SCHOOL OF SCIENCE & ENGINEERING

CSC 4301 Intro. To Artificial Intelligence

Project#2

Realized by:

Imane Bouchtaoui, Chahd Maatallaoui

Supervised by:

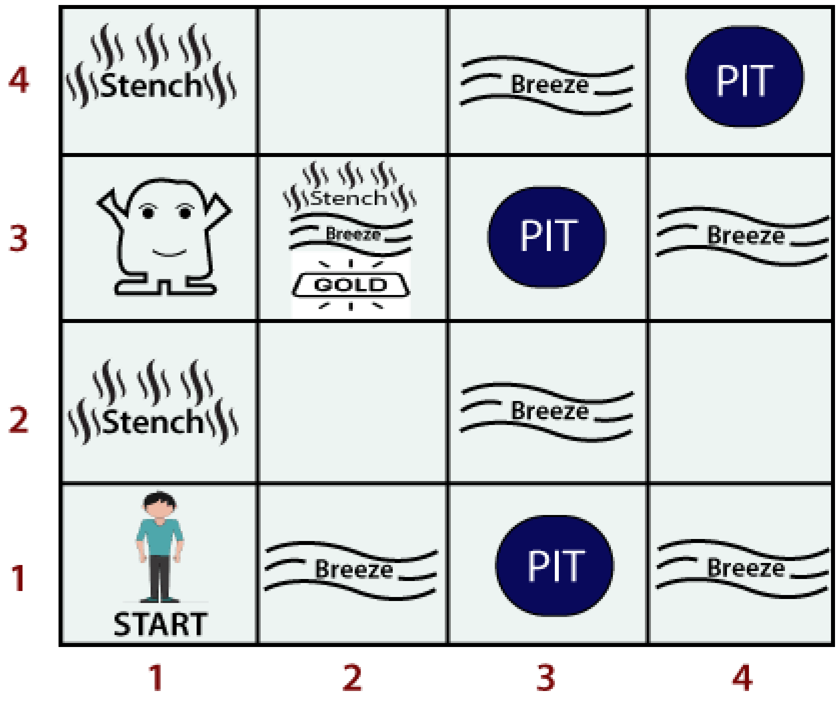
Professor Rachidi Tajjeddine

Spring 2024

Table of Contents

Introduction:

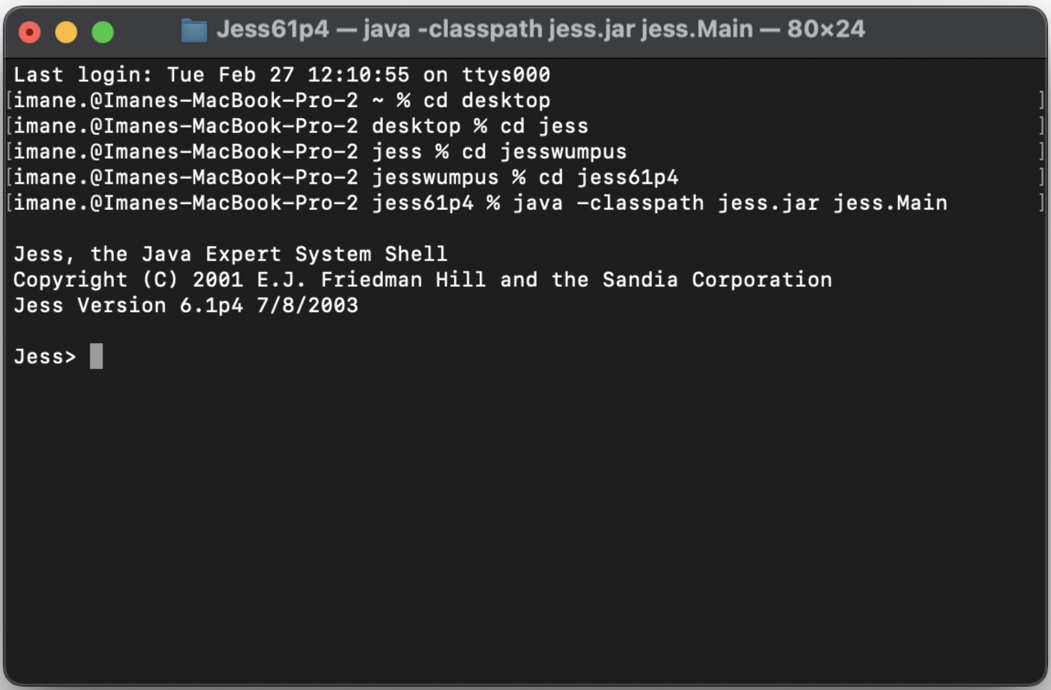
The Wumpus World game is a cornerstone in the development of intelligent agents. Here, a knowledge-based agent navigates a cave, facing hazards like pits and the terrifying Wumpus. Its goal is to find gold and return safely. This project aims to enhance the agent's decision-making for efficient and accurate task completion. Wumpus World implementations exist in various languages, but this project utilizes Jess, a rule engine for Java. Jess empowers the construction of robust rule-based expert systems perfectly suited for our agent's decision-making improvements. The agent relies on sensors to navigate the cave. These sensors detect breezes (indicating nearby pits), stenches (signaling the Wumpus' presence), and glitters (pointing towards gold). As the agent explores, these sensory inputs are stored in its knowledge base, guiding its decisions and movements.



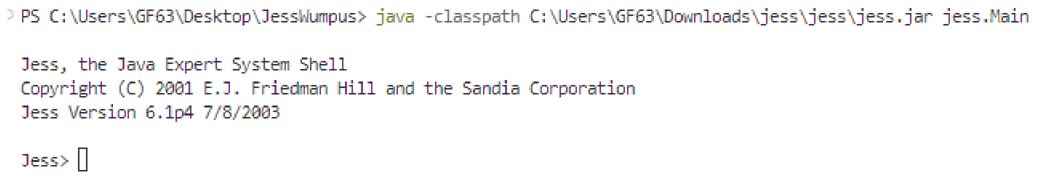
A Typical Wumpus World

Jess Setup:

Before modifying the agent, we'll need to create a suitable environment for testing and execution. First, we downloaded the Jess file including the tools and documentation we require. We can then launch Jess by running the java command in the terminal to get the following result:

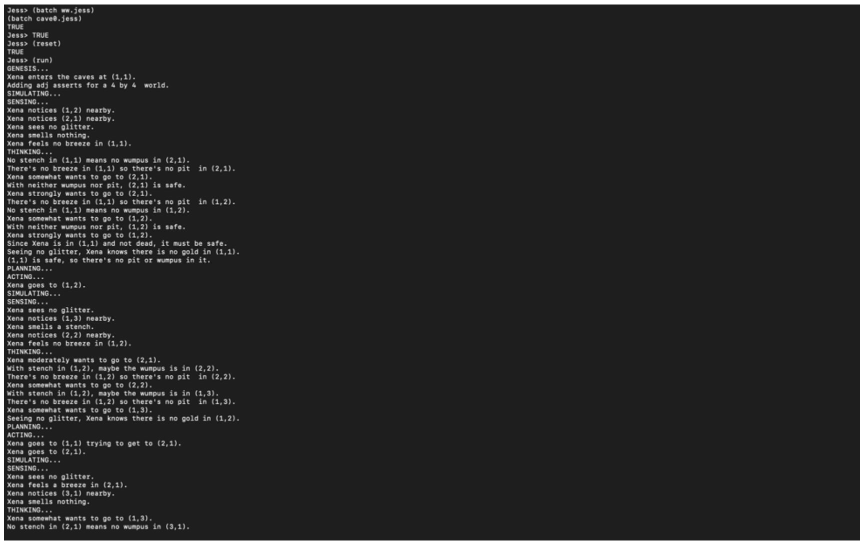


```
Last login: Tue Feb 27 12:10:55 on ttys000
imane.@Imanes-MacBook-Pro-2 ~ % cd desktop
imane.@Imanes-MacBook-Pro-2 desktop % cd jess
imane.@Imanes-MacBook-Pro-2 jess % cd jesswumpus
imane.@Imanes-MacBook-Pro-2 jesswumpus % cd jess61p4
imane.@Imanes-MacBook-Pro-2 jess61p4 % java -classpath jess.jar jess.Main

Jess, the Java Expert System Shell
Copyright (C) 2001 E.J. Friedman Hill and the Sandia Corporation
Jess Version 6.1p4 7/8/2003

Jess> 
```

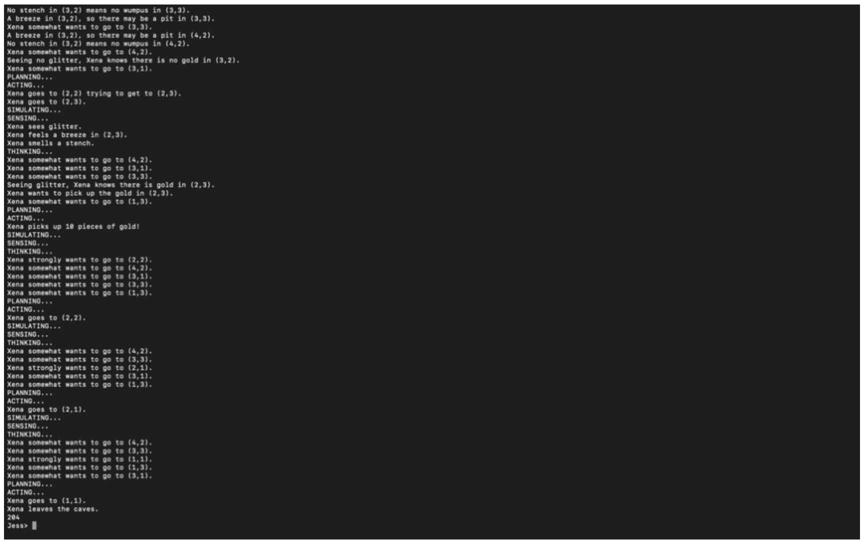


```
PS C:\Users\GF63\Desktop\JessWumpus> java -classpath C:\Users\GF63\Downloads\jess\jess\jess.jar jess.Main

Jess, the Java Expert System Shell
Copyright (C) 2001 E.J. Friedman Hill and the Sandia Corporation
Jess Version 6.1p4 7/8/2003

Jess> 
```

