gold. The Hunter can now choose to find the gold and kill the Wumpus in any order before leaving the caves.

Implementation:

To enable the agent to use the "shoot" action against the Wumpus, we introduced a slot that keeps track of the agent's arrow count.

```
;; Start Change Task 2

(deftemplate hunter "A hunter"
  (slot agent (default "Xena"))
  (slot x (type INTEGER))
  (slot y (type INTEGER))
  (slot gold (default 0)(type INTEGER))
  (slot alive (default TRUE))
  (slot arrows (default 1)(type INTEGER)))

;; End Change Task 2
```

Setting desires:

- **Desire to leave caves:**

BEFORE:

```
(defrule desire-to-leave-caves
  (task think)
  (hunter (agent ?a)(x ?x)(y ?y)(gold ~0))
  (cave (x ?x)(y ?y)(has-exit TRUE))
  =>
  (printout t "Having found the gold, " ?a " want to leave the caves." crlf)
  (assert (desire (agent ?a)(strength ?*veryhigh*)(action leavecaves))))
```

AFTER: We added (arrows 0), to higher the desire once gold is found and the wumpus is killed.

```
;; Start Change Task 2

(defrule desire-to-leave-caves
  (task think)
  (hunter (agent ?a)(x ?x)(y ?y)(gold ~0)(arrows 0))
  (cave (x ?x)(y ?y)(has-exit TRUE))
  =>
  (printout t "Having found the gold, " ?a " want to leave the caves." crlf)
  (assert (desire (agent ?a)(strength ?*veryhigh*)(action leavecaves))))

;; End Change Task 2
```

❖ Desire shoot wumpus:

We introduced a new desire, aiming to eliminate the Wumpus, to enable the agent to move forward and execute the Wumpus elimination. Therefore, we have crafted the code below:

```
;; Start Change Task 2

(defrule desire-shoot-wumpus
  (task think)
  (hunter (agent ?agent)(x ?x1)(y ?y1))
  (cave (x ?x2)(y ?y2)(has-wumpus TRUE))
  (test (or (eq ?x1 ?x2) (eq ?y1 ?y2)))
  (not (desire (agent ?agent) (strength ?*veryhigh*) (action shootwumpus) (x ?x2) (y ?y2)))
  =>
  (assert (desire (agent ?agent) (strength ?*veryhigh*) (action shootwumpus) (x ?x2) (y ?y2)))
  (printout t ?agent " strongly wants to shoot the wumpus at (" ?x2 "," ?y2 ")." crlf)
)

;; End Change Task 2
```

Let's break down what this code does:

- **(task think)**: The rule is executed when the task is set to "think," which implies that the hunter is actively contemplating its actions.
- **(hunter (agent ?agent)(x ?x1)(y ?y1))**: This part of the rule matches a pattern where the hunter (agent) is located at certain coordinates (?x1, ?y1).
- **(cave (x ?x2)(y ?y2)(has-wumpus TRUE))**: This part of the rule matches a cave that has a Wumpus present at coordinates (?x2, ?y2).
- **(test (or (eq ?x1 ?x2) (eq ?y1 ?y2))):** This condition checks for direct adjacency by comparing the x and y coordinates of the hunter and the wumpus. This ensures that the desire to shoot the wumpus is only created when they are directly adjacent, not diagonal.
- **(not (desire (agent ?agent) (strength ?*veryhigh*) (action shootwumpus) (x ?x2) (y ?y2)))**: The rule checks that there is no existing desire (goal) for the agent to shoot the Wumpus at the same location with very high strength. If such a desire does not already exist, this part of the rule matches.

=>: This symbol indicates that if all the conditions above are met, the following actions should be executed.

- **(assert (desire (agent ?agent) (strength ?*veryhigh*) (action shootwumpus) (x ?x2) (y ?y2)))**: This action creates a new desire (goal) for the agent to shoot the Wumpus at the specified location (?x2, ?y2) with a very high strength. This desire is added to the agent's goals.
- **(printout t ?agent " strongly wants to shoot the wumpus at (" ?x2 "," ?y2 ")." crlf)**: This action generates a message indicating that the agent strongly desires to shoot the Wumpus at the specified location. The message is printed to the standard output.

In summary, this rule captures the agent's desire to shoot the Wumpus when it is located in a cave with a Wumpus, provided that no existing strong desire for the same action at the same location exists. If the conditions are met, a new strong desire is asserted, and a corresponding message is

printed. This rule helps the agent make decisions based on its goals and the presence of the Wumpus in the game world.

Act Rules:

- **Found exit:**

BEFORE:

```
(defrule found-exit
  "If the hunter has gold and finds an exit, she leaves."
  (task act)
  (hunter (agent ?agent) (x ?x)(y ?y)(gold ~0))
  (exit (x ?x)(y ?y))
  =>
 (printout t ?agent " leaves the caves." crlf)
 (halt))
```

AFTER: We have updated and set the arrows to 0 after taking the gold and finding an exit.

```
;; Start Change Task 2

(defrule found-exit
  "If the hunter has gold and no arrows and finds an exit, it leaves."
  (task act)
  (hunter (agent ?agent) (x ?x)(y ?y)(gold ~0)(arrows 0))
  (exit (x ?x)(y ?y))
  =>
 (printout t ?agent " leaves the caves." crlf)
 (halt))

;; End Change Task 2
```

- **Shoot Wumpus:**

```
;; Start Change Task 2

(defrule shoot-wumpus
  (task act)
  ?goal <- (goal (action shootwumpus))
  ?f1 <- (hunter (agent ?a)(x ?x)(y ?y)(arrows ?current))
  ?cave <- (cave (x ?x2)(y ?y2)(has-wumpus TRUE))
  ?world<-(wumpus (x ?x3) (y ?y3) (alive TRUE))
  =>
  (printout t "Found wumpus at (" ?x3 "," ?y3 ")" crlf)
  (printout t ?a " shoots " ?current " arrow at the wumpus in (" ?x2 "," ?y2 ")and kills it!" crlf)
  (modify ?f1 (arrows (- 1 ?current)))
  (modify ?cave (has-wumpus FALSE)(safe TRUE))
  (modify ?world (alive FALSE))
  (retract ?goal)
)

;; End Change Task 2
```

The above piece of code is designed to handle the action of the agent shooting the Wumpus under certain conditions.

- **(task act)**: The rule is executed when the task is set to "act," which means that the hunter is actively taking an action.
- **?goal <- (goal (action shootwumpus))**: This part of the rule matches a goal where the action is specified as "shootwumpus" and captures it as the variable ?goal.
- **?f1 <- (hunter (agent ?a)(x ?x)(y ?y)(arrows ?current))**: This part of the rule matches the hunter's current state, including its agent name, location (x and y coordinates), and the number of arrows it has, capturing these values in variables.
- **?cave <- (cave (x ?x2)(y ?y2)(has-wumpus TRUE))**: This part of the rule matches a cave where a Wumpus is present (has-wumpus set to TRUE) and captures the cave's location in variables ?x2 and ?y2.
- **?world<-(wumpus (x ?x3) (y ?y3) (alive TRUE))**: This part of the rule matches a Wumpus in the game world that is still alive (alive set to TRUE) and captures its location in variables ?x3 and ?y3.

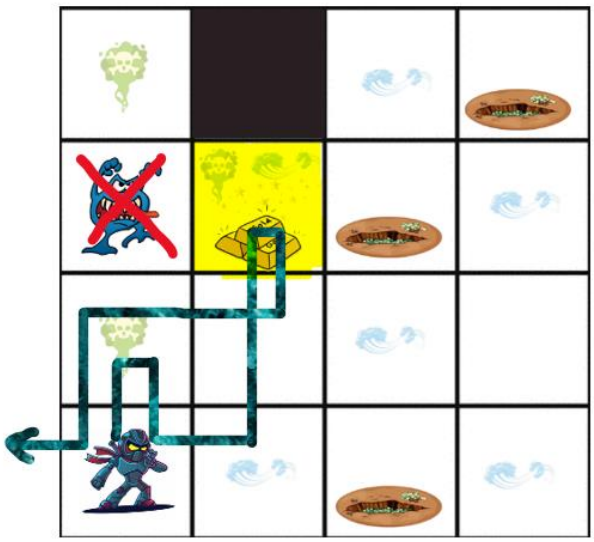
=>: This symbol indicates that if all the conditions above are met, the following actions should be executed.

- **(printout t "Found wumpus at (" ?x3 "," ?y3 ")" crlf)**: This action prints a message indicating that the Wumpus has been found at the specified location, which is the location of the living Wumpus.
- **(printout t ?a " shoots " ?current " arrow at the wumpus in (" ?x2 "," ?y2 ")and kills it!" crlf)**: This action generates a message indicating that the agent (the hunter) is shooting an arrow at the Wumpus' location and successfully kills it. The message includes the agent's name, the number of arrows used, and the location of the Wumpus.
- **(modify ?f1 (arrows (- 1 ?current)))**: This action updates the number of arrows the hunter has by subtracting one arrow after shooting.
- **(modify ?cave (has-wumpus FALSE)(safe TRUE))**: This action modifies the cave's state, setting "has-wumpus" to FALSE to indicate that the Wumpus is no longer present in that cave, and "safe" to TRUE, meaning it is now safe.
- **(modify ?world (alive FALSE))**: This action changes the state of the Wumpus in the game world to indicate that it is no longer alive by setting "alive" to FALSE.
- (retract ?goal): Finally, this action retracts the goal related to shooting the Wumpus, as it has been successfully executed.

Indeed, this rule models the shooting action of the hunter when the goal is set to "shootwumpus," the hunter has arrows, there is a living Wumpus in a specific cave, and the task is to "act." When these conditions are met, the rule executes actions to handle the shooting of the Wumpus and updates the game's state accordingly.

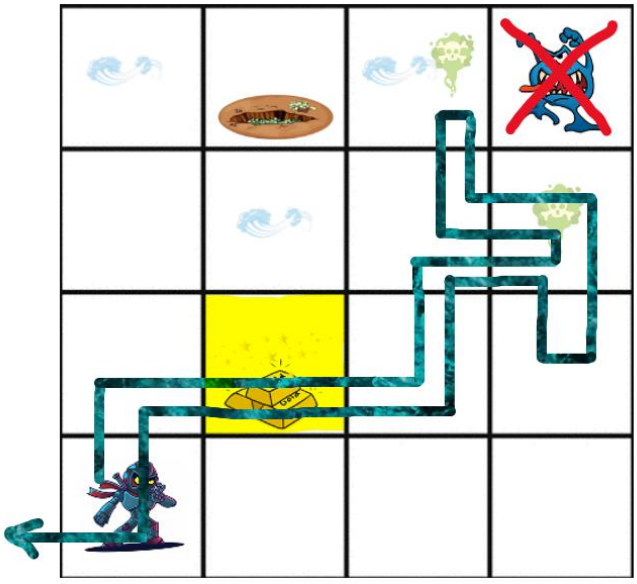
Showing it works:

Cave0: (Score=178)



```
ACTING...
Found wumpus at (1,3)
Xena shoots 1 arrow at the wumpus in (1,3)and kills it!
SIMULATING...
SENSING...
THINKING...
Xena moderately wants to go to (3,2).
Xena strongly wants to go to (1,3).
Xena wants to pick up the gold in (2,3).
PLANNING...
ACTING...
Xena picks up 10 pieces of gold!
```

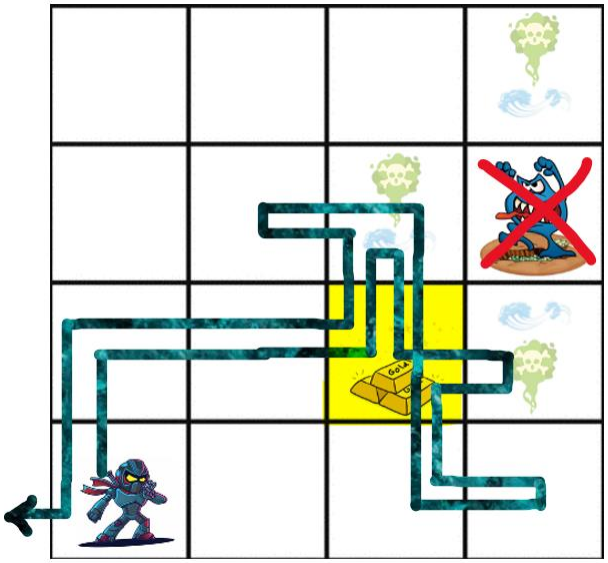
Cave1: (Score=393)



```
Actaeon goes to (2,2).
SIMULATING...
SENSING...
Actaeon sees glitter.
Actaeon notices (2,3) nearby.
Actaeon notices (3,2) nearby.
Actaeon smells nothing.
Actaeon feels no breeze in (2,2).
THINKING...
Actaeon moderately wants to go to (1,3).
No stench in (2,2) means no wumpus in (2,3).
There's no breeze in (2,2) so there's no pit  in (2,3).
Actaeon somewhat wants to go to (2,3).
With neither wumpus nor pit, (2,3) is safe.
Actaeon strongly wants to go to (2,3).
There's no breeze in (2,2) so there's no pit  in (3,2).
No stench in (2,2) means no wumpus in (3,2).
Actaeon somewhat wants to go to (3,2).
With neither wumpus nor pit, (3,2) is safe.
Actaeon strongly wants to go to (3,2).
Seeing glitter, Actaeon knows there is gold in (2,2).
Actaeon wants to pick up the gold in (2,2).
Actaeon strongly wants to go to (2,1).
PLANNING...
ACTING...
Actaeon picks up 100 pieces of gold!
```