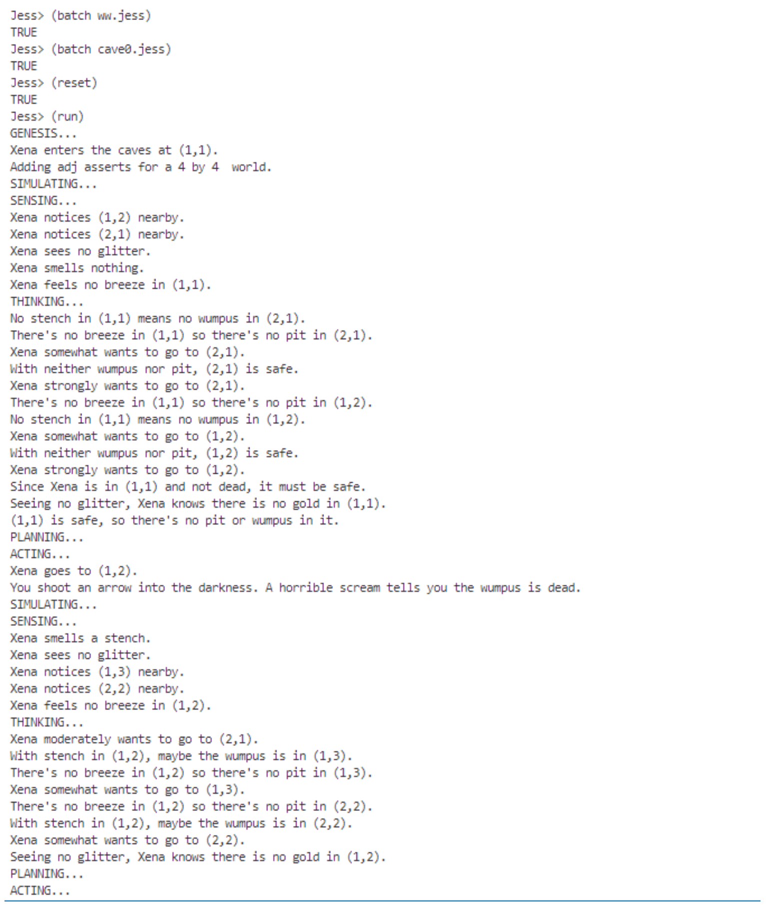
Then, we add the knowledge base and cave files to compile the Jess code and simulate the Wumpus universe. After batching the files and running the application, the terminal displays the following output:



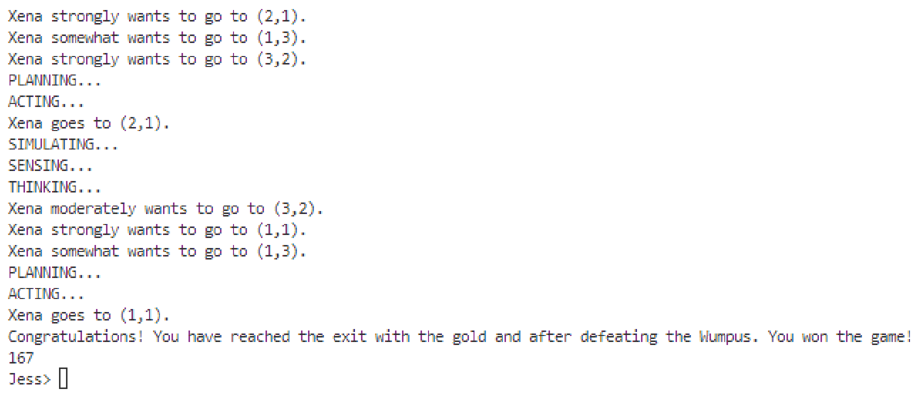
```
Jess> (batch ww.jess)
(batch cave0.jess)
TRUE
Jess> TRUE
Jess> (reset)
TRUE
Jess> (run)
GENESIS...
Xena enters the caves at (1,1).
Adding adj asserts for a 4 by 4  world.
SIMULATING...
SENSING...
Xena notices (1,2) nearby.
Xena notices (2,1) nearby.
Xena sees no glitter.
Xena smells nothing.
Xena 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).
Xena somewhat wants to go to (2,1).
With neither wumpus nor pit, (2,1) is safe.
Xena 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).
Xena somewhat wants to go to (1,2).
With neither wumpus nor pit, (1,2) is safe.
Xena strongly wants to go to (1,2).
Since Xena is in (1,1) and not dead, it must be safe.
Seeing no glitter, Xena knows there is no gold in (1,1).
(1,1) is safe, so there's no pit or wumpus in it.
PLANNING...
ACTING...
Xena goes to (1,2).
SIMULATING...
SENSING...
Xena sees no glitter.
Xena notices (1,3) nearby.
Xena smells a stench.
Xena notices (2,2) nearby.
Xena feels no breeze in (1,2).
THINKING...
Xena moderately wants to go to (2,1).
With stench in (1,2), maybe the wumpus is in (2,2).
There's no breeze in (1,2) so there's no pit  in (2,2).
Xena somewhat wants to go to (2,2).
With stench in (1,2), maybe the wumpus is in (1,3).
There's no breeze in (1,2) so there's no pit  in (1,3).
Xena somewhat wants to go to (1,3).
Seeing no glitter, Xena knows there is no gold in (1,2).
PLANNING...
ACTING...
Xena goes to (1,1) trying to get to (2,1).
Xena goes to (2,1).
SIMULATING...
SENSING...
Xena sees no glitter.
Xena feels a breeze in (2,1).
Xena notices (3,1) nearby.
Xena smells nothing.
THINKING...
Xena somewhat wants to go to (1,3).
No stench in (2,1) means no wumpus in (3,1).
```



```
No stench in (3,2) means no wumpus in (3,3).
A breeze in (3,2), so there may be a pit in (3,3).
Xena somewhat wants to go to (3,3).
A breeze in (3,2), so there may be a pit in (4,2).
No stench in (3,2) means no wumpus in (4,2).
Xena somewhat wants to go to (4,2).
Seeing no glitter, Xena knows there is no gold in (3,2).
Xena somewhat wants to go to (3,1).
PLANNING...
ACTING...
Xena goes to (2,2) trying to get to (2,3).
Xena goes to (2,3).
SIMULATING...
SENSING...
Xena sees glitter.
Xena feels a breeze in (2,3).
Xena smells a stench.
THINKING...
Xena somewhat wants to go to (4,2).
Xena somewhat wants to go to (3,1).
Xena somewhat wants to go to (3,3).
Seeing glitter, Xena knows there is gold in (2,3).
Xena wants to pick up the gold in (2,3).
Xena somewhat wants to go to (1,3).
PLANNING...
ACTING...
Xena picks up 10 pieces of gold!
SIMULATING...
SENSING...
THINKING...
Xena strongly wants to go to (2,2).
Xena somewhat wants to go to (4,2).
Xena somewhat wants to go to (3,1).
Xena somewhat wants to go to (3,3).
Xena somewhat wants to go to (1,3).
PLANNING...
ACTING...
Xena goes to (2,2).
SIMULATING...
SENSING...
THINKING...
Xena somewhat wants to go to (4,2).
Xena somewhat wants to go to (3,3).
Xena strongly wants to go to (2,1).
Xena somewhat wants to go to (3,1).
Xena somewhat wants to go to (1,3).
PLANNING...
ACTING...
Xena goes to (2,1).
SIMULATING...
SENSING...
THINKING...
Xena somewhat wants to go to (4,2).
Xena somewhat wants to go to (3,3).
Xena strongly wants to go to (1,1).
Xena somewhat wants to go to (1,3).
Xena somewhat wants to go to (3,1).
PLANNING...
ACTING...
Xena goes to (1,1).
Xena leaves the caves.
204
Jess> 
```



```
Jess> (batch ww.jess)
TRUE
Jess> (batch cave0.jess)
TRUE
Jess> (reset)
TRUE
Jess> (run)
GENESIS...
Xena enters the caves at (1,1).
Adding adj asserts for a 4 by 4  world.
SIMULATING...
SENSING...
Xena notices (1,2) nearby.
Xena notices (2,1) nearby.
Xena sees no glitter.
Xena smells nothing.
Xena 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).
Xena somewhat wants to go to (2,1).
With neither wumpus nor pit, (2,1) is safe.
Xena 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).
Xena somewhat wants to go to (1,2).
With neither wumpus nor pit, (1,2) is safe.
Xena strongly wants to go to (1,2).
Since Xena is in (1,1) and not dead, it must be safe.
Seeing no glitter, Xena knows there is no gold in (1,1).
(1,1) is safe, so there's no pit or wumpus in it.
PLANNING...
ACTING...
Xena goes to (1,2).
You shoot an arrow into the darkness. A horrible scream tells you the wumpus is dead.
SIMULATING...
SENSING...
Xena smells a stench.
Xena sees no glitter.
Xena notices (1,3) nearby.
Xena notices (2,2) nearby.
Xena feels no breeze in (1,2).
THINKING...
Xena moderately wants to go to (2,1).
With stench in (1,2), maybe the wumpus is in (1,3).
There's no breeze in (1,2) so there's no pit in (1,3).
Xena somewhat wants to go to (1,3).
There's no breeze in (1,2) so there's no pit in (2,2).
With stench in (1,2), maybe the wumpus is in (2,2).
Xena somewhat wants to go to (2,2).
Seeing no glitter, Xena knows there is no gold in (1,2).
PLANNING...
ACTING...
```



```
Xena strongly wants to go to (2,1).
Xena somewhat wants to go to (1,3).
Xena strongly wants to go to (3,2).
PLANNING...
ACTING...
Xena goes to (2,1).
SIMULATING...
SENSING...
THINKING...
Xena moderately wants to go to (3,2).
Xena strongly wants to go to (1,1).
Xena somewhat wants to go to (1,3).
PLANNING...
ACTING...
Xena goes to (1,1).
Congratulations! You have reached the exit with the gold and after defeating the Wumpus. You won the game!
167
Jess> 
```

*This output confirms that indeed the hunter successfully navigated the "cave0". We will now proceed by altering the code to accomplish the necessary tasks and improve the agent's algorithm.