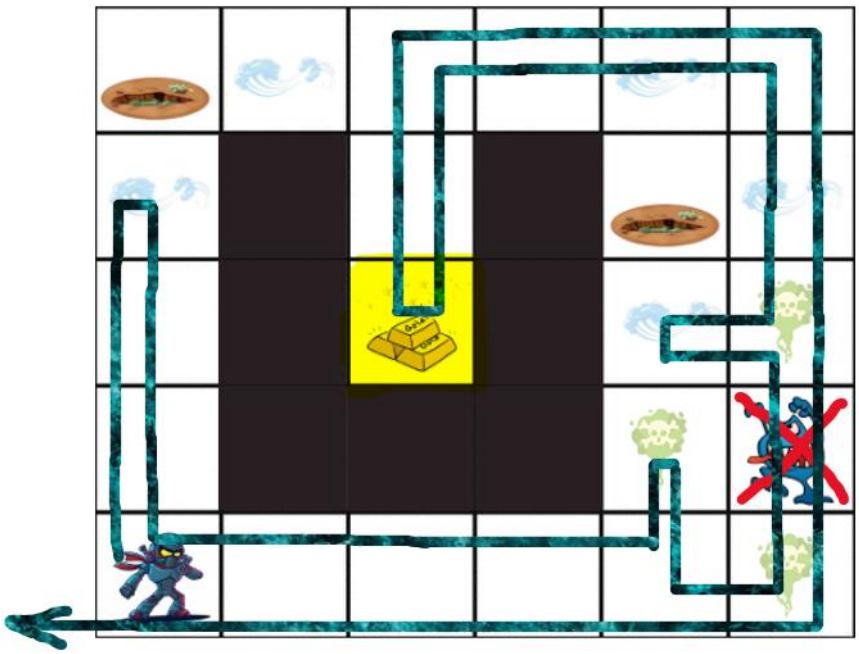
```
The wumpus is in (4,4).
Actaeon strongly wants to shoot the wumpus at (4,4).
There's no breeze in (4,2) so there's no pit  in (4,1).
No stench in (4,2) means no wumpus in (4,1).
Actaeon somewhat wants to go to (4,1).
With neither wumpus nor pit, (4,1) is safe.
Actaeon strongly wants to go to (4,1).
Seeing no glitter, Actaeon knows there is no gold in (4,2).
PLANNING...
ACTING...
Found wumpus at (4,4)
Actaeon shoots 1 arrow at the wumpus in (4,4)and kills it!
```

Cave2: (Score=370)



```
No stench in (2,3) means no wumpus in (2,4).
Orion somewhat wants to go to (2,4).
With neither wumpus nor pit, (2,4) is safe.
Orion strongly wants to go to (2,4).
Seeing no glitter, Orion knows there is no gold in (2,3).
Orion strongly wants to go to (1,3).
PLANNING...
ACTING...
Found wumpus at (4,3)
Orion shoots 1 arrow at the wumpus in (4,3)and kills it!
```

Cave3: (Score=610)



```
SIMULATING...
SENSING...
Seeker sees no glitter.
Seeker smells a stench.
Seeker feels no breeze in (6,1).
THINKING...
The wumpus is in (6,2).
Seeker strongly wants to shoot the wumpus at (6,2).
Seeker somewhat wants to go to (5,3).
Seeing no glitter, Seeker knows there is no gold in (6,1).
PLANNING...
ACTING...
Found wumpus at (6,2)
Seeker shoots 1 arrow at the wumpus in (6,2)and kills it!
SIMULATING...
SENSING...
THINKING...
Seeker somewhat wants to go to (5,3).
Seeker strongly wants to go to (6,2).
PLANNING...
ACTING...
Seeker goes to (6,2).
```

```
SENSING...
Seeker sees glitter.
Seeker smells nothing.
Seeker feels no breeze in (3,3).
THINKING...
Seeker moderately wants to go to (2,5).
Seeing glitter, Seeker knows there is gold in (3,3).
Seeker wants to pick up the gold in (3,3).
PLANNING...
ACTING...
Seeker picks up 100 pieces of gold!
```

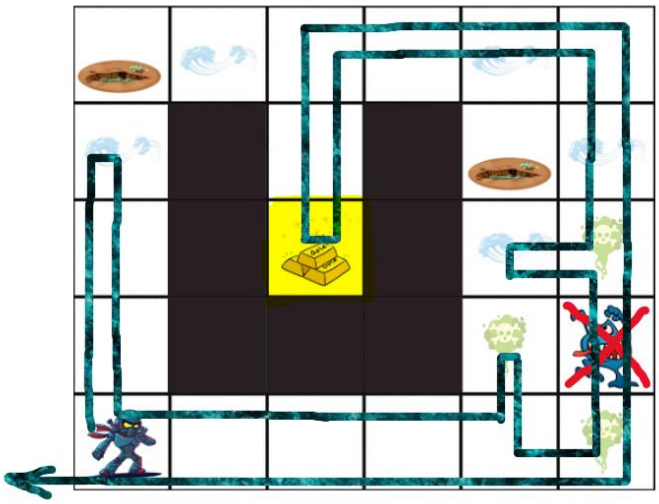
V. Task 3: Improve the Hunter's Ability to Go to a Given Location

Objective:

The objective of Task 3 is to enhance the hunter's ability to navigate to a distant cave successfully in "cave3.jess." The aim is to address the issue where the hunter gets stuck in a cave and is unable to determine how to proceed. However, and according to the results of Task 2, it appears that our agent does not get stuck…

What we did:

- **Scenario Analysis**: First, we need to analyze the behavior of the hunter in the world defined by "cave3.jess."



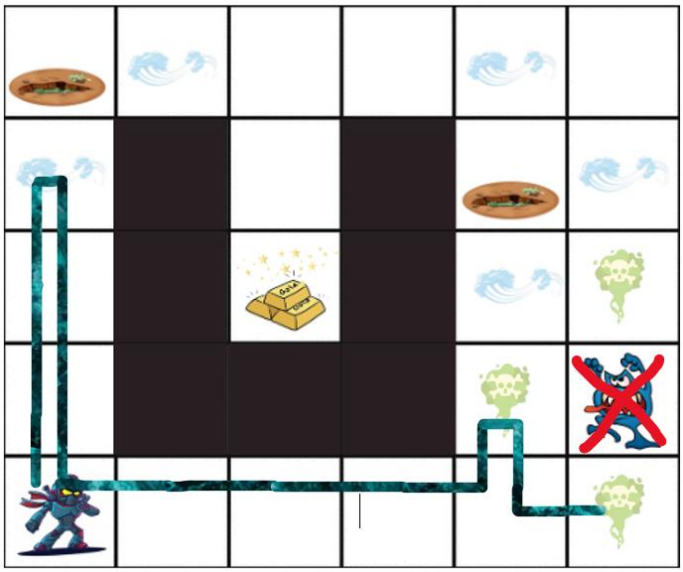
After knowing that there is a stench in (5,2) and in (6,1), the agent concludes that there is a wumpus in (6,2). Armed with an arrow, the agent shoots the wumpus and kills it… Since the game environment has only one wumpus, we can happily mark it as wumpus free and ignore any stench we may encounter. Indeed, this is what we have done in task 2.

```
(defrule shoot-wumpus
  (task act)
  ?goal <- (goal (action shootwumpus))
  ?f1 <- (hunter (agent ?a)(x ?x)(y ?y)(arrows ?current))
  ?cave <- (cave (x ?x2)(y ?y2)(has-wumpus TRUE))
  ?world<-(wumpus (x ?x3) (y ?y3) (alive TRUE))
  =>
  (printout t "Found wumpus at (" ?x3 "," ?y3 ")" crlf)
  (printout t ?a " shoots " ?current " arrow at the wumpus in (" ?x2 "," ?y2 ")and kills it!" crlf)
  (modify ?f1 (arrows (- 1 ?current)))
  (modify ?cave (has-wumpus FALSE)(safe TRUE))
  (modify ?world (alive FALSE))
  (retract ?goal)
)
```

Now, let's assume that we did not brainstorm well in task 2, and we underestimated the role of marking (6,2) as **(has-wumpus FALSE)**. The code becomes:

```
(defrule shoot-wumpus
  (task act)
  ?goal <- (goal (action shootwumpus))
  ?f1 <- (hunter (agent ?a)(x ?x)(y ?y)(arrows ?current))
  ?cave <- (cave (x ?x2)(y ?y2)(has-wumpus TRUE))
  ?world<-(wumpus (x ?x3) (y ?y3) (alive TRUE))
  =>
  (printout t "Found wumpus at (" ?x3 "," ?y3 ")" crlf)
  (printout t ?a " shoots " ?current " arrow at the wumpus in (" ?x2 "," ?y2 ")and kills it!" crlf)
  (modify ?f1 (arrows (- 1 ?current)))
  (modify ?cave (safe TRUE))
  (modify ?world (alive FALSE))
  (retract ?goal)
)
```

Running this modified code on cave3.jess results in the following:



```
THINKING...
he wumpus is in (6,2).
eeker strongly wants to shoot the wumpus at (6,2).
eeker somewhat wants to go to (5,3).
eeing no glitter, Seeker knows there is no gold in (6,1).
LANNING...
CTING...
ound wumpus at (6,2)
eeker shoots 1 arrow at the wumpus in (6,2)and kills it!
IMULATING...
ENSING...
HINKING...
eeker strongly wants to shoot the wumpus at (6,2).
eeker somewhat wants to go to (5,3).
eeker strongly wants to go to (6,2).
(6,2) is safe, so there's no pit or wumpus in it.
eeker strongly wants to go to (6,2).
LANNING...
CTING...
ARNING: unsatisfied goal: shootwumpus 6 2.
74
ess>
```

- **Identifying the Issue**: We identified that the problem was primarily related to the absence of (has-wumpus FALSE) to handle this specific case. The agent lacked the ability to set a goal to go to a distant cave and take the appropriate actions to reach that goal.
- **Concluding Task 2's Role**: We finally noted that Task 2, which involved shooting the Wumpus, played a role in addressing this issue. In scenarios where "has-wumpus FALSE" was not implemented, there was a potential issue with the hunter failing to go to distant caves. However, this problem did not persist since we had successfully implemented Task 2. The synergy between the tasks contributed to a more capable agent overall.

VI. Task 4: Extra Credit: Allow the Hunter to Choose the Best Action

Objective:

In Task 4, we introduce a feature that enhances the Hunter's decision-making ability. This feature enables the Hunter to evaluate and select the best action to take when multiple actions are possible in a given situation. We achieve this by prioritizing the available actions. For example, the Hunter may have options like picking up gold or shooting the wumpus.

Implementation-1 (Killing the Wumpus before picking up Gold):

A crucial step was to introduce a new global variable, ?*superhigh*, to encode a strength level of 10. The global variables that encode the strength of desires were updated as follows:

```
(defglobal ; these global variables encode the strength of desires
  ;; Change Task 4
  ?*superhigh* = 10
  ?*veryhigh* = 5
  ?*high* = 4
  ?*medium* =  3
  ?*low* = 2
  ?*verylow* = 1)
```

Furthermore, the desire-shoot-wumpus rule was updated to incorporate the new strength level. The changes are summarized as follows:

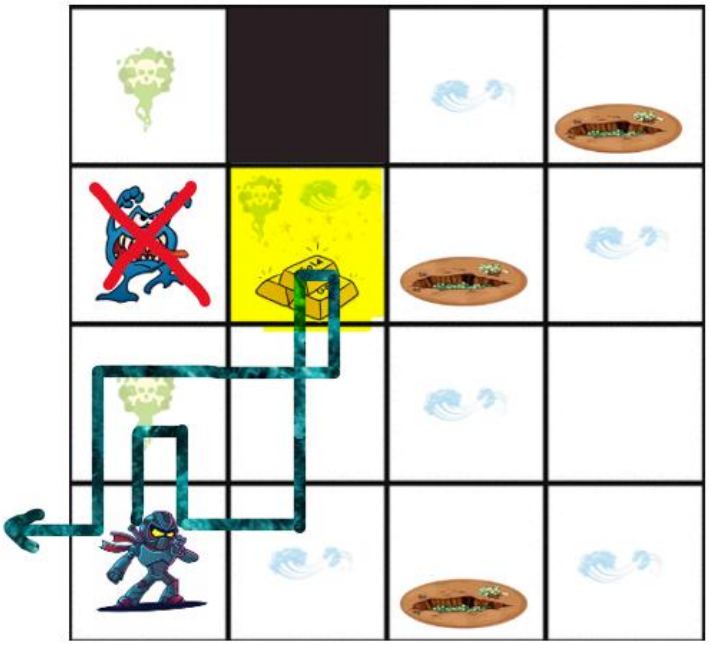
```
;; Start Change Task 2 + Task 4

(defrule desire-shoot-wumpus
  (task think)
  (hunter (agent ?agent)(x ?x1)(y ?y1))
  (cave (x ?x2)(y ?y2)(has-wumpus TRUE))
  (test (or (eq ?x1 ?x2) (eq ?y1 ?y2)))
  (not (desire (agent ?agent) (strength ?*superhigh*) (action shootwumpus) (x ?x2) (y ?y2)))
  =>
  (assert (desire (agent ?agent) (strength ?*superhigh*) (action shootwumpus) (x ?x2) (y ?y2)))
  (printout t ?agent " strongly wants to shoot the wumpus at (" ?x2 "," ?y2 ")." crlf)
)

;; End Change Task 2 + Task 4
```

- The strength level in this rule was originally set to ?*veryhigh* for the Hunter's strong desire to shoot the Wumpus. It was changed to ?*superhigh* to emphasize the Hunter's priority in shooting the Wumpus when it is within close proximity.

These changes align with the task's requirement to allow the Hunter to select the best action among available choices. By increasing the desire strength to ?*superhigh* when shooting the Wumpus is a viable option, the Hunter now strongly prioritizes this action when the opportunity arises.

Showing it works:

```
Found wumpus at (1,3)
Xena shoots 1 arrow at the wumpus in (1,3)and kills it!
SIMULATING...
SENSING...
THINKING...
Xena moderately wants to go to (3,2).
Xena strongly wants to go to (1,3).
Xena wants to pick up the gold in (2,3).
PLANNING...
ACTING...
Xena picks up 10 pieces of gold!
SIMULATING...
SENSING...
THINKING...
Xena strongly wants to go to (2,2).
Xena moderately wants to go to (3,2).
Xena strongly wants to go to (1,3).
PLANNING...
ACTING...
Xena goes to (2,2).
SIMULATING...
SENSING...
THINKING...
Xena moderately wants to go to (1,3).
Xena strongly wants to go to (2,1).
Xena strongly wants to go to (3,2).
PLANNING...
ACTING...
Xena goes to (2,1).
SIMULATING...
SENSING...
THINKING...
Xena moderately wants to go to (1,3).
Xena moderately wants to go to (3,2).
Xena strongly wants to go to (1,1).
PLANNING...
ACTING...
Xena goes to (1,1).
Xena leaves the caves.
180
```



Implementation-2 (Picking up Gold before killing the Wumpus):

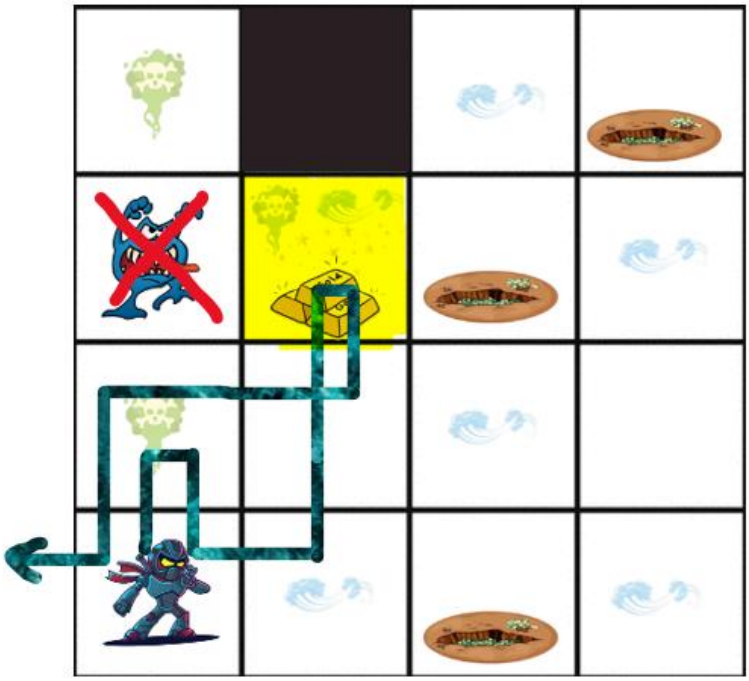
To prioritize picking up gold, lust-for-gold rule was updated to incorporate the new strength level. The changes are summarized as follows:

```
(defrule lust-for-gold
  (task think)
  (hunter (agent ?a)(x ?x)(y ?y))
  (cave (x ?x)(y ?y)(has-gold TRUE))
  =>
  (printout t ?a " wants to pick up the gold in (" ?x "," ?y ")." crlf)
  (assert (desire (agent ?a)(strength ?*superhigh*)(action pickupgold))))
```

- The strength level in this rule was originally set to ?*veryhigh* for the Hunter's strong desire to pick up gold. It was changed to ?*superhigh* to emphasize the Hunter's priority in taking gold when it is within close proximity.

Showing it works:

```
Xena strongly wants to shoot the wumpus at (1,3).
Xena moderately wants to go to (3,2).
With stench in (2,3), maybe the wumpus is in (3,3).
A breeze in (2,3), so there may be a pit in (3,3).
The pit is in (3,3).
Seeing glitter, Xena knows there is gold in (2,3).
Xena wants to pick up the gold in (2,3).
PLANNING...
ACTING...
Xena picks up 10 pieces of gold!
SIMULATING...
SENSING...
THINKING...
Xena moderately wants to go to (3,2).
PLANNING...
ACTING...
Found wumpus at (1,3)
Xena shoots 1 arrow at the wumpus in (1,3)and kills it!
```



VII. Task 5: Extra Credit: Evaluate the Precision of the Agent

Objective:

The objective of this task is to evaluate the precision of our AI agent in the Wumpus World game by simulating and testing its performance in 50 random scenarios. Specifically, we aim to assess the agent's ability to correctly detect the presence of the Wumpus and pick up the gold in various game configurations. The primary goal is to analyze the agent's decision-making accuracy and report the results in a **confusion matrix**.

Implementation:

For the implementation of this task, we will follow the steps outlined below:

1) **Scenario Generation**:
   Generate 50 random scenarios for the Wumpus World game. These scenarios will include different configurations of caves, gold placement, and the Wumpus's presence. Each scenario will represent a unique game setup (to ensure that, we wrote a python code😉).

```python
import random

num_caves = 50

for i in range(num_caves):
    # Randomly choose if x and y dimensions should be equal or not
    equal_dimensions = random.choice([True, False])

    if equal_dimensions:
        # If equal dimensions, choose a random size for the cave
        world_size = (random.randint(4, 5), random.randint(4, 5))
    else:
        # If unequal dimensions, choose random values for x and y dimensions
        world_size = (random.randint(4, 5), random.randint(4, 5))

    gold_x = random.randint(1, world_size[0])
    gold_y = random.randint(1, world_size[1])

    # Generate multiple pits
    num_pits = random.randint(1, 3)  # Allow up to 3 pits in a cave
    pits_condition = ''
    for _ in range(num_pits):
        pit_x = random.randint(1, world_size[0])
        pit_y = random.randint(1, world_size[1])
        pits_condition += f'(pit (x {pit_x})(y {pit_y}))\n'

    # Generate multiple nocaves
    num_nocaves = random.randint(1, 3)  # Allow up to 3 nocaves in a cave
    nocaves_condition = ''
    for _ in range(num_nocaves):
        nocave_x = random.randint(1, world_size[0])
        nocave_y = random.randint(1, world_size[1])
        nocaves_condition += f'(nocave (x {nocave_x})(y {nocave_y}))\n'

    wumpus_x = random.randint(1, world_size[0])
    wumpus_y = random.randint(1, world_size[1])
    exit_x = 1
    exit_y = 1
    agent_name = "C" + str(i + 1)

    with open(f'cave{i + 1}.jess', 'w') as f:
        f.write(f'(worldsize {world_size[0]} {world_size[1]})\n')
        f.write(f'(gold (x {gold_x})(y {gold_y})(amount 100))\n')
        f.write(pits_condition)
        f.write(nocaves_condition)
        f.write(f'(wumpus (x {wumpus_x})(y {wumpus_y}))\n')
        f.write(f'(exit (x {exit_x})(y {exit_y}))\n')
        f.write(f'(hunter (agent {agent_name})))\n')
```

2) **Data Collection:**

During each scenario, collect data on the agent's performance:

- Record whether the agent successfully killed the Wumpus and picked up the gold.

3) **Confusion Matrix:**

Utilize the collected data to construct a confusion matrix for each scenario. The confusion matrix will consist of the following components for both Wumpus detection and gold collection:

- **True Positives (TP)**: These are situations where the agent makes a positive decision, and it is the correct decision (we expect the agent to succeed, and it does).
- **True Negatives (TN)**: In this case, the agent makes a negative decision, and that decision is correct (we expect the agent to fail, and it does).
- **False Positives (FP)**: These occur when the agent makes a positive decision, but it is incorrect (we expect the agent to fail, but it succeeds).
- **False Negatives (FN)**: These arise when the agent makes a negative decision when it should have made a positive one (we expect the agent to succeed, but it fails).

4) **Precision Evaluation:**

Calculate the precision of the agent. Precision measures the accuracy of the agent's positive decisions (e.g., correctly detecting the Wumpus or picking up the gold).

$$\text{Precision} = \frac{TP}{TP+FP} \qquad Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

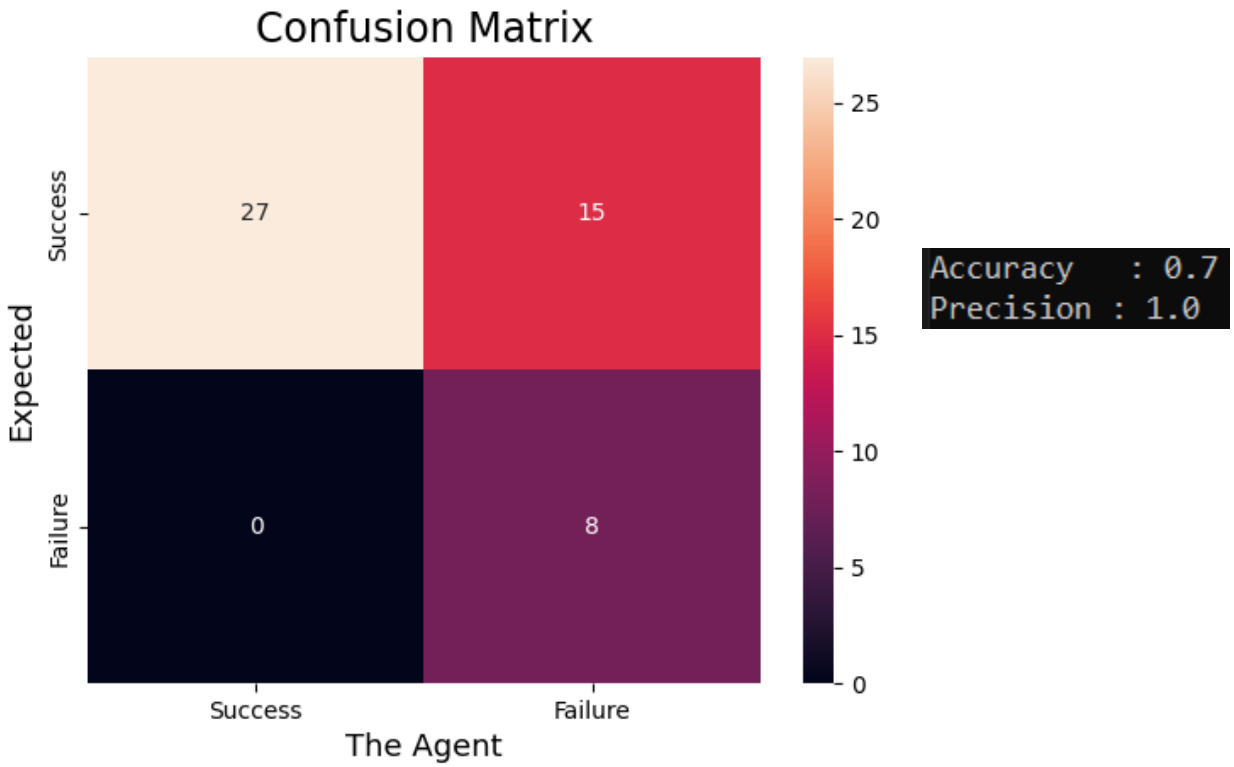
By implementing this approach, we aim to assess the AI agent's precision in the Wumpus World game across multiple scenarios and provide valuable insights into its performance and decision-making capabilities.

Python code:

```
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score

#Create the NumPy array for actual and expected labels.
expected    = np.array(
  ['Success','Success','Success','Success','Success','Success','Success','Success','Failure','Failure','Failure','Failure','Failure','Failure',
actual = np.array(
  ['Success','Success','Success','Success','Success','Success','Success','Success','Failure','Failure','Failure','Failure','Failure','Failure',

#compute the confusion matrix.
cm = confusion_matrix(actual,expected)

#Plot the confusion matrix.
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Success','Failure'],
            yticklabels=['Success','Failure'])
plt.ylabel('Expected',fontsize=13)
plt.xlabel('The Agent',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()


accuracy = accuracy_score(actual, expected)
print("Accuracy  :", accuracy)
precision = precision_score(actual, expected, pos_label='Failure')
print("Precision :", precision)
```

The attached code helps assess the agent's performance by visualizing its decisions in a **confusion matrix** and providing quantitative metrics such as accuracy and precision. It is a valuable tool for evaluating how well the agent is doing in terms of correctly identifying and acting upon specific situations, such as picking up gold and successfully shooting the Wumpus.

The Confusion Matrix: *(Please, scroll down to find the scenarios we used to test our agent 😊)*



Confusion Matrix Analysis:

- **True Positives (TP)**: 27 - The agent correctly identified the presence of the Wumpus and made the decision to shoot it, also, it correctly identified the presence of the gold and decided of picking it up. A high number of true positives (27) indicates the agent's ability to accurately detect the Wumpus and gold.
- **True Negatives (TN)**: 8 - The agent accurately failed when it was expected to fail.
- **False Positives (FP)**: 0 - Impressively, the agent did not make any false positive decisions, This demonstrates the agent's skill in avoiding false alarms.
- **False Negatives (FN)**: 15 - The agent incorrectly failed in 15 instances. Reducing this number should be a priority to improve the agent's performance.

Performance Metrics:

- **Precision**: 100% - Precision evaluates how accurately the agent makes positive decisions, particularly in the context of picking up gold and shooting the Wumpus. An exceptional precision score of 100% implies that when the agent decides to shoot and pick up the gold, it is almost always correct.
- **Accuracy**: 70% - Accuracy represents the ratio of correctly predicted instances to the total instances. While precision focuses on the accuracy of positive decisions, accuracy provides a broader perspective on overall correctness. An accuracy rate of 70% indicates that the agent performs well but has room for improvement, particularly in reducing false negatives.