Project Tasks:

- **Task 1: Improve the Hunter's reasoning ability**

To enhance the game's strategic depth and the Hunter's chances of success, this task proposes an upgrade to the Hunter's reasoning capabilities. The goal is to equip the system with new rules that enable it to leverage sensory data and logical deduction to pinpoint the precise location of the Wumpus whenever possible. This improvement has the potential to significantly bolster the Hunter's overall effectiveness and substantially increase its odds of emerging victorious.

(i). Code changes:

1. Identifying Potential Wumpus Locations:

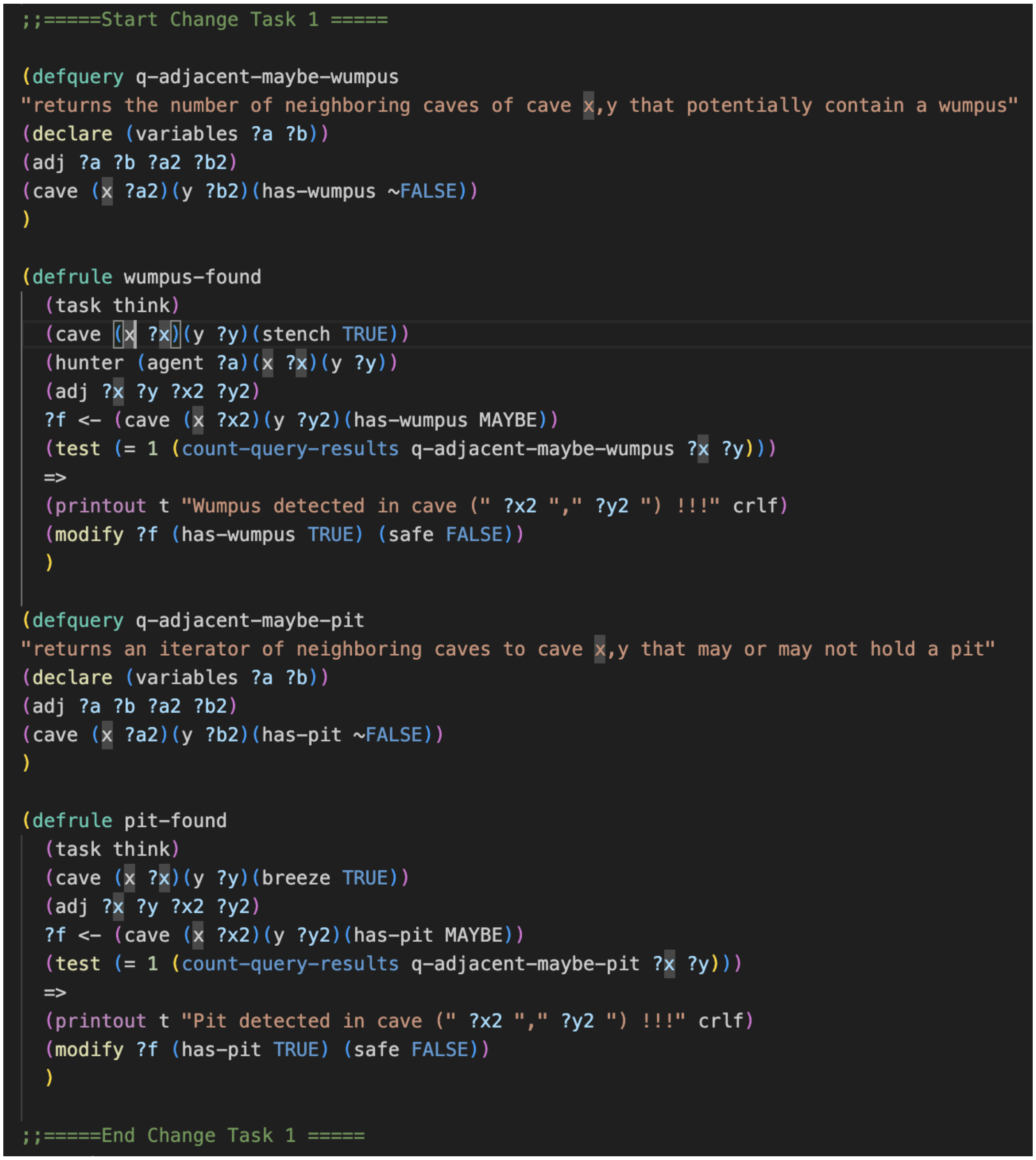
A new function named q-adjacent-maybe-Wumpus has been implemented. This function identifies squares adjacent to the current location that **could potentially harbor the Wumpus** based on existing sensory data, such as the presence of a stench. This information is crucial for the system to narrow down the Wumpus's search space and make informed decisions about its next move.

2. Confirming Wumpus Location:

A new rule named Wumpus-found has been introduced. This rule leverages the information gathered by q-adjacent-maybe-Wumpus and potentially other sensory data to **deduce the Wumpus's definitive location** with certainty. This definitive identification is crucial for the system to take decisive action, such as attacking the Wumpus or navigating away from its location.

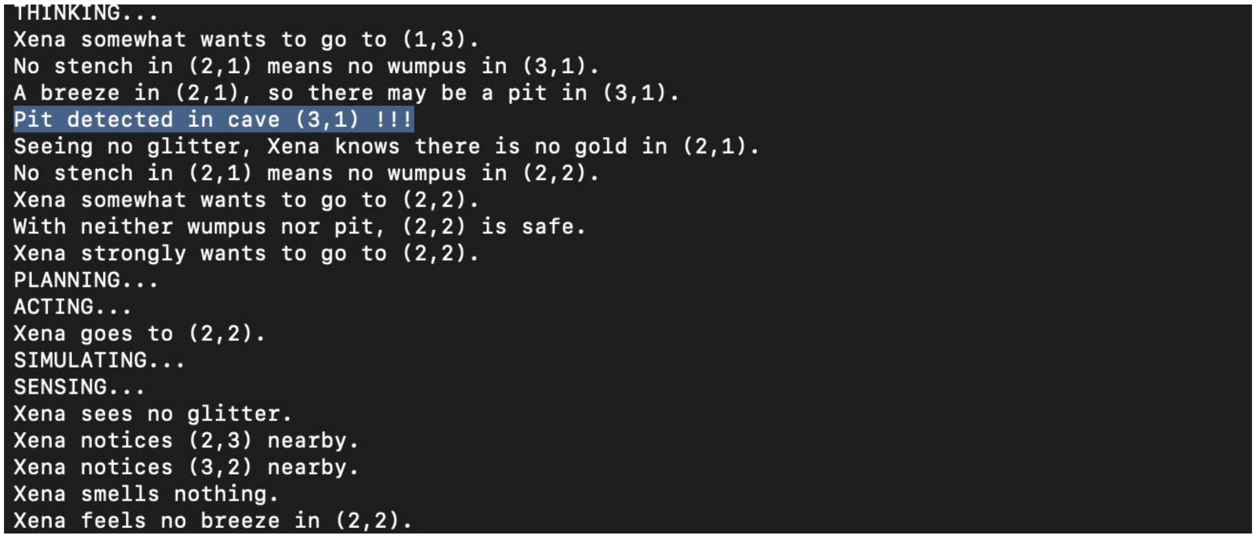
3. Extending Functionality to Pits:

Following the same logic as the Wumpus, a new function named q-adjacent-maybe-pit has been implemented to identify squares adjacent to the current location that **could potentially contain a pit**. Additionally, a new rule named pit-found has been introduced to **determine the exact location of a pit** based on the information gathered by the function and potentially other sensory data. This functionality mirrors the approach taken for the Wumpus, allowing the system to reason about and avoid potential hazards.

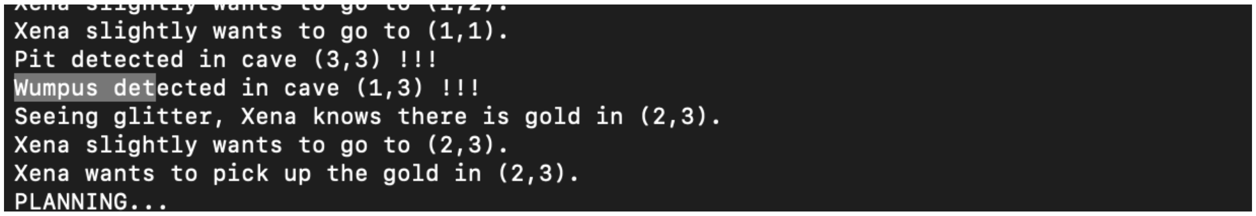


```
;;=====Start Change Task 1 =====

(defquery q-adjacent-maybe-wumpus
"returns the number of neighboring caves of cave x,y that potentially contain a wumpus"
(declare (variables ?a ?b))
(adj ?a ?b ?a2 ?b2)
(cave (x ?a2)(y ?b2)(has-wumpus ~FALSE))
)

(defrule wumpus-found
  (task think)
  (cave (x ?x)(y ?y)(stench TRUE))
  (hunter (agent ?a)(x ?x)(y ?y))
  (adj ?x ?y ?x2 ?y2)
  ?f <- (cave (x ?x2)(y ?y2)(has-wumpus MAYBE))
  (test (= 1 (count-query-results q-adjacent-maybe-wumpus ?x ?y)))
  =>
  (printout t "Wumpus detected in cave (" ?x2 "," ?y2 ") !!!" crlf)
  (modify ?f (has-wumpus TRUE) (safe FALSE))
  )

(defquery q-adjacent-maybe-pit
"returns an iterator of neighboring caves to cave x,y that may or may not hold a pit"
(declare (variables ?a ?b))
(adj ?a ?b ?a2 ?b2)
(cave (x ?a2)(y ?b2)(has-pit ~FALSE))
)

(defrule pit-found
  (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 q-adjacent-maybe-pit ?x ?y)))
  =>
  (printout t "Pit detected in cave (" ?x2 "," ?y2 ") !!!" crlf)
  (modify ?f (has-pit TRUE) (safe FALSE))
  )

;;=====End Change Task 1 =====
```

(ii). Execution traces:



```
THINKING...
Xena somewhat wants to go to (1,3).
No stench in (2,1) means no wumpus in (3,1).
A breeze in (2,1), so there may be a pit in (3,1).
Pit detected in cave (3,1) !!!
Seeing no glitter, Xena knows there is no gold in (2,1).
No stench in (2,1) means no wumpus in (2,2).
Xena somewhat wants to go to (2,2).
With neither wumpus nor pit, (2,2) is safe.
Xena strongly wants to go to (2,2).
PLANNING...
ACTING...
Xena goes to (2,2).
SIMULATING...
SENSING...
Xena sees no glitter.
Xena notices (2,3) nearby.
Xena notices (3,2) nearby.
Xena smells nothing.
Xena feels no breeze in (2,2).
```

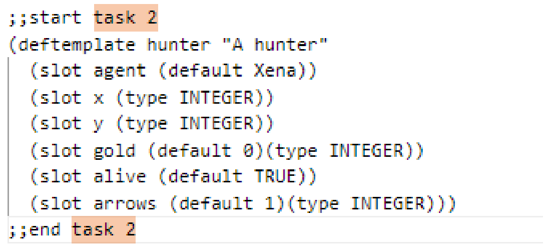


```
Xena slightly wants to go to (1,2).
Xena slightly wants to go to (1,1).
Pit detected in cave (3,3) !!!
Wumpus detected in cave (1,3) !!!
Seeing glitter, Xena knows there is gold in (2,3).
Xena slightly wants to go to (2,3).
Xena wants to pick up the gold in (2,3).
PLANNING...
```

**Snapshots from cave0*

- **<u>Task 2: New Top Level Goal -- Killing the Wumpus</u>**

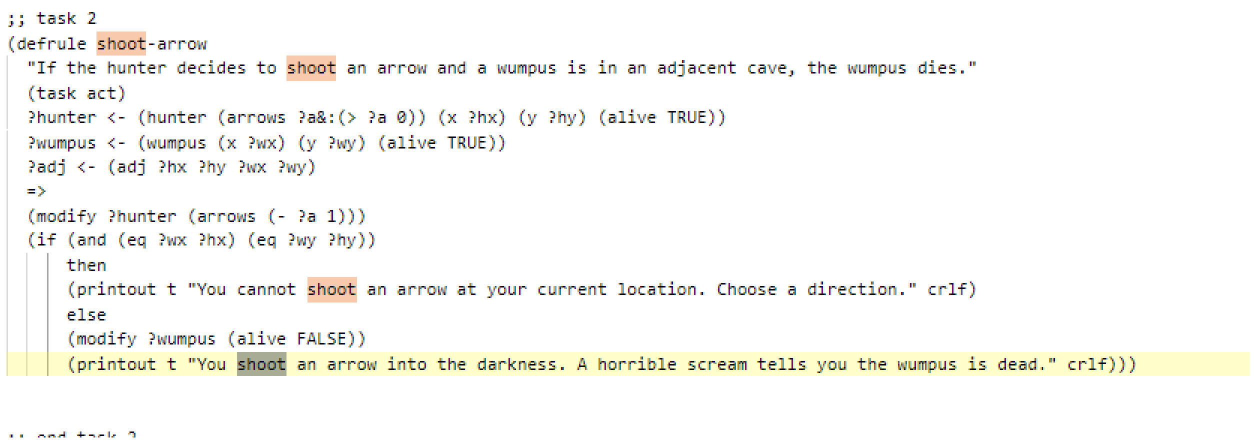
While the goal of the hunter was to only collect the gold and exit the cave, we will now add killing the Wumpus as one of the goals. To do this, we will start by updating the hunter's definition by including a slot for the arrows as well as the number of how many arrows the hunter has (here we suppose that the hunter only has one arrow and therefore can shoot only once).



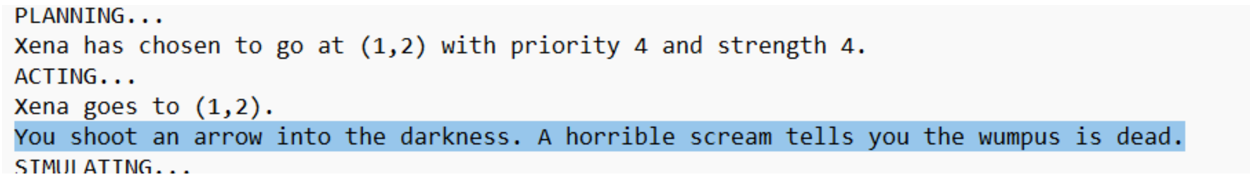
```
;;start task 2
(deftemplate hunter "A hunter"
  (slot agent (default Xena))
  (slot x (type INTEGER))
  (slot y (type INTEGER))
  (slot gold (default 0)(type INTEGER))
  (slot alive (default TRUE))
  (slot arrows (default 1)(type INTEGER)))
;;end task 2
```

- **Shoot-arrow rule:**

Allows the hunter to shoot an arrow at a Wumpus in an adjacent cave. If the action is made and the hunter shoots and there's a wumpus in the adjacent cave, then the Wumpus will die, and we will be hearing a scream. This also makes sure that the hunter has at least one arrow left to be able to do the action or else it points out that the hunter cannot shoot an arrow. After shooting, the arrows slot is decrement by one and the Wumpus' alive slot is changed from TRUE to FALSE, ensuring that it has been killed.

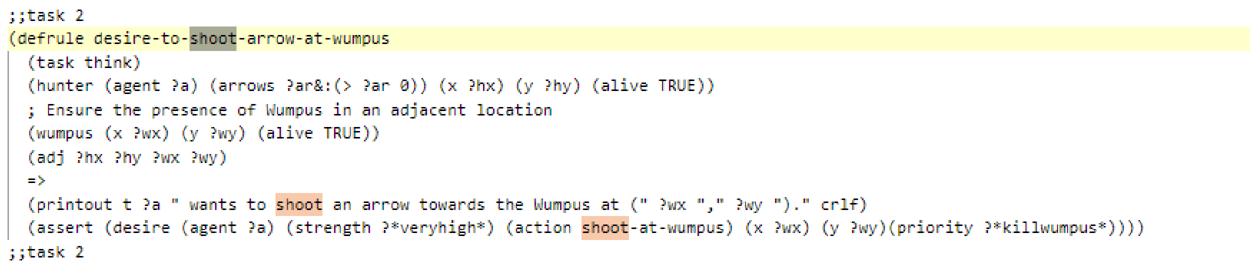


```
;; task 2
(defrule shoot-arrow
  "If the hunter decides to shoot an arrow and a wumpus is in an adjacent cave, the wumpus dies."
  (task act)
  ?hunter <- (hunter (arrows ?a&:(> ?a 0)) (x ?hx) (y ?hy) (alive TRUE))
  ?wumpus <- (wumpus (x ?wx) (y ?wy) (alive TRUE))
  ?adj <- (adj ?hx ?hy ?wx ?wy)
  =>
  (modify ?hunter (arrows (- ?a 1)))
  (if (and (eq ?wx ?hx) (eq ?wy ?hy))
      then
      (printout t "You cannot shoot an arrow at your current location. Choose a direction." crlf)
      else
      (modify ?wumpus (alive FALSE))
      (printout t "You shoot an arrow into the darkness. A horrible scream tells you the wumpus is dead." crlf)))

;; end task 2
```



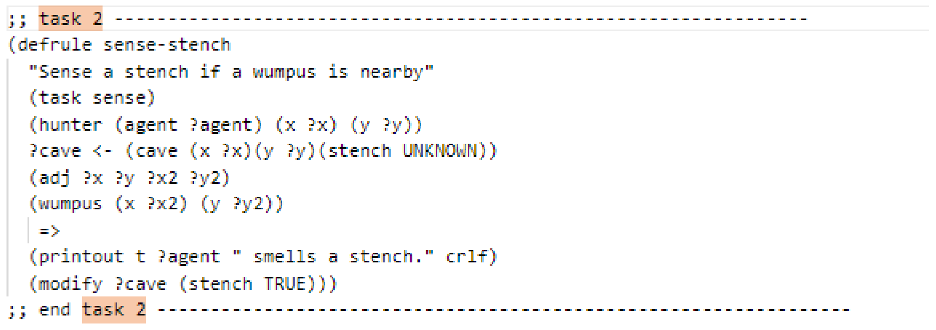
```
PLANNING...
Xena has chosen to go at (1,2) with priority 4 and strength 4.
ACTING...
Xena goes to (1,2).
You shoot an arrow into the darkness. A horrible scream tells you the wumpus is dead.
SIMULATING...
```

- **Desire-to-shoot-arrow-at-Wumpus rule:** On top of the shoot-arrow action, we will add a desire-to-shoot-arrow-at-Wumpus. This will make the hunter have a strong desire to attack the Wumpus when it's in an adjacent cave and there is at least one arrow left.