In summary, the agent exhibits an outstanding precision score, suggesting its remarkable accuracy in shooting the Wumpus when required. However, to enhance overall performance, there is a need to address false negatives (missed opportunities) that occur when the agent fails to shoot the Wumpus or pick up gold. By reducing false negatives, the agent can further boost its accuracy and effectiveness in navigating the cave.

The summary of the generated caves in the table below 😊:

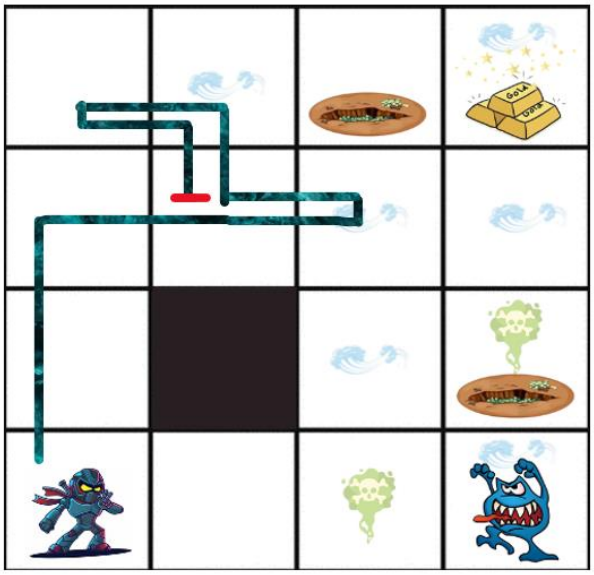
Cave	Expected	Agent Result	Confusion Matrix
1.jess	Success	Failure	FN
2.jess	Success	Success	TP
3.jess	Failure	Failure	TN
4.jess	Success	Success	TP
5.jess	Failure	Failure	TN
6.jess	Success	Failure	FN
7.jess	Failure	Failure	TN
8.jess	Success	Success	TP
9.jess	Success	Failure	FN
10.jess	Success	Success	TP
11.jess	Success	Success	TP
12.jess	Failure	Failure	TN
13.jess	Success	Failure	FN
14.jess	Success	Failure	FN
15.jess	Success	Failure	FN
16.jess	Success	Success	TP
17.jess	Success	Failure	FN
18.jess	Success	Success	TP
19.jess	Failure	Failure	TN
20.jess	Success	Success	TP
21.jess	Success	Failure	FN
22.jess	Success	Success	TP

23.jess	Success	Success	TP
24.jess	Success	Success	TP
25.jess	Success	Failure	FN
26.jess	Success	Success	TP
27.jess	Success	Success	TP
28.jess	Success	Failure	FN
29.jess	Success	Success	TP
30.jess	Success	Failure	FN
31.jess	Success	Failure	FN
32.jess	Success	Success	TP
33.jess	Success	Success	TP
34.jess	Success	Success	TP
35.jess	Success	Success	TP
36.jess	Success	Success	TP
37.jess	Success	Failure	FN
38.jess	Failure	Failure	TN
39.jess	Success	Failure	FN
40.jess	Success	Success	TP
41.jess	Success	Success	TP
42.jess	Failure	Failure	TN
43.jess	Success	Success	TP
44.jess	Failure	Failure	TN
45.jess	Success	Success	TP
46.jess	Success	Success	TP
47.jess	Success	Success	TP
48.jess	Success	Success	TP
49.jess	Success	Success	TP
50.jess	Success	Failure	FN

The generated scenarios:

<u>1.jess:</u> We expect the agent to succeed.

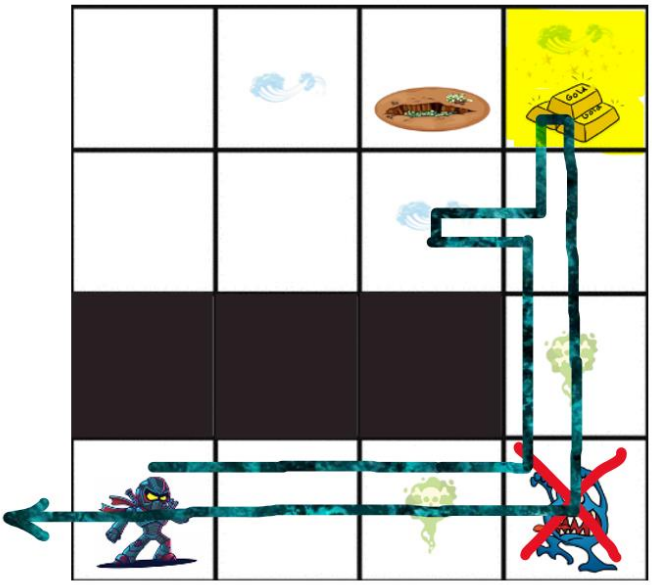
However, the agent fails! (FN)



```
C1 goes to (1,4).
SIMULATING...
SENSING...
C1 sees no glitter.
C1 smells nothing.
C1 feels no breeze in (1,4).
THINKING...
C1 somewhat wants to go to (4,3).
C1 somewhat wants to go to (3,2).
C1 somewhat wants to go to (3,4).
C1 moderately wants to go to (2,1).
The pit is in (3,4).
Seeing no glitter, C1 knows there is no gold in (1,4).
PLANNING...
ACTING...
C1 goes to (2,4) trying to get to (2,1).
C1 goes to (2,3) trying to get to (2,1).
WARNING: unsatisfied goal: go 2 1.
192
```

<u>2.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)



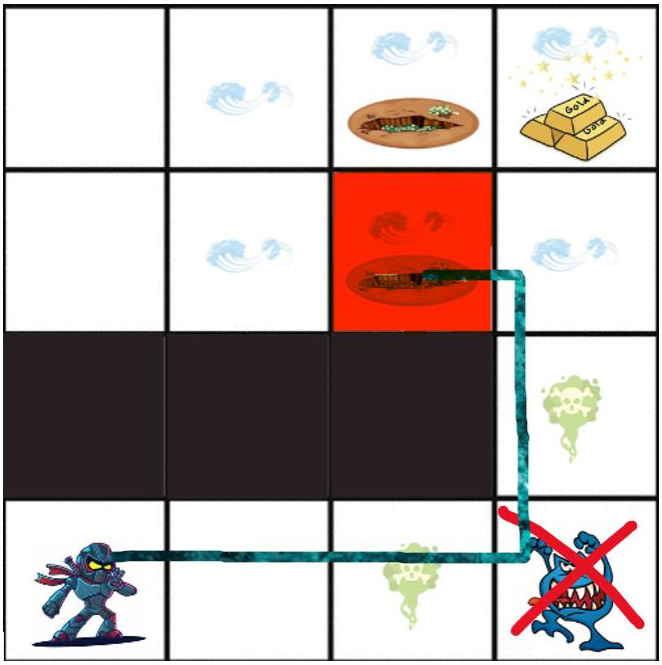
```
Found wumpus at (4,1)
C2 shoots 1 arrow at the wumpus in (4,1)and kills it!
SIMULATING...
SENSING...
THINKING...
C2 strongly wants to go to (4,1).
PLANNING...
ACTING...
C2 goes to (4,1).
```

```
C2 goes to (4,4).
SIMULATING...
SENSING...
C2 sees glitter.
C2 feels a breeze in (4,4).
C2 smells nothing.
THINKING...
C2 somewhat wants to go to (2,3).
The pit is in (3,4).
Seeing glitter, C2 knows there is gold in (4,4).
C2 wants to pick up the gold in (4,4).
PLANNING...
ACTING...
C2 picks up 100 pieces of gold!
```

<u>3.jess:</u> We expect the agent to fail.

Indeed, the agent fails! (TN)

When the agent feels a breeze, it goes to the cave next to it, not above it.

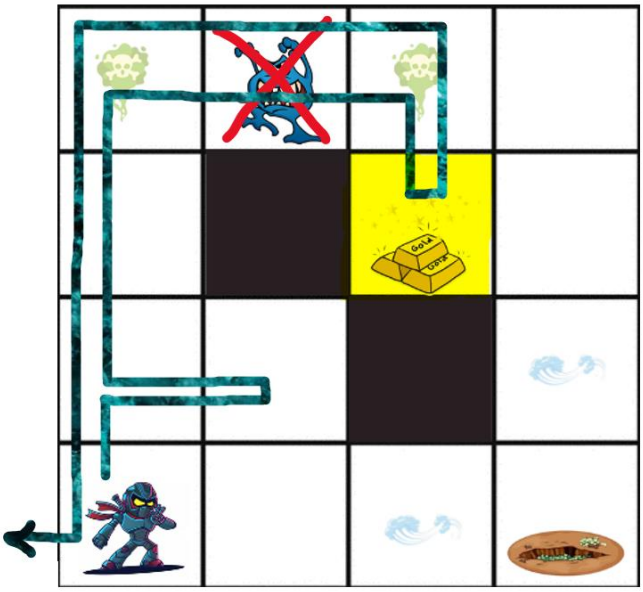


```
No stench in (4,3) means no wumpus in (4,4).
A breeze in (4,3), so there may be a pit in (4,4).
C3 somewhat wants to go to (4,4).
A breeze in (4,3), so there may be a pit in (3,3).
No stench in (4,3) means no wumpus in (3,3).
C3 somewhat wants to go to (3,3).
Seeing no glitter, C3 knows there is no gold in (4,3).
PLANNING...
ACTING...
C3 goes to (3,3).
SIMULATING...
Aaarrrggghhhhhh....plop
146
```

```
C3 goes to (3,1).
SIMULATING...
SENSING...
C3 sees no glitter.
C3 smells a stench.
C3 notices (4,1) nearby.
C3 feels no breeze in (3,1).
THINKING...
With stench in (3,1), maybe the wumpus is in (4,1).
The wumpus is in (4,1).
C3 strongly wants to shoot the wumpus at (4,1).
There's no breeze in (3,1) so there's no pit  in (4,1).
Seeing no glitter, C3 knows there is no gold in (3,1).
PLANNING...
ACTING...
Found wumpus at (4,1)
C3 shoots 1 arrow at the wumpus in (4,1)and kills it!
```

<u>4.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

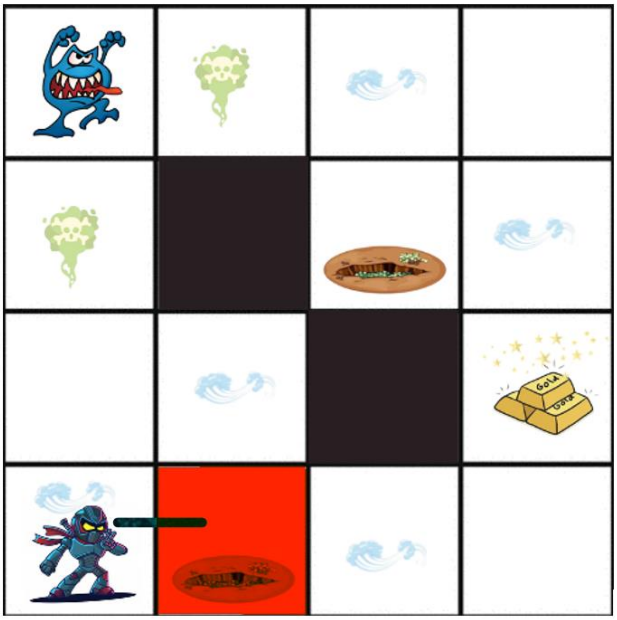


```
ACTING...
Found wumpus at (2,4)
C4 shoots 1 arrow at the wumpus in (2,4)and kills it!
```

```
C4 sees glitter.
C4 notices (4,3) nearby.
C4 smells nothing.
C4 feels no breeze in (3,3).
THINKING...
C4 moderately wants to go to (4,4).
C4 moderately wants to go to (2,1).
No stench in (3,3) means no wumpus in (4,3).
There's no breeze in (3,3) so there's no pit  in (4,3).
C4 somewhat wants to go to (4,3).
With neither wumpus nor pit, (4,3) is safe.
C4 strongly wants to go to (4,3).
Seeing glitter, C4 knows there is gold in (3,3).
C4 wants to pick up the gold in (3,3).
PLANNING...
ACTING...
C4 picks up 100 pieces of gold!
```

<u>5.jess:</u> We expect the agent to fail.

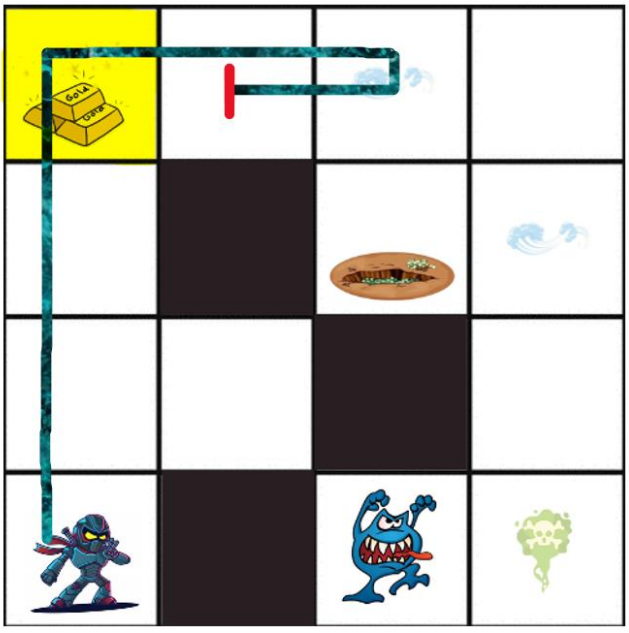
Indeed, the agent fails! (TN)



```
C5 notices (1,2) nearby.
C5 notices (2,1) nearby.
C5 sees no glitter.
C5 feels a breeze in (1,1).
C5 smells nothing.
THINKING...
No stench in (1,1) means no wumpus in (2,1).
No stench in (1,1) means no wumpus in (1,2).
A breeze in (1,1), so there may be a pit in (1,2).
C5 somewhat wants to go to (1,2).
A breeze in (1,1), so there may be a pit in (2,1).
C5 somewhat wants to go to (2,1).
Seeing no glitter, C5 knows there is no gold in (1,1).
Since C5 is in (1,1) and not dead, it must be safe.
(1,1) is safe, so there's no pit or wumpus in it.
PLANNING...
ACTING...
C5 goes to (2,1).
SIMULATING...
Aaarrrggghhhhhh....plop
43
```

<u>6.jess:</u> We expect the agent to succeed.