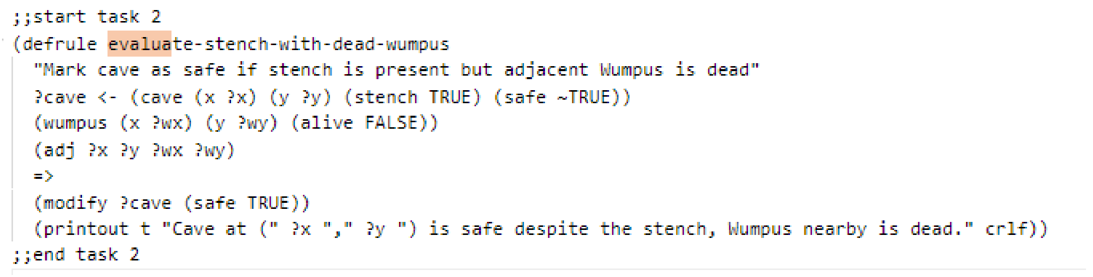
```
;;task 2
(defrule desire-to-shoot-arrow-at-wumpus
  (task think)
  (hunter (agent ?a) (arrows ?ar&:(> ?ar 0)) (x ?hx) (y ?hy) (alive TRUE))
  ; Ensure the presence of Wumpus in an adjacent location
  (wumpus (x ?wx) (y ?wy) (alive TRUE))
  (adj ?hx ?hy ?wx ?wy)
  =>
  (printout t ?a " wants to shoot an arrow towards the Wumpus at (" ?wx "," ?wy ")." crlf)
  (assert (desire (agent ?a) (strength ?*veryhigh*) (action shoot-at-wumpus) (x ?wx) (y ?wy)(priority ?*killwumpus*))))
;;task 2
```

- **Sense-stench rule:** While this rule used to be triggered only when the Wumpus is alive, it has been changed to detect stench whether the Wumpus is alive or not.



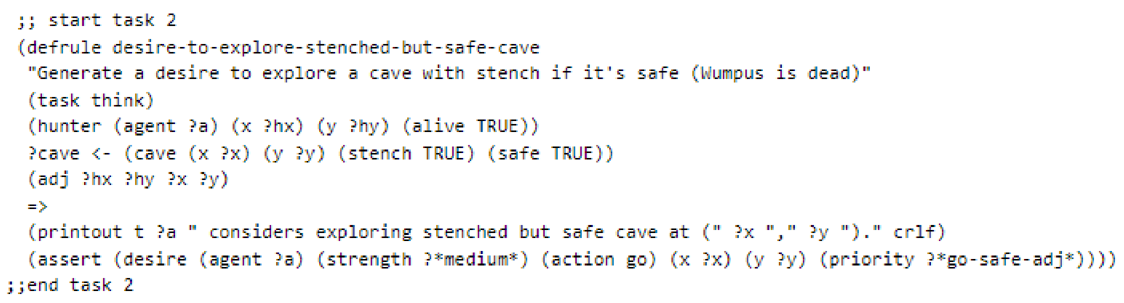
```
;; task 2 ------------------------------------------------------------------
(defrule sense-stench
  "Sense a stench if a wumpus is nearby"
  (task sense)
  (hunter (agent ?agent) (x ?x) (y ?y))
  ?cave <- (cave (x ?x)(y ?y)(stench UNKNOWN))
  (adj ?x ?y ?x2 ?y2)
  (wumpus (x ?x2) (y ?y2))
   =>
  (printout t ?agent " smells a stench." crlf)
  (modify ?cave (stench TRUE)))
;; end task 2 ------------------------------------------------------------------
```

- **Evaluate-stench-with-dead-Wumpus rule:** This rule is made to mark a cave close to the Wumpus as safe when the Wumpus has already been killed and this despite the stench still lingering. We check for the existence of a Wumpus that is adjacent to the cave with the stench and confirm that it is dead and that the cave is safe to explore.

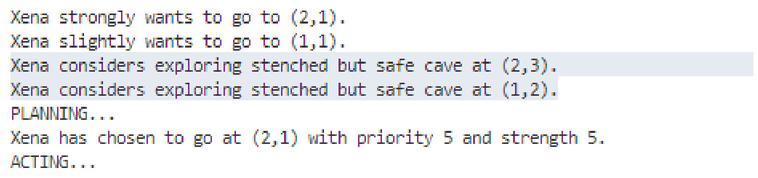


```
;;start task 2
(defrule evaluate-stench-with-dead-wumpus
  "Mark cave as safe if stench is present but adjacent Wumpus is dead"
  ?cave <- (cave (x ?x) (y ?y) (stench TRUE) (safe ~TRUE))
  (wumpus (x ?wx) (y ?wy) (alive FALSE))
  (adj ?x ?y ?wx ?wy)
  =>
  (modify ?cave (safe TRUE))
  (printout t "Cave at (" ?x "," ?y ") is safe despite the stench, Wumpus nearby is dead." crlf))
;;end task 2
```

- **Desire-to-explore-stenched-but-safe-care rule:** This rule is for caves that are adjacent to the Wumpus and therefore have a stench but are marked as safe since the Wumpus has been shot and is dead. Therefore, these caves are recognized as interesting places to explore despite the stench.

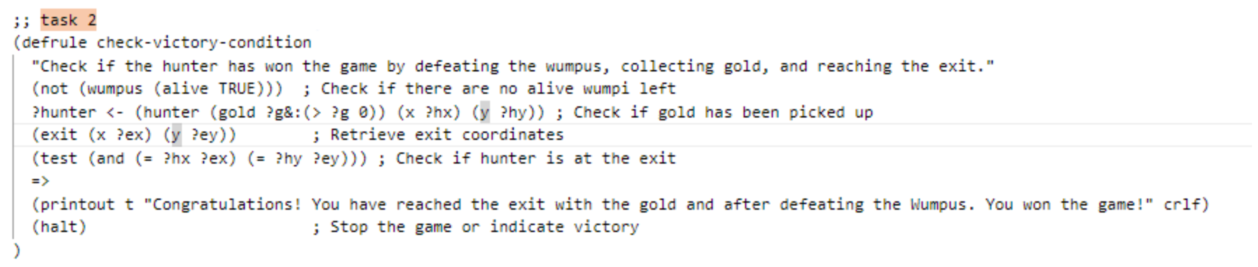


```
;; start task 2
(defrule desire-to-explore-stenched-but-safe-cave
 "Generate a desire to explore a cave with stench if it's safe (Wumpus is dead)"
 (task think)
 (hunter (agent ?a) (x ?hx) (y ?hy) (alive TRUE))
 ?cave <- (cave (x ?x) (y ?y) (stench TRUE) (safe TRUE))
 (adj ?hx ?hy ?x ?y)
 =>
 (printout t ?a " considers exploring stenched but safe cave at (" ?x "," ?y ")." crlf)
 (assert (desire (agent ?a) (strength ?*medium*) (action go) (x ?x) (y ?y) (priority ?*go-safe-adj*))))
;;end task 2
```

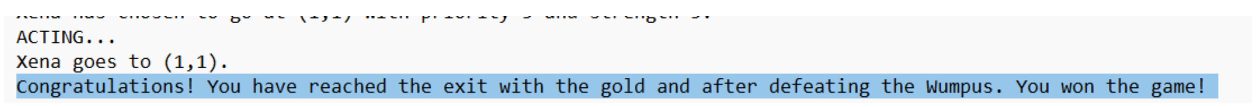


```
Xena strongly wants to go to (2,1).
Xena slightly wants to go to (1,1).
Xena considers exploring stenched but safe cave at (2,3).
Xena considers exploring stenched but safe cave at (1,2).
PLANNING...
Xena has chosen to go at (2,1) with priority 5 and strength 5.
ACTING...
```

- **Check-victory-condition rule:** This rule determines whether the hunter has achieved all the winning conditions (get gold, kill Wumpus and exit). We therefore check if the Wumpus is dead, if the hunter has picked the gold and if the hunter is at the exit position. If all 3 conditions are met, we print a congratulations message and stop the game.



```
;; task 2
(defrule check-victory-condition
  "Check if the hunter has won the game by defeating the wumpus, collecting gold, and reaching the exit."
  (not (wumpus (alive TRUE)))  ; Check if there are no alive wumpi left
  ?hunter <- (hunter (gold ?g&:(> ?g 0)) (x ?hx) (y ?hy)) ; Check if gold has been picked up
  (exit (x ?ex) (y ?ey))        ; Retrieve exit coordinates
  (test (and (= ?hx ?ex) (= ?hy ?ey))) ; Check if hunter is at the exit
  =>
  (printout t "Congratulations! You have reached the exit with the gold and after defeating the Wumpus. You won the game!" crlf)
  (halt)                        ; Stop the game or indicate victory
)
```



```
ACTING...
Xena goes to (1,1).
Congratulations! You have reached the exit with the gold and after defeating the Wumpus. You won the game!
```

- **Task 3: Improve the Hunters Ability to go to a Given Location**

The objective of this task is to enhance the Hunter's navigational proficiency within the cave3.jess environment. This improvement involves the addition of new rules prioritizing newly discovered cave locations and facilitating the investigation of more direct routes to the destination.

Observations and Desire Adjustments:

Through analysis of the existing rules (Task 3), the following observations were made regarding cave exploration desires:

- **“Safe” Caves:**
    - Adjacent safe caves elicit very high/high desires, emphasizing safety prioritization.
    - Distant safe caves have medium desires, representing a balanced approach.
    - Distant, already-visited safe caves have low desires, encouraging exploration of new areas.
- **“Risky/Dangerous” Caves:**
    - All risky caves (adjacent, distant, and visited) have progressively lower desires (low, very low, extremely low), reflecting a cautious approach towards potential hazards.

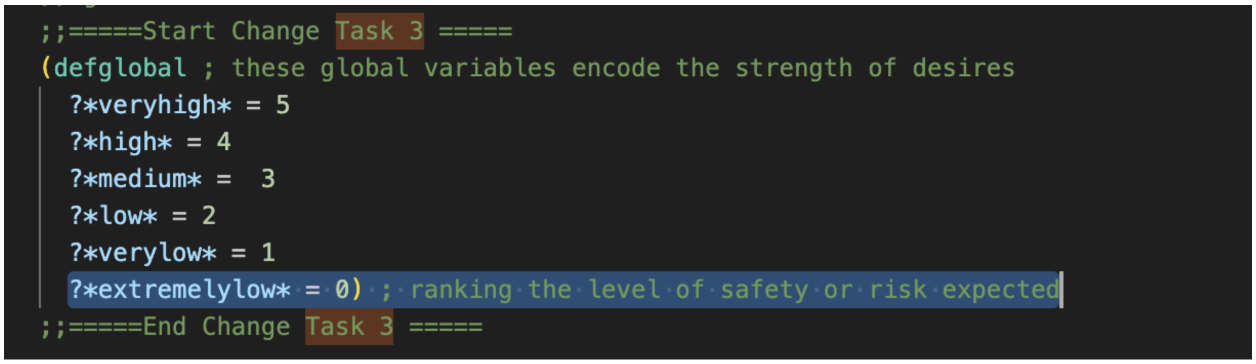
Desire Printouts:

The output reflects the different desire levels through descriptive phrases:

- Very high/High: "Strongly"
- Medium: "Moderately"
- Low: "Somewhat"
- Very low: "Slightly"
- Extremely low: "Barely"

(i). Code changes:

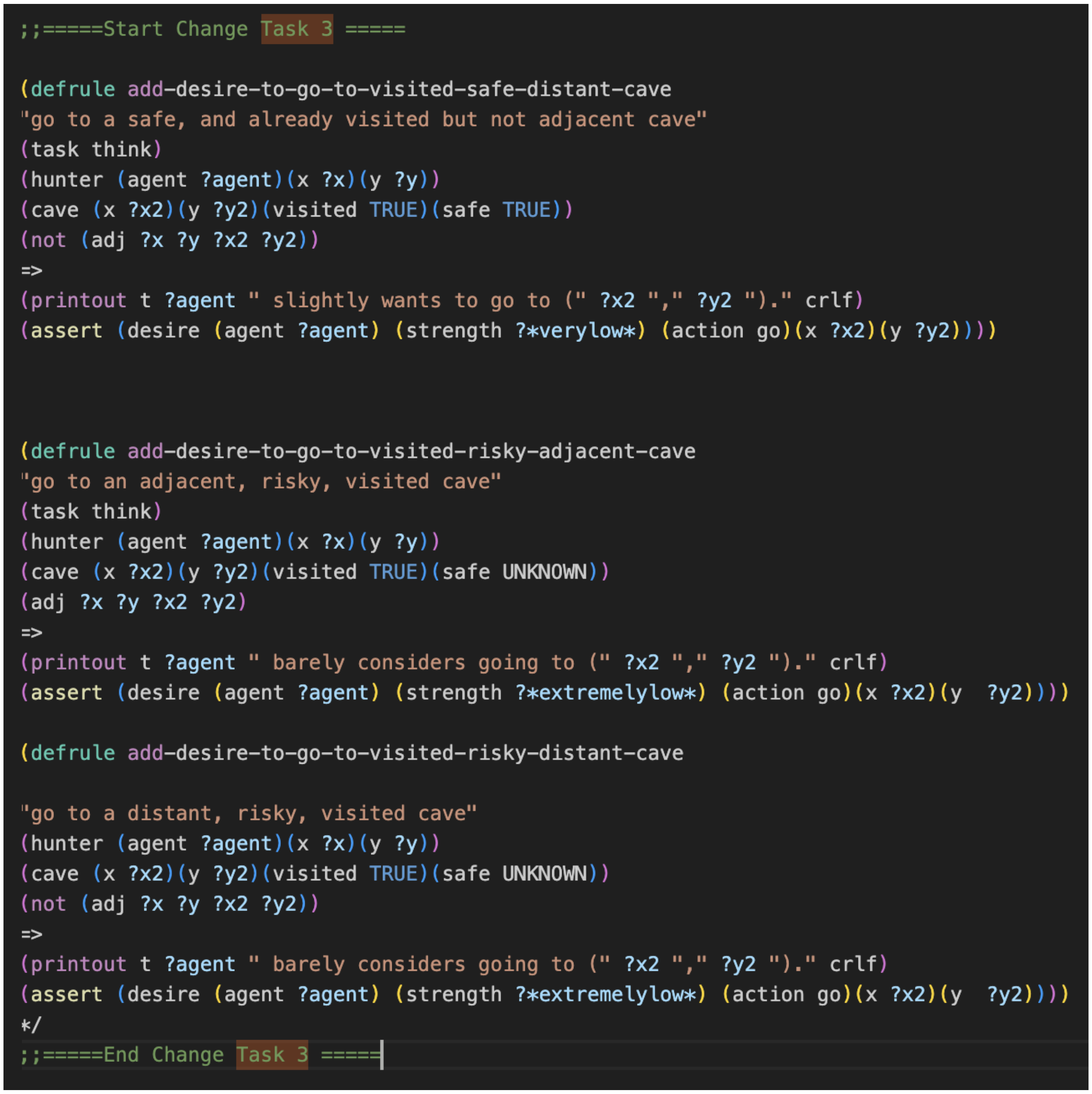
To start with, a new global variable named "extremelylow" has been implemented to refer to the lowest desire strength, assigned a value of 0.



```
;;=====Start Change Task 3 =====
(defglobal ; these global variables encode the strength of desires
  ?*veryhigh* = 5
  ?*high* = 4
  ?*medium* =  3
  ?*low* = 2
  ?*verylow* = 1
  ?*extremelylow* = 0) ; ranking the level of safety or risk expected
;;=====End Change Task 3 =====
```

Moreover, three new rules have been introduced to the code, focusing on incorporating desires related to navigating to different caves based on their safety and distance from the agent's current position. These desires are assigned different strengths, ranging from very low to extremely low, depending on the cave's safety level and distance.

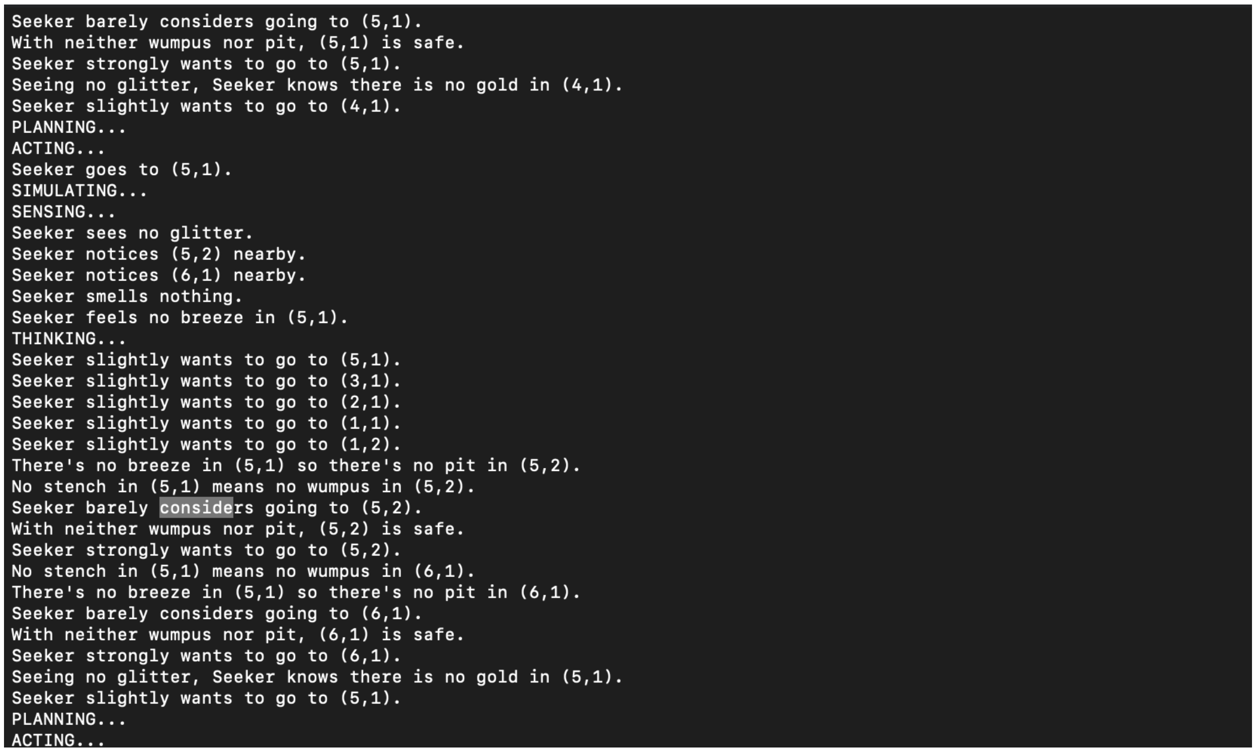
By employing desires with varying strengths, the agent can prioritize actions that are more likely to **achieve its goals**. For instance, if the goal is to find gold safely and efficiently, **stronger desires** (e.g., moving to a safe but distant cave) will be **promoted to goals** before **weaker desires** (e.g., moving to a risky but adjacent cave).



```
;;=====Start Change Task 3 =====

(defrule add-desire-to-go-to-visited-safe-distant-cave
"go to a safe, and already visited but not adjacent cave"
(task think)
(hunter (agent ?agent)(x ?x)(y ?y))
(cave (x ?x2)(y ?y2)(visited TRUE)(safe TRUE))
(not (adj ?x ?y ?x2 ?y2))
=>
(printout t ?agent " slightly wants to go to (" ?x2 "," ?y2 ")." crlf)
(assert (desire (agent ?agent) (strength ?*verylow*) (action go)(x ?x2)(y ?y2))))



(defrule add-desire-to-go-to-visited-risky-adjacent-cave
"go to an adjacent, risky, visited cave"
(task think)
(hunter (agent ?agent)(x ?x)(y ?y))
(cave (x ?x2)(y ?y2)(visited TRUE)(safe UNKNOWN))
(adj ?x ?y ?x2 ?y2)
=>
(printout t ?agent " barely considers going to (" ?x2 "," ?y2 ")." crlf)
(assert (desire (agent ?agent) (strength ?*extremelylow*) (action go)(x ?x2)(y  ?y2))))

(defrule add-desire-to-go-to-visited-risky-distant-cave

"go to a distant, risky, visited cave"
(hunter (agent ?agent)(x ?x)(y ?y))
(cave (x ?x2)(y ?y2)(visited TRUE)(safe UNKNOWN))
(not (adj ?x ?y ?x2 ?y2))
=>
(printout t ?agent " barely considers going to (" ?x2 "," ?y2 ")." crlf)
(assert (desire (agent ?agent) (strength ?*extremelylow*) (action go)(x ?x2)(y  ?y2))))
*/
;;=====End Change Task 3 =====
```