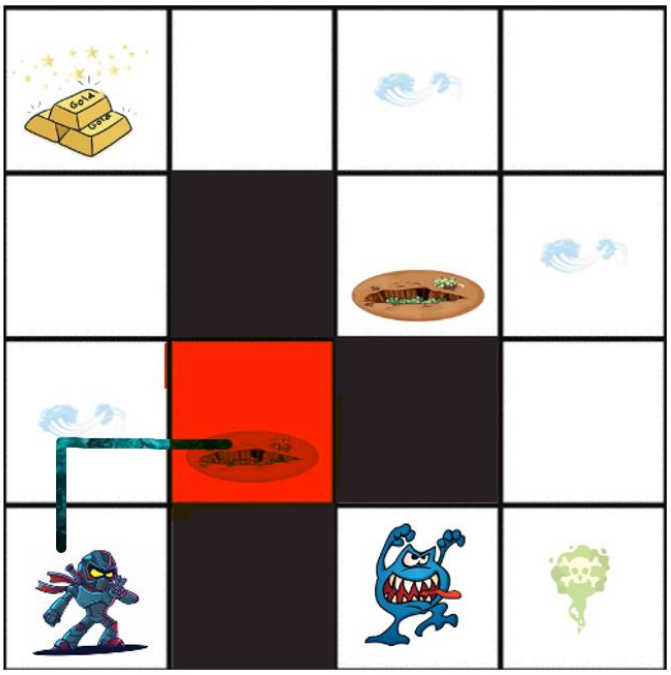
However, the agent fails! (FN)



```
C6 picks up 100 pieces of gold!
SIMULATING...
SENSING...
THINKING...
C6 moderately wants to go to (2,2).
C6 strongly wants to go to (2,4).
PLANNING...
ACTING...
C6 goes to (2,4).
SIMULATING...
SENSING...
C6 sees no glitter.
C6 notices (3,4) nearby.
C6 smells nothing.
C6 feels no breeze in (2,4).
THINKING...
C6 moderately wants to go to (2,2).
There's no breeze in (2,4) so there's no pit  in (3,4).
No stench in (2,4) means no wumpus in (3,4).
C6 somewhat wants to go to (3,4).
With neither wumpus nor pit, (3,4) is safe.
C6 strongly wants to go to (3,4).
Seeing no glitter, C6 knows there is no gold in (2,4).
PLANNING...
ACTING...
C6 goes to (3,4).
SIMULATING...
SENSING...
C6 sees no glitter.
C6 feels a breeze in (3,4).
C6 notices (4,4) nearby.
C6 notices (3,3) nearby.
C6 smells nothing.
THINKING...
C6 moderately wants to go to (2,2).
No stench in (3,4) means no wumpus in (4,4).
A breeze in (3,4), so there may be a pit in (4,4).
C6 somewhat wants to go to (4,4).
A breeze in (3,4), so there may be a pit in (3,3).
No stench in (3,4) means no wumpus in (3,3).
C6 somewhat wants to go to (3,3).
Seeing no glitter, C6 knows there is no gold in (3,4).
PLANNING...
ACTING...
C6 goes to (2,4) trying to get to (2,2).
WARNING: unsatisfied goal: go 2 2.
169
```

<u>7.jess:</u> We expect the agent to fail.

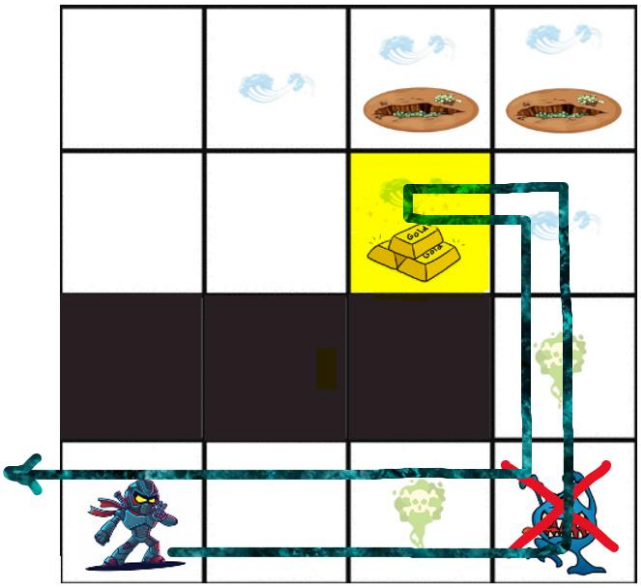
Indeed, the agent fails! (TN)



```
ACTING...
C7 goes to (1,2).
SIMULATING...
SENSING...
C7 sees no glitter.
C7 feels a breeze in (1,2).
C7 notices (1,3) nearby.
C7 notices (2,2) nearby.
C7 smells nothing.
THINKING...
No stench in (1,2) means no wumpus in (2,2).
No stench in (1,2) means no wumpus in (1,3).
A breeze in (1,2), so there may be a pit in (1,3).
C7 somewhat wants to go to (1,3).
A breeze in (1,2), so there may be a pit in (2,2).
C7 somewhat wants to go to (2,2).
Seeing no glitter, C7 knows there is no gold in (1,2).
PLANNING...
ACTING...
C7 goes to (2,2).
SIMULATING...
Aaarrrggghhhhhh....plop
68
```

<u>8.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

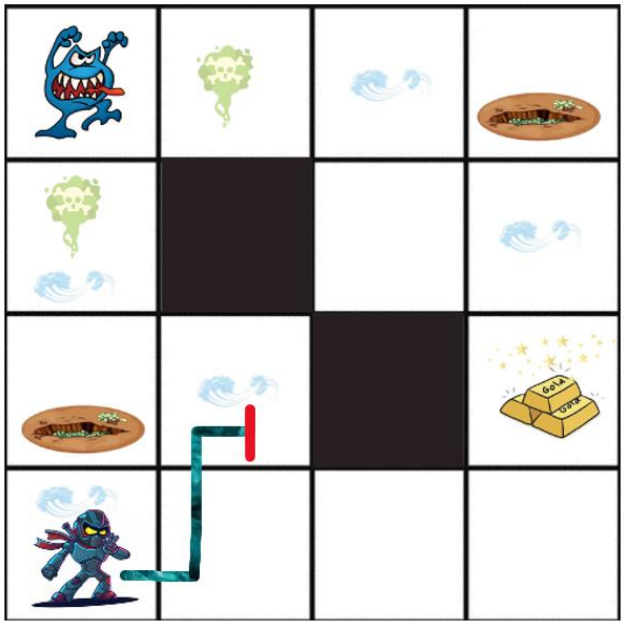


```
With stench in (3,1), maybe the wumpus is in (4,1).
The wumpus is in (4,1).
C8 strongly wants to shoot the wumpus at (4,1).
There's no breeze in (3,1) so there's no pit  in (4,1).
Seeing no glitter, C8 knows there is no gold in (3,1).
PLANNING...
ACTING...
Found wumpus at (4,1)
C8 shoots 1 arrow at the wumpus in (4,1)and kills it!
```

```
C8 goes to (3,3).
SIMULATING...
SENSING...
C8 sees glitter.
C8 feels a breeze in (3,3).
C8 notices (3,4) nearby.
C8 notices (2,3) nearby.
C8 smells nothing.
THINKING...
The pit is in (4,4).
The pit is in (3,3).
No stench in (3,3) means no wumpus in (3,4).
A breeze in (3,3), so there may be a pit in (3,4).
C8 somewhat wants to go to (3,4).
A breeze in (3,3), so there may be a pit in (2,3).
No stench in (3,3) means no wumpus in (2,3).
C8 somewhat wants to go to (2,3).
Seeing glitter, C8 knows there is gold in (3,3).
C8 wants to pick up the gold in (3,3).
PLANNING...
ACTING...
C8 picks up 100 pieces of gold!
```

9.jess: *We expect the agent to succeed.*

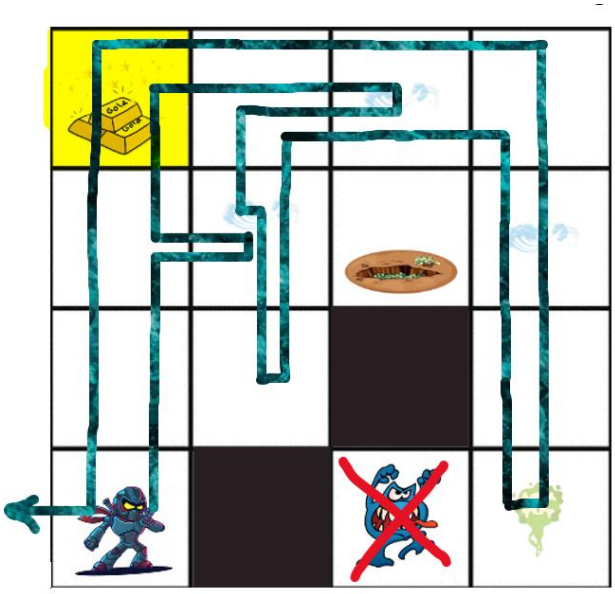
However, the agent fails! (FN)



```
The pit is in (1,2).
The pit is in (2,1).
No stench in (2,1) means no wumpus in (3,1).
No stench in (2,1) means no wumpus in (2,2).
There's no breeze in (2,1) so there's no pit  in (3,1).
C9 somewhat wants to go to (3,1).
With neither wumpus nor pit, (3,1) is safe.
C9 strongly wants to go to (3,1).
There's no breeze in (2,1) so there's no pit  in (2,2).
C9 somewhat wants to go to (2,2).
With neither wumpus nor pit, (2,2) is safe.
C9 strongly wants to go to (2,2).
Seeing no glitter, C9 knows there is no gold in (2,1).
PLANNING...
ACTING...
C9 goes to (2,2).
SIMULATING...
SENSING...
C9 sees no glitter.
C9 feels a breeze in (2,2).
C9 smells nothing.
THINKING...
C9 moderately wants to go to (3,1).
Seeing no glitter, C9 knows there is no gold in (2,2).
PLANNING...
ACTING...
WARNING: unsatisfied goal: go 3 1.
83
```

10.jess: *We expect the agent to succeed.*

Indeed, the agent succeeds! (TP)

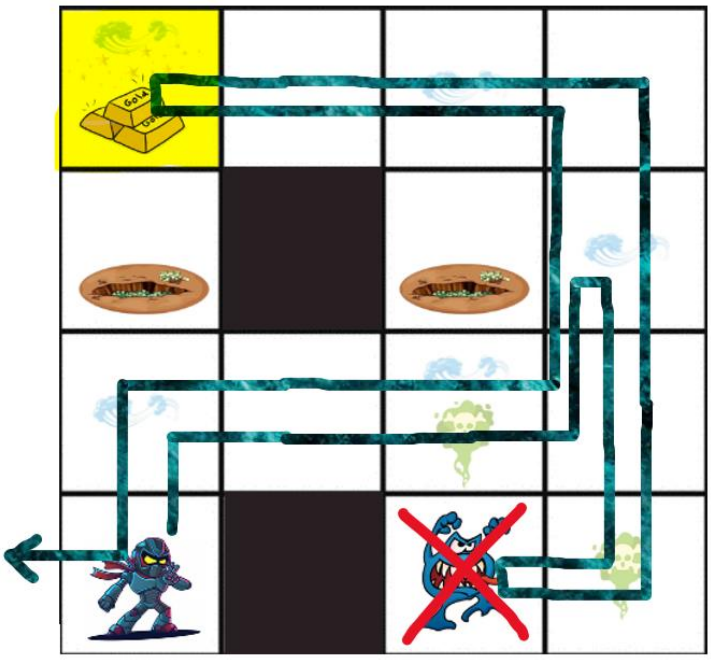


```
C10 sees glitter.
C10 smells nothing.
C10 feels no breeze in (1,4).
THINKING...
C10 somewhat wants to go to (3,3).
C10 moderately wants to go to (2,2).
Seeing glitter, C10 knows there is gold in (1,4).
C10 wants to pick up the gold in (1,4).
There's no breeze in (1,4) so there's no pit  in (2,4).
C10 somewhat wants to go to (2,4).
With neither wumpus nor pit, (2,4) is safe.
C10 strongly wants to go to (2,4).
PLANNING...
ACTING...
C10 picks up 100 pieces of gold!
```

```
ACTING...
Found wumpus at (3,1)
C10 shoots 1 arrow at the wumpus in (3,1)and kills it!
```

<u>11.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

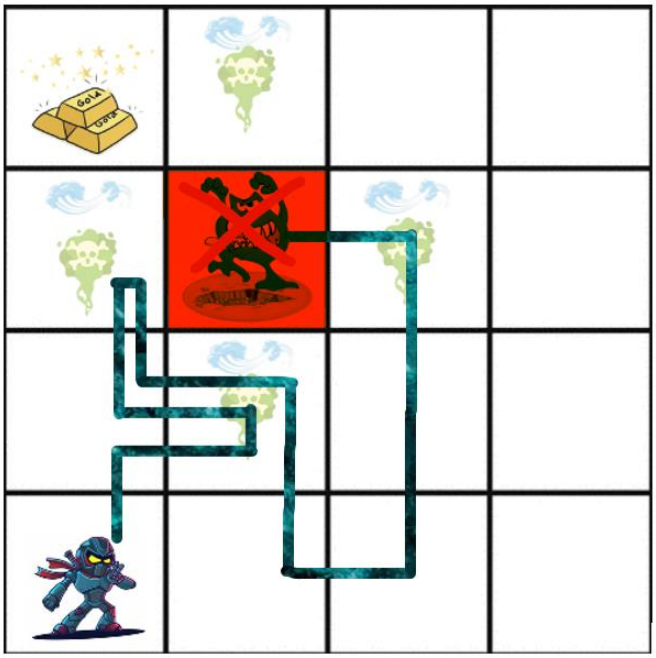


```
ACTING...
Found wumpus at (3,1)
C11 shoots 1 arrow at the wumpus in (3,1)and kills it!
SIMULATING...
SENSING...
THINKING...
C11 somewhat wants to go to (3,3).
C11 somewhat wants to go to (4,4).
C11 strongly wants to go to (3,1).
PLANNING...
ACTING...
C11 goes to (3,1).
```

```
Seeing glitter, C11 knows there is gold in (1,4).
C11 wants to pick up the gold in (1,4).
PLANNING...
ACTING...
C11 picks up 100 pieces of gold!
```

<u>12.jess:</u> We expect the agent to fail.

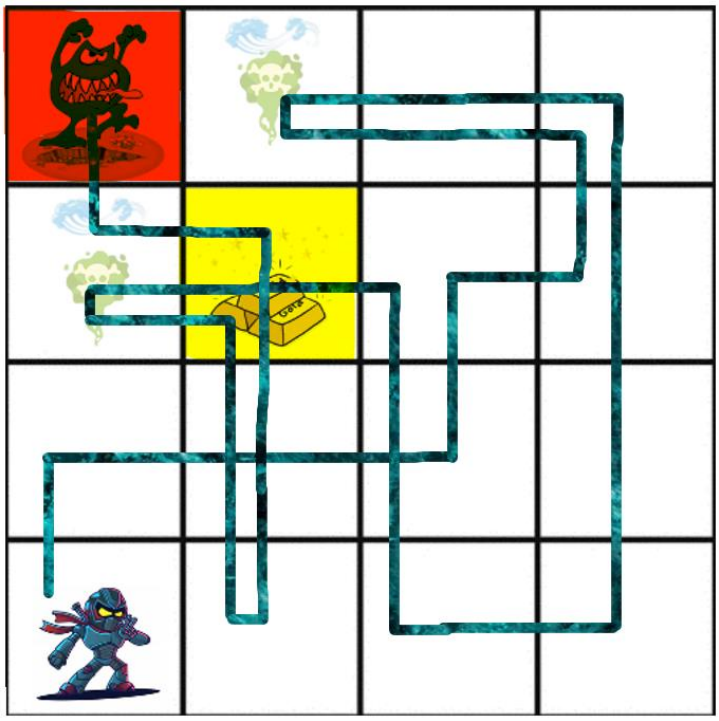
Indeed, the agent fails! (TN)



```
C12 strongly wants to shoot the wumpus at (2,3).
C12 moderately wants to go to (4,2).
C12 moderately wants to go to (4,1).
C12 somewhat wants to go to (1,4).
The pit is in (1,4).
A breeze in (3,3), so there may be a pit in (4,3).
With stench in (3,3), maybe the wumpus is in (4,3).
C12 somewhat wants to go to (4,3).
With stench in (3,3), maybe the wumpus is in (3,4).
A breeze in (3,3), so there may be a pit in (3,4).
C12 somewhat wants to go to (3,4).
Seeing no glitter, C12 knows there is no gold in (3,3).
PLANNING...
ACTING...
Found wumpus at (2,3)
C12 shoots 1 arrow at the wumpus in (2,3)and kills it!
SIMULATING...
SENSING...
THINKING...
C12 moderately wants to go to (4,2).
C12 moderately wants to go to (4,1).
C12 strongly wants to go to (2,3).
C12 somewhat wants to go to (3,4).
C12 somewhat wants to go to (4,3).
(2,3) is safe, so there's no pit or wumpus in it.
C12 strongly wants to go to (2,3).
PLANNING...
ACTING...
C12 goes to (2,3).
SIMULATING...
Aaarrrggghhhhhh....plop
241
Jess>
```

13.jess: *We expect the agent to succeed.*

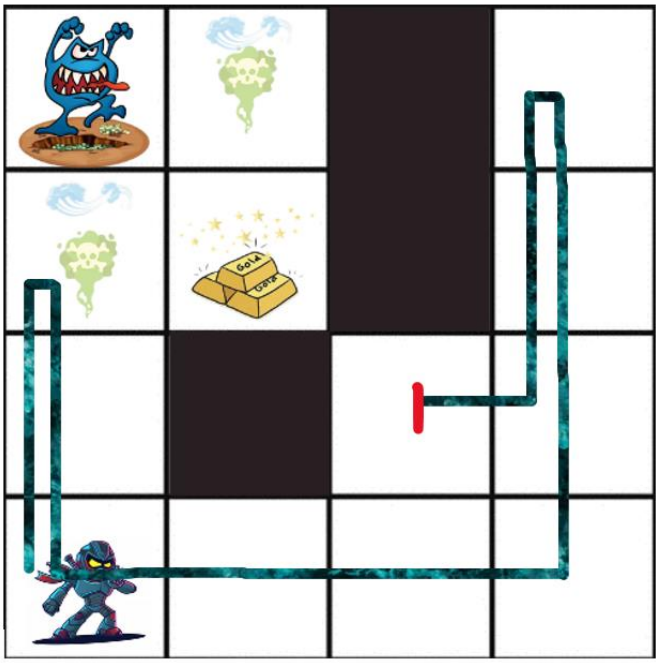
However, the agent fails! (FN)



```
The pit is in (1,4).
C13 moderately wants to go to (2,1).
Seeing glitter, C13 knows there is gold in (2,3).
C13 wants to pick up the gold in (2,3).
C13 strongly wants to go to (1,3).
PLANNING...
ACTING...
C13 picks up 100 pieces of gold!
SIMULATING...
SENSING...
THINKING...
C13 moderately wants to go to (2,1).
C13 strongly wants to go to (1,3).
PLANNING...
ACTING...
C13 goes to (1,3).
SIMULATING...
SENSING...
C13 sees no glitter.
C13 feels a breeze in (1,3).
C13 smells a stench.
THINKING...
C13 moderately wants to go to (2,1).
Seeing no glitter, C13 knows there is no gold in (1,3).
PLANNING...
ACTING...
C13 goes to (2,3) trying to get to (2,1).
C13 goes to (2,2) trying to get to (2,1).
C13 goes to (2,1).
SIMULATING...
SENSING...
C13 sees no glitter.
C13 smells nothing.
C13 feels no breeze in (2,1).
THINKING...
Seeing no glitter, C13 knows there is no gold in (2,1).
PLANNING...
ACTING...
C13 goes to (2,2) trying to get to (1,4).
C13 goes to (2,3) trying to get to (1,4).
C13 goes to (1,3) trying to get to (1,4).
C13 goes to (1,4).
SIMULATING...
Aaarrrggghhhhhh....plop
406
```

14.jess: *We expect the agent to succeed.*

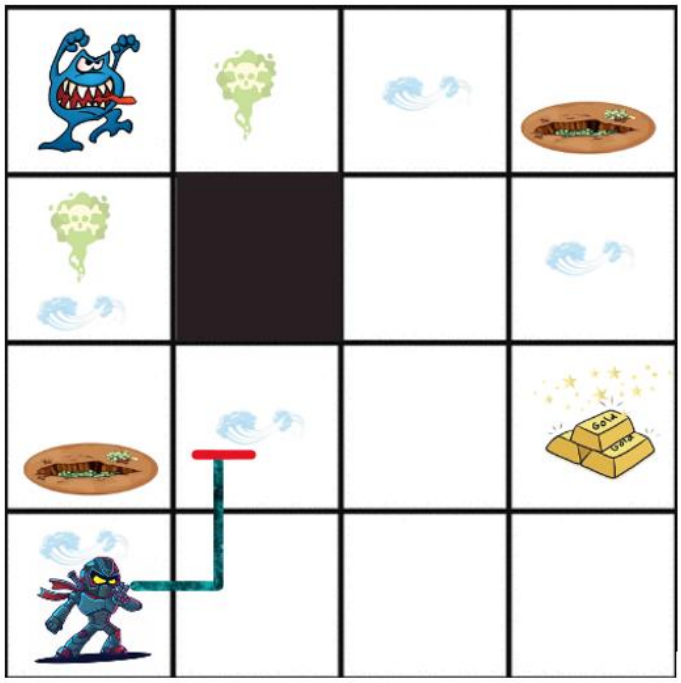
However, the agent fails! (FN)



```
PLANNING...
ACTING...
C14 goes to (4,3) trying to get to (3,2).
C14 goes to (4,2) trying to get to (3,2).
C14 goes to (3,2).
SIMULATING...
SENSING...
C14 sees no glitter.
C14 smells nothing.
C14 feels no breeze in (3,2).
THINKING...
C14 somewhat wants to go to (2,3).
C14 somewhat wants to go to (1,4).
Seeing no glitter, C14 knows there is no gold in (3,2).
PLANNING...
ACTING...
WARNING: unsatisfied goal: go 2 3.
259
```

<u>15.jess:</u> We expect the agent to succeed.

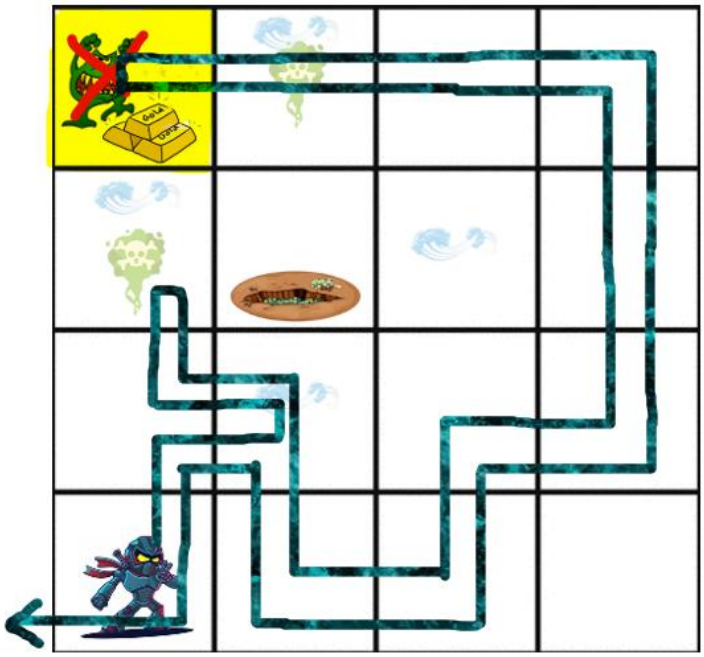
However, the agent fails! (FN)



```
C15 goes to (2,2).
SIMULATING...
SENSING...
C15 sees no glitter.
C15 feels a breeze in (2,2).
C15 notices (3,2) nearby.
C15 smells nothing.
THINKING...
C15 moderately wants to go to (3,1).
A breeze in (2,2), so there may be a pit in (3,2).
No stench in (2,2) means no wumpus in (3,2).
C15 somewhat wants to go to (3,2).
Seeing no glitter, C15 knows there is no gold in (2,2).
PLANNING...
ACTING...
WARNING: unsatisfied goal: go 3 1.
79
```

<u>16.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

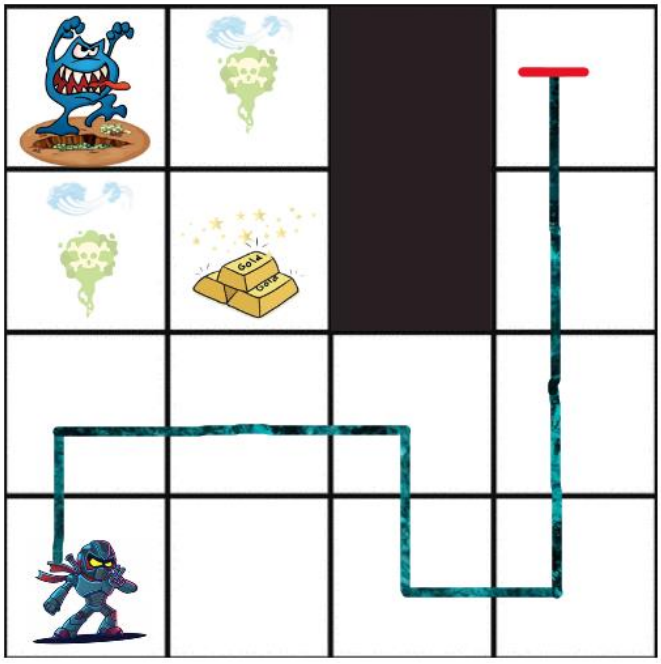


```
ACTING...
Found wumpus at (1,4)
C16 shoots 1 arrow at the wumpus in (1,4)and kills it!
```

```
THINKING...
C16 moderately wants to go to (3,3).
C16 moderately wants to go to (4,1).
Seeing glitter, C16 knows there is gold in (1,4).
C16 wants to pick up the gold in (1,4).
PLANNING...
ACTING...
C16 picks up 100 pieces of gold!
```

<u>17.jess:</u> We expect the agent to succeed.