And therefore the strategical prioritizing of actions based on their alignment with the agent's goals, we optimize the exploration process, leading to both reduced exploration time and resource expenditure as well as a higher likelihood of achieving the agent's objectives.

(ii). Execution traces:



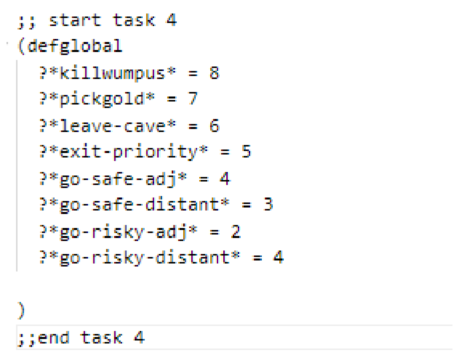
```
Seeker barely considers going to (5,1).
With neither wumpus nor pit, (5,1) is safe.
Seeker strongly wants to go to (5,1).
Seeing no glitter, Seeker knows there is no gold in (4,1).
Seeker slightly wants to go to (4,1).
PLANNING...
ACTING...
Seeker goes to (5,1).
SIMULATING...
SENSING...
Seeker sees no glitter.
Seeker notices (5,2) nearby.
Seeker notices (6,1) nearby.
Seeker smells nothing.
Seeker feels no breeze in (5,1).
THINKING...
Seeker slightly wants to go to (5,1).
Seeker slightly wants to go to (3,1).
Seeker slightly wants to go to (2,1).
Seeker slightly wants to go to (1,1).
Seeker slightly wants to go to (1,2).
There's no breeze in (5,1) so there's no pit in (5,2).
No stench in (5,1) means no wumpus in (5,2).
Seeker barely considers going to (5,2).
With neither wumpus nor pit, (5,2) is safe.
Seeker strongly wants to go to (5,2).
No stench in (5,1) means no wumpus in (6,1).
There's no breeze in (5,1) so there's no pit in (6,1).
Seeker barely considers going to (6,1).
With neither wumpus nor pit, (6,1) is safe.
Seeker strongly wants to go to (6,1).
Seeing no glitter, Seeker knows there is no gold in (5,1).
Seeker slightly wants to go to (5,1).
PLANNING...
ACTING...
```

**Snapshot from cave3*

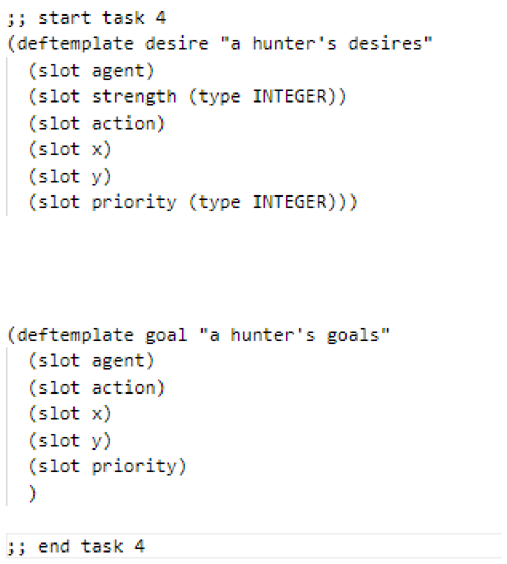
- **<u>Task 4: Allow the Hunter to choose the best action among all possible</u>**

For this task, we will conduct modifications to equip the hunter with the possibility to choose the best action possible when presented to different ones. To do that, we will make usage of ordering when it comes to the actions following a priority order.

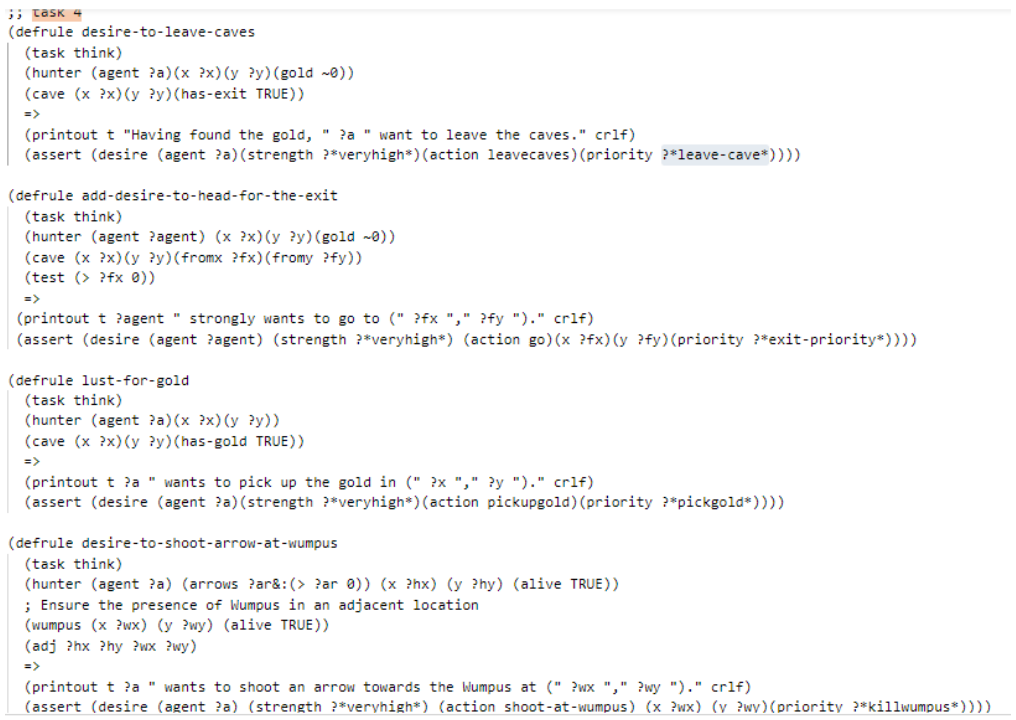
- **Definition of global variables for the prioritization:** We make usage here of these global variables to assign a numerical value to the different actions that the hunter will take to be able to organize them by order of priority. The highest priority is given to killing the Wumpus, followed by the one to pick the gold. This is due to the assumption that, if the Wumpus and the gold are in the same position, then the hunter will have to kill and eliminate the Wumpus first before picking the gold. The following priority is given to leaving the case and then comes the desire to find the exit, showcasing the goal of exiting with the treasure while doing it safely. After that, the priority is based on the safety and on the distance with the movement to safe adjacent caves being prioritized, then following movement to safe distant caves, then movement to risky adjacent caves and finally movement to risky distant caves.


```
;; start task 4
(defglobal
  ?*killwumpus* = 8
  ?*pickgold* = 7
  ?*leave-cave* = 6
  ?*exit-priority* = 5
  ?*go-safe-adj* = 4
  ?*go-safe-distant* = 3
  ?*go-risky-adj* = 2
  ?*go-risky-distant* = 4

)
;;end task 4
```

- **Modification of the desire and goal templates to support the prioritization:**


```
;; start task 4
(deftemplate desire "a hunter's desires"
  (slot agent)
  (slot strength (type INTEGER))
  (slot action)
  (slot x)
  (slot y)
  (slot priority (type INTEGER)))


(deftemplate goal "a hunter's goals"
  (slot agent)
  (slot action)
  (slot x)
  (slot y)
  (slot priority)
  )

;; end task 4
```