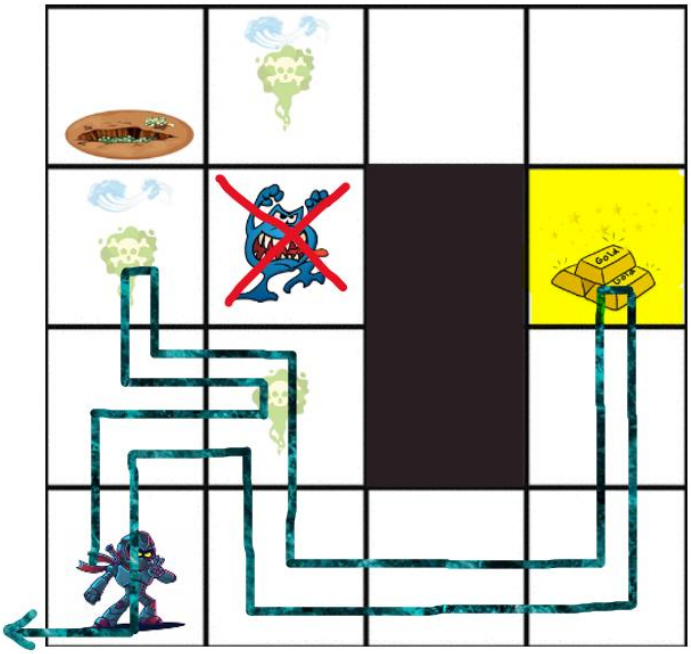
However, the agent fails! (FN)



```
PLANNING...
ACTING...
C17 goes to (4,4).
SIMULATING...
SENSING...
C17 sees no glitter.
C17 smells nothing.
C17 feels no breeze in (4,4).
THINKING...
C17 moderately wants to go to (2,3).
C17 moderately wants to go to (1,3).
C17 moderately wants to go to (2,1).
Seeing no glitter, C17 knows there is no gold in (4,4).
PLANNING...
ACTING...
C17 goes to (4,3) trying to get to (2,3).
WARNING: unsatisfied goal: go 2 3.
246
Jess>
```

<u>18.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)



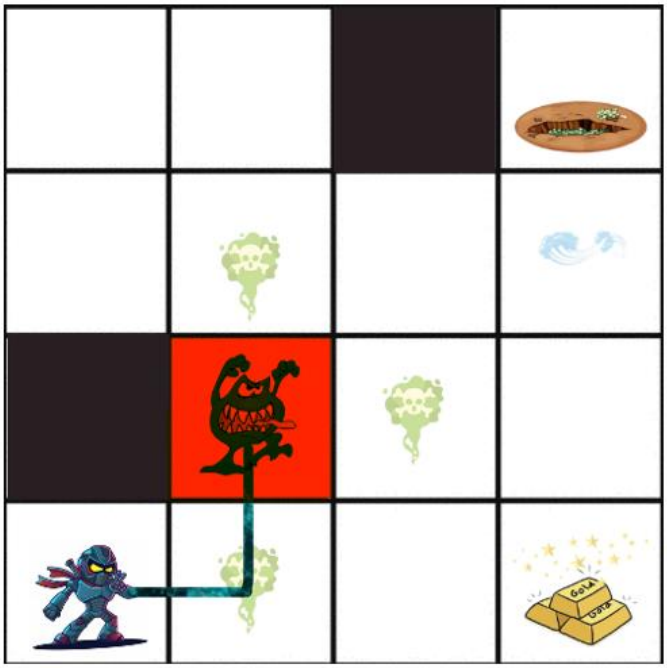
```
ACTING...
Found wumpus at (2,3)
C18 shoots 1 arrow at the wumpus in (2,3)and kills it!
```

```
C18 goes to (4,3).
SIMULATING...
SENSING...
C18 sees glitter.
C18 notices (4,4) nearby.
C18 smells nothing.
C18 feels no breeze in (4,3).
THINKING...
C18 moderately wants to go to (2,3).
C18 somewhat wants to go to (1,4).
There's no breeze in (4,3) so there's no pit  in (4,4).
No stench in (4,3) means no wumpus in (4,4).
C18 somewhat wants to go to (4,4).
With neither wumpus nor pit, (4,4) is safe.
C18 strongly wants to go to (4,4).
Seeing glitter, C18 knows there is gold in (4,3).
C18 wants to pick up the gold in (4,3).
PLANNING...
ACTING...
C18 picks up 100 pieces of gold!
```

<u>19.jess:</u> We expect the agent to fail.

Indeed, the agent fails! (TN)

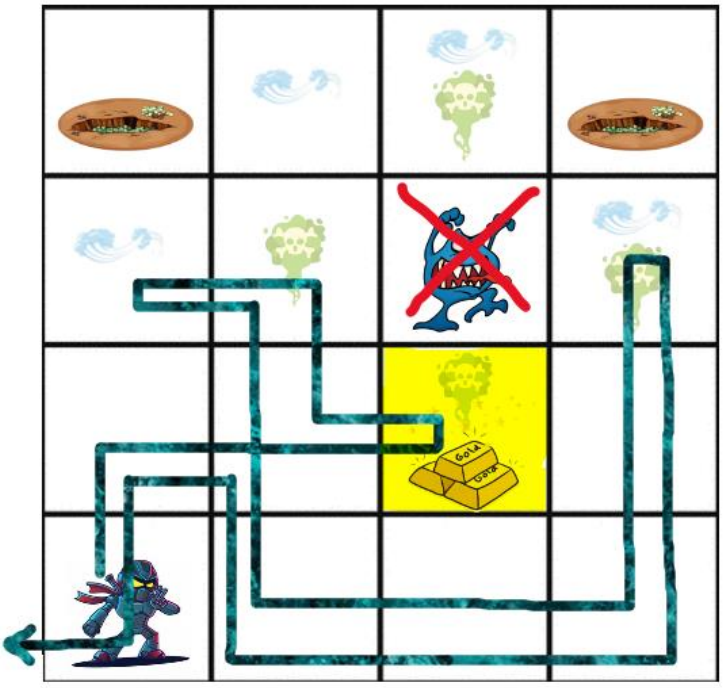
When the agent smells a stench, it goes to the cave above it, not next to it.



```
ACTING...
C19 goes to (2,1).
SIMULATING...
SENSING...
C19 sees no glitter.
C19 smells a stench.
C19 notices (2,2) nearby.
C19 notices (3,1) nearby.
C19 feels no breeze in (2,1).
THINKING...
With stench in (2,1), maybe the wumpus is in (3,1).
There's no breeze in (2,1) so there's no pit  in (3,1).
C19 somewhat wants to go to (3,1).
With stench in (2,1), maybe the wumpus is in (2,2).
There's no breeze in (2,1) so there's no pit  in (2,2).
C19 somewhat wants to go to (2,2).
Seeing no glitter, C19 knows there is no gold in (2,1).
PLANNING...
ACTING...
C19 goes to (2,2).
SIMULATING...
Aaarrrggghhhhhh....munch...munch...munch
58
```

<u>20.jess:</u> We expect the agent to succeed.

Indeed, the agent succeds! (TP)

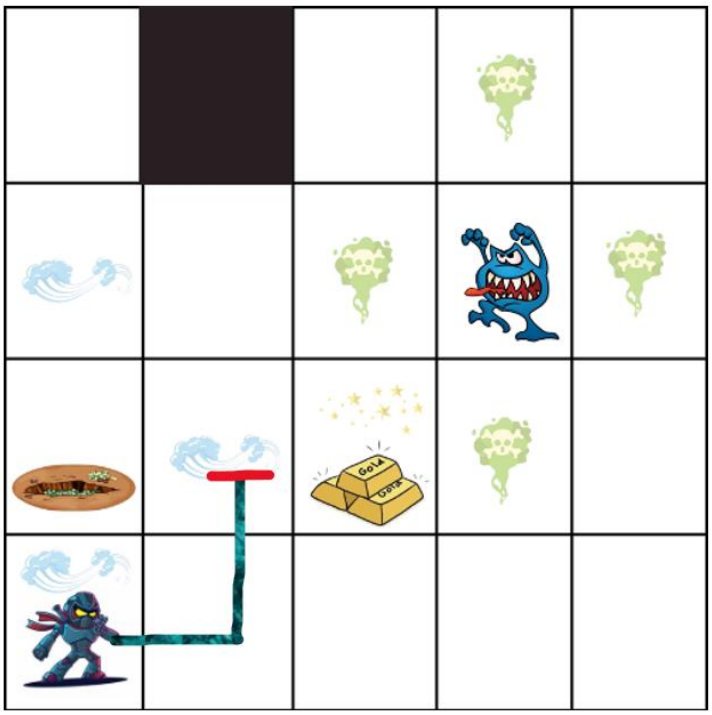


```
Seeing glitter, C20 knows there is gold in (3,2).
C20 wants to pick up the gold in (3,2).
PLANNING...
ACTING...
C20 picks up 100 pieces of gold!
```

```
C20 goes to (4,3).
SIMULATING...
SENSING...
C20 sees no glitter.
C20 feels a breeze in (4,3).
C20 smells a stench.
C20 notices (4,4) nearby.
THINKING...
C20 strongly wants to shoot the wumpus at (3,3).
C20 somewhat wants to go to (2,4).
A breeze in (4,3), so there may be a pit in (4,4).
The pit is in (4,4).
With stench in (4,3), maybe the wumpus is in (4,4).
Seeing no glitter, C20 knows there is no gold in (4,3).
PLANNING...
ACTING...
Found wumpus at (3,3)
C20 shoots 1 arrow at the wumpus in (3,3)and kills it!
```

<u>21.jess:</u> We expect the agent to succeed.

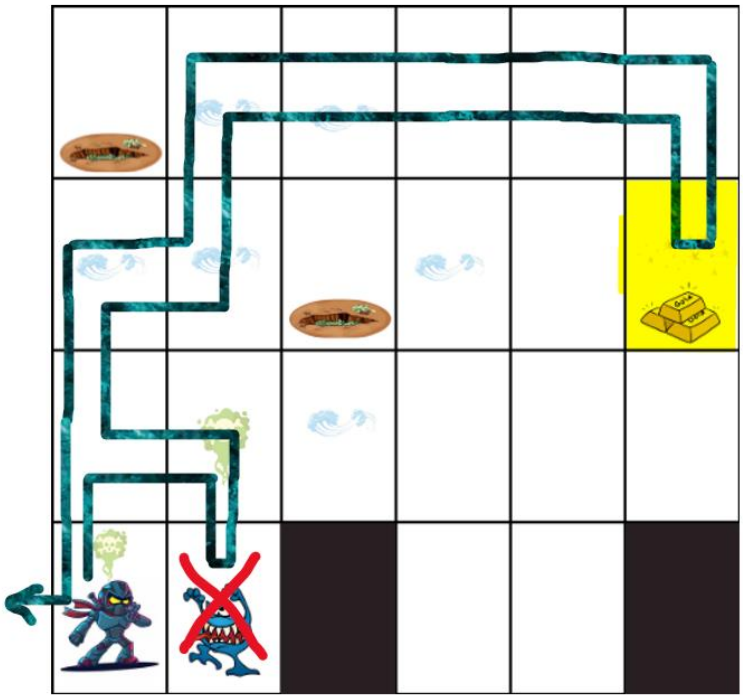
However, the agent fails! (FN)



```
ACTING...
C21 goes to (2,2).
SIMULATING...
SENSING...
C21 sees no glitter.
C21 feels a breeze in (2,2).
C21 notices (2,3) nearby.
C21 notices (3,2) nearby.
C21 smells nothing.
THINKING...
C21 moderately wants to go to (3,1).
No stench in (2,2) means no wumpus in (2,3).
A breeze in (2,2), so there may be a pit in (2,3).
C21 somewhat wants to go to (2,3).
A breeze in (2,2), so there may be a pit in (3,2).
No stench in (2,2) means no wumpus in (3,2).
C21 somewhat wants to go to (3,2).
Seeing no glitter, C21 knows there is no gold in (2,2).
PLANNING...
ACTING...
WARNING: unsatisfied goal: go 3 1.
81
```

<u>22.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

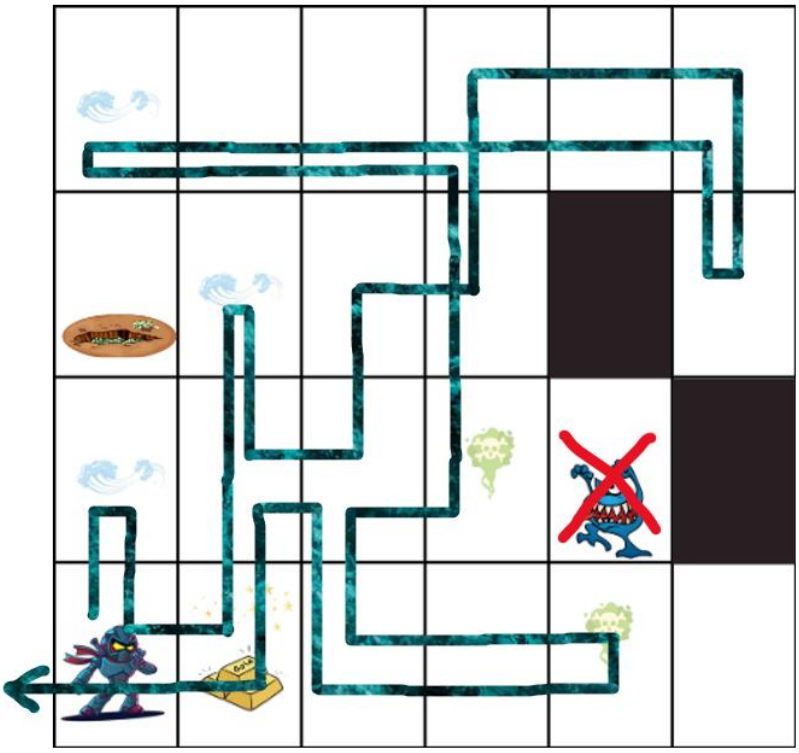


```
ACTING...
Found wumpus at (2,1)
C22 shoots 1 arrow at the wumpus in (2,1)and kills it!
```

```
C22 goes to (6,3).
SIMULATING...
SENSING...
C22 sees glitter.
C22 notices (6,2) nearby.
C22 smells nothing.
C22 feels no breeze in (6,3).
THINKING...
C22 moderately wants to go to (4,3).
C22 somewhat wants to go to (3,2).
There's no breeze in (6,3) so there's no pit  in (6,2).
No stench in (6,3) means no wumpus in (6,2).
C22 somewhat wants to go to (6,2).
With neither wumpus nor pit, (6,2) is safe.
C22 strongly wants to go to (6,2).
C22 strongly wants to go to (5,3).
Seeing glitter, C22 knows there is gold in (6,3).
C22 wants to pick up the gold in (6,3).
PLANNING...
ACTING...
C22 picks up 100 pieces of gold!
```

<u>23.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

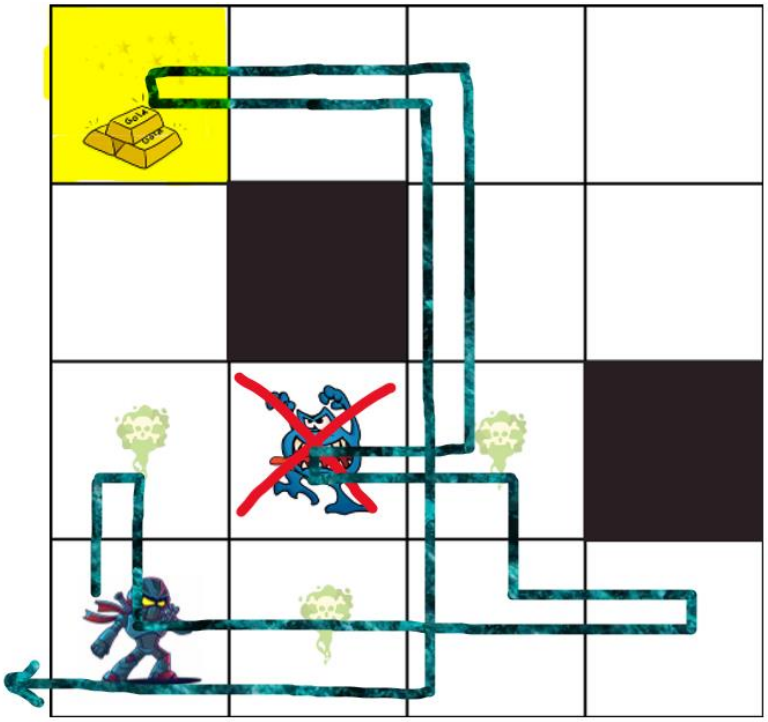


```
Seeing glitter, C23 knows there is gold in (2,1).
C23 wants to pick up the gold in (2,1).
PLANNING...
ACTING...
C23 picks up 100 pieces of gold!
```

```
C23 smells a stench.
C23 notices (6,1) nearby.
C23 feels no breeze in (5,1).
THINKING...
C23 strongly wants to shoot the wumpus at (5,2).
With stench in (5,1), maybe the wumpus is in (6,1).
There's no breeze in (5,1) so there's no pit  in (6,1).
C23 somewhat wants to go to (6,1).
Seeing no glitter, C23 knows there is no gold in (5,1).
PLANNING...
ACTING...
Found wumpus at (5,2)
C23 shoots 1 arrow at the wumpus in (5,2)and kills it!
```

<u>24.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)

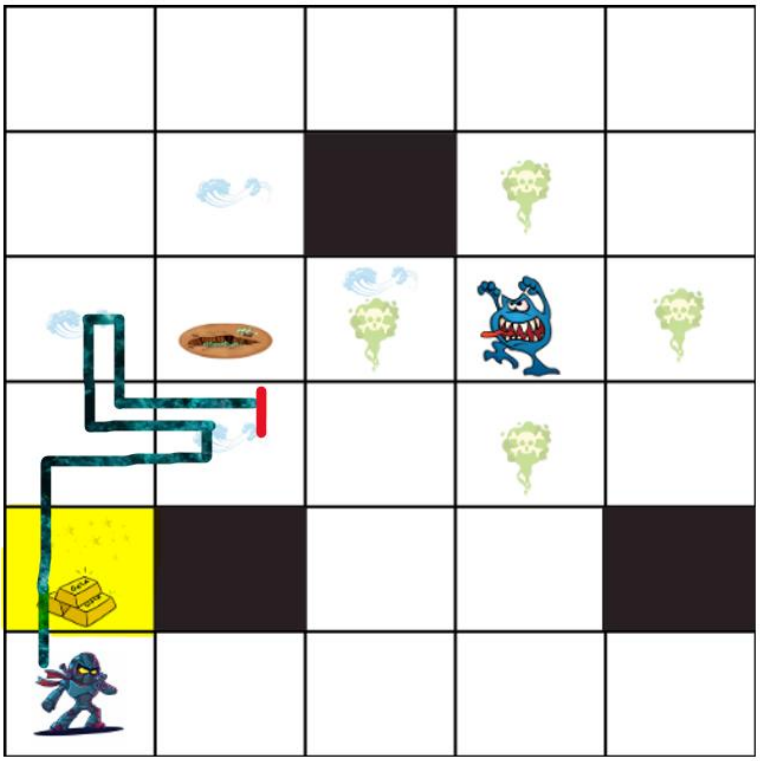


```
C24 sees no glitter.
C24 smells a stench.
C24 notices (3,3) nearby.
C24 feels no breeze in (3,2).
THINKING...
C24 strongly wants to shoot the wumpus at (2,2).
C24 somewhat wants to go to (1,3).
There's no breeze in (3,2) so there's no pit  in (3,3).
With stench in (3,2), maybe the wumpus is in (3,3).
C24 somewhat wants to go to (3,3).
Seeing no glitter, C24 knows there is no gold in (3,2).
PLANNING...
ACTING...
Found wumpus at (2,2)
C24 shoots 1 arrow at the wumpus in (2,2)and kills it!
```

```
C24 sees glitter.
C24 smells nothing.
C24 feels no breeze in (1,4).
THINKING...
C24 moderately wants to go to (4,4).
C24 moderately wants to go to (4,3).
Seeing glitter, C24 knows there is gold in (1,4).
C24 wants to pick up the gold in (1,4).
No stench in (1,4) means no wumpus in (1,3).
C24 somewhat wants to go to (1,3).
With neither wumpus nor pit, (1,3) is safe.
C24 strongly wants to go to (1,3).
PLANNING...
ACTING...
C24 picks up 100 pieces of gold!
```

<u>25.jess:</u> We expect the agent to succeed.

However, the agent fails! (FN)

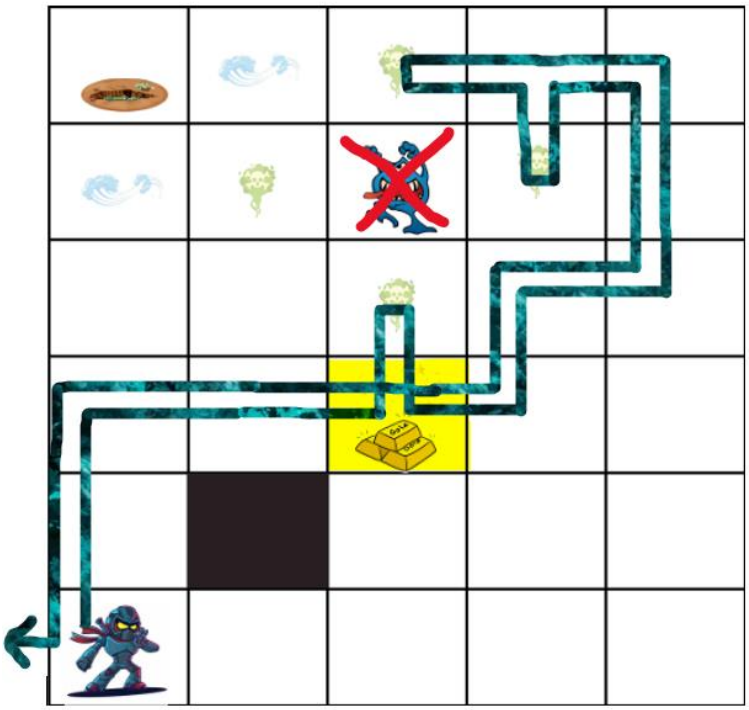


```
C25 goes to (1,2).
SIMULATING...
SENSING...
C25 sees glitter.
C25 notices (1,3) nearby.
C25 smells nothing.
C25 feels no breeze in (1,2).
THINKING...
C25 moderately wants to go to (2,1).
There's no breeze in (1,2) so there's no pit  in (1,3).
No stench in (1,2) means no wumpus in (1,3).
C25 somewhat wants to go to (1,3).
With neither wumpus nor pit, (1,3) is safe.
C25 strongly wants to go to (1,3).
Seeing glitter, C25 knows there is gold in (1,2).
C25 wants to pick up the gold in (1,2).
PLANNING...
ACTING...
C25 picks up 100 pieces of gold!
```

```
ACTING...
C25 goes to (1,3) trying to get to (2,1).
C25 goes to (2,3) trying to get to (2,1).
WARNING: unsatisfied goal: go 2 1.
161
Jess>
```

<u>26.jess:</u> We expect the agent to succeed.

Indeed, the agent succeeds! (TP)



```
C26 wants to pick up the gold in (3,3).
PLANNING...
ACTING...
C26 picks up 100 pieces of gold!
```

```
C26 goes to (4,5).
SIMULATING...
SENSING...
C26 sees no glitter.
C26 smells a stench.
C26 feels no breeze in (4,5).
THINKING...
C26 somewhat wants to go to (2,6).
C26 moderately wants to go to (5,3).
C26 moderately wants to go to (4,2).
C26 moderately wants to go to (3,2).
C26 moderately wants to go to (2,4).
C26 moderately wants to go to (1,4).
C26 moderately wants to go to (2,1).
The wumpus is in (3,5).
C26 strongly wants to shoot the wumpus at (3,5).
Seeing no glitter, C26 knows there is no gold in (4,5).
PLANNING...
ACTING...
Found wumpus at (3,5)
C26 shoots 1 arrow at the wumpus in (3,5)and kills it!
```

Conclusion

In conclusion, our project has successfully tackled the classic game of "Wumpus" and demonstrated the power of Artificial Intelligence to enhance the gameplay experience. Through a series of well-defined tasks, we have empowered our AI agent to navigate the challenging Wumpus World, make strategic decisions, and adapt dynamically to various scenarios. The following key challenges emerge from our project:

1) Enhancing Deductive Capabilities:

In Task 1, we focused on augmenting the Hunter's deductive abilities. We went beyond merely marking caves as "possibly containing the Wumpus" and empowered the system to deduce the exact location of the Wumpus and pits when sufficient information was available. This greatly improved the Hunter's precision in navigating the Wumpus World.

2) Adding a New Goal:

Task 2 introduced a significant enhancement by adding the goal of eliminating the Wumpus to the Hunter's objectives. We equipped the Hunter with a new action, "shoot," allowing it to take out the Wumpus, changing the dynamics of the game.

3) Improved Navigation and Goal Setting:

In Task 3, we addressed scenarios where the Hunter could get stuck in a cave. We made sure that our agent's ability to set goals and navigate to distant caves is well defined, making the Hunter a more dynamic and adaptive explorer.

4) Smart Decision-Making:

Task 4 introduced an extra layer of intelligence, enabling the Hunter to choose the best action when faced with many possibilities. It added a valuable dimension to the game by prioritizing actions such as picking up gold or shooting the Wumpus.

5) Precision Assessment:

In Task 5, we evaluated the precision of our AI agent across 50 random scenarios in the Wumpus World. The agent exhibited impressive precision with a 100% success rate in correctly detecting the Wumpus and picking up gold. This analysis helped us understand where the agent excels and where there is room for improvement.

Challenges:

Throughout the development of this project, we encountered several notable challenges. The initial one involved familiarizing ourselves with the JESS environment, a robust rule-based expert system shell. This required a comprehensive understanding of its functions and scripting language, which was pivotal in implementing the AI agent's decision-making capabilities. Another significant challenge was comprehending the intricate rules of the Wumpus game and adapting them into a coherent framework. This process involved translating the complexities of the game's mechanics into a rule-based system, requiring precise logic and meticulous planning. Additionally, tracing the

agent’s path in various scenarios demanded deep patience and a lot of focus. By addressing these challenges, we managed to craft an effective AI agent.

Final Note:

Throughout this project we recognized that there are numerous exciting opportunities for future extensions. One avenue involves enhancing the system’s adaptability and learning capabilities, enabling it to evolve and make more informed decisions as it encounters new cave configurations. Incorporating additional dimensions of intelligence, such as machine learning or deep reinforcement learning techniques, could open doors to improved gameplay. Moreover, expanding the Wumpus World to multi-agent environments could introduce competitive elements and further complexity. The project’s framework can also serve as a stepping stone for real-time simulations, introducing dynamic elements like moving Wumpuses or evolving caves, thereby increasing the overall complexity and engagement of the game. The path forward is rich with possibilities, offering the chance to advance our AI agent’s skills and exploring more sophisticated challenges within the Wumpus World and other AI applications.

References

(n.d.). Retrieved from javaTpoint: https://www.javatpoint.com/the-wumpus-world-in-artificial-intelligence

Siriwardhane, P. (2022, February 2). Retrieved October 28, 2023, from Medium: https://medium.com/nerd-for-tech/an-introduction-to-expert-system-shells-530043914ec0