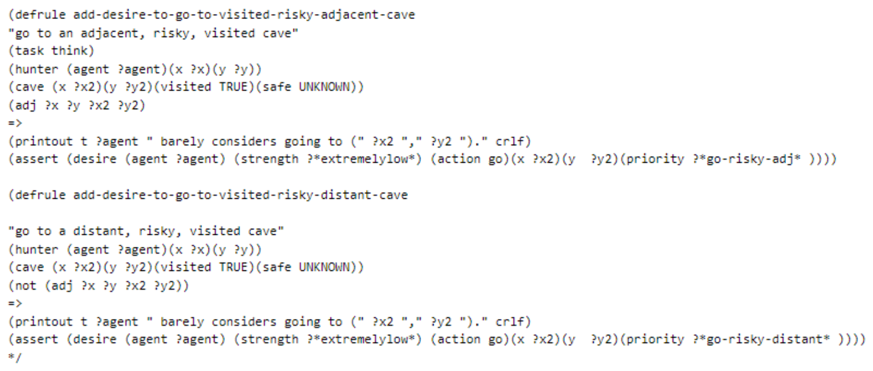
- **Inclusion of the prioritization in the setting of desires:**



```
;; task 4
(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)(priority ?*leave-cave*))))

(defrule add-desire-to-head-for-the-exit
  (task think)
  (hunter (agent ?agent) (x ?x)(y ?y)(gold ~0))
  (cave (x ?x)(y ?y)(fromx ?fx)(fromy ?fy))
  (test (> ?fx 0))
  =>
 (printout t ?agent " strongly wants to go to (" ?fx "," ?fy ")." crlf)
 (assert (desire (agent ?agent) (strength ?*veryhigh*) (action go)(x ?fx)(y ?fy)(priority ?*exit-priority*))))

(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 ?*veryhigh*)(action pickupgold)(priority ?*pickgold*))))

(defrule desire-to-shoot-arrow-at-wumpus
  (task think)
  (hunter (agent ?a) (arrows ?ar&:(> ?ar 0)) (x ?hx) (y ?hy) (alive TRUE))
  ; Ensure the presence of Wumpus in an adjacent location
  (wumpus (x ?wx) (y ?wy) (alive TRUE))
  (adj ?hx ?hy ?wx ?wy)
  =>
  (printout t ?a " wants to shoot an arrow towards the Wumpus at (" ?wx "," ?wy ")." crlf)
  (assert (desire (agent ?a) (strength ?*veryhigh*) (action shoot-at-wumpus) (x ?wx) (y ?wy)(priority ?*killwumpus*))))
```

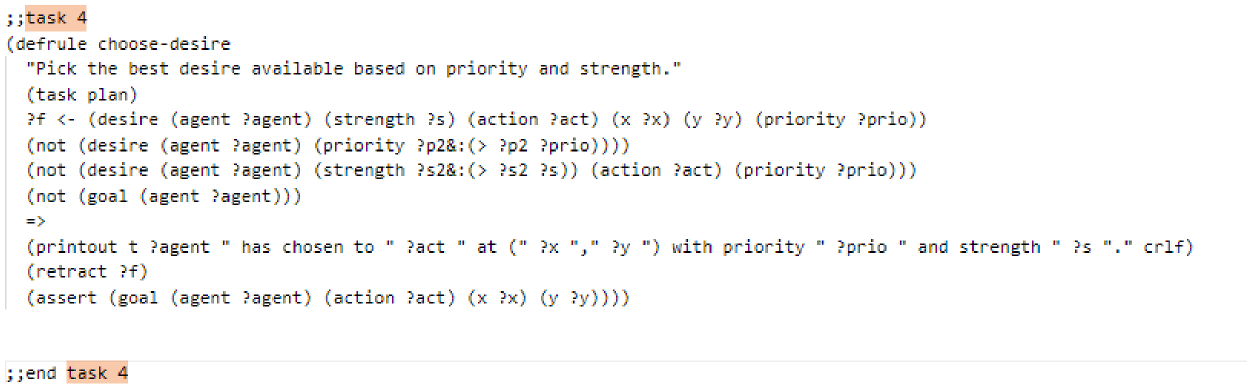
```
(defrule retract-lesser-desire
  "If we have two desires for the same thing, remove the one with lesser strength"
 (task think)
 (desire (agent ?agent)(strength ?s1)(action ?a)(x ?x)(y ?y))
 ?desire2 <- (desire (agent ?agent)(strength ?s2)(action ?a)(x ?x)(y ?y))
 (test (< ?s2 ?s1))
 =>
 (retract ?desire2))

(defrule add-desire-to-go-to-safe-adjacent-cave
 "go to an adjacent, safe, unvisited cave"
 (task think)
 (hunter (agent ?agent)(x ?x)(y ?y))
 (adj ?x ?y ?x2 ?y2)
 (cave (x ?x2)(y ?y2)(visited FALSE)(safe TRUE))
 =>
 (printout t ?agent " strongly wants to go to (" ?x2 "," ?y2 ")." crlf)
 (assert (desire (agent ?agent) (strength ?*high*) (action go)(x ?x2)(y ?y2)(priority ?*go-safe-adj*))))

(defrule add-desire-to-go-to-safe-distant-cave
 "go to a non-adjacent, safe, unvisited cave"
 (task think)
 (hunter (agent ?agent)(x ?x)(y ?y))
 (cave (x ?x2)(y ?y2)(visited FALSE)(safe TRUE))
 (not (adj ?x ?y ?x2 ?y2))
 =>
 (printout t ?agent " moderately wants to go to (" ?x2 "," ?y2 ")." crlf)
 (assert (desire (agent ?agent) (strength ?*medium*) (action go)(x ?x2)(y ?y2)(priority ?*go-safe-distant*))))

(defrule add-desire-to-go-to-risky-adjacent-cave
 "go to an adjacent, risky, unvisited cave"
 (task think)
```

```
 (defrule add-desire-to-go-to-risky-adjacent-cave
  "go to an adjacent, risky, unvisited cave"
  (task think)
  (hunter (agent ?agent)(x ?x)(y ?y))
  (cave (x ?x2)(y ?y2)(visited FALSE)(safe UNKNOWN))
  (adj ?x ?y ?x2 ?y2)
  =>
  (printout t ?agent " somewhat wants to go to (" ?x2 "," ?y2 ")." crlf)
  (assert (desire (agent ?agent) (strength ?*low*) (action go)(x ?x2)(y  ?y2)(priority ?*go-risky-adj*))))

 (defrule add-desire-to-go-to-risky-distant-cave
  "go to a distant, risky, unvisited cave"
  (task think)
  (hunter (agent ?agent)(x ?x)(y ?y))
  (cave (x ?x2)(y ?y2)(visited FALSE)(safe UNKNOWN))
  (not (adj ?x ?y ?x2 ?y2))
  =>
  (printout t ?agent " somewhat wants to go to (" ?x2 "," ?y2 ")." crlf)
  (assert (desire (agent ?agent) (strength ?*verylow*) (action go)(x ?x2)(y  ?y2)(priority ?*go-risky-distant*))))

  ;; task 3

 (defrule add-desire-to-go-to-visited-safe-distant-cave
 "go to a safe, and already visited but not adjacent cave"
 (task think)
 (hunter (agent ?agent)(x ?x)(y ?y))
 (cave (x ?x2)(y ?y2)(visited TRUE)(safe TRUE))
 (not (adj ?x ?y ?x2 ?y2))
 =>
 (printout t ?agent " slightly wants to go to (" ?x2 "," ?y2 ")." crlf)
 (assert (desire (agent ?agent) (strength ?*verylow*) (action go)(x ?x2)(y ?y2)(priority ?*go-safe-distant* ))))
```

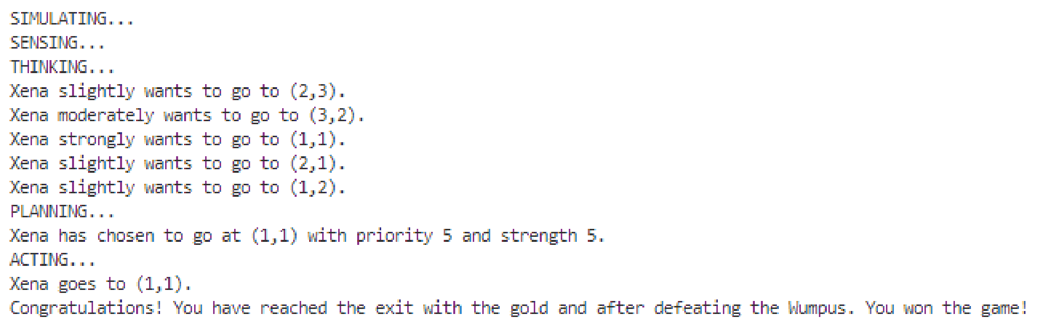


```
(defrule add-desire-to-go-to-visited-risky-adjacent-cave
"go to an adjacent, risky, visited cave"
(task think)
(hunter (agent ?agent)(x ?x)(y ?y))
(cave (x ?x2)(y ?y2)(visited TRUE)(safe UNKNOWN))
(adj ?x ?y ?x2 ?y2)
=>
(printout t ?agent " barely considers going to (" ?x2 "," ?y2 ")." crlf)
(assert (desire (agent ?agent) (strength ?*extremelylow*) (action go)(x ?x2)(y  ?y2)(priority ?*go-risky-adj* ))))

(defrule add-desire-to-go-to-visited-risky-distant-cave

"go to a distant, risky, visited cave"
(hunter (agent ?agent)(x ?x)(y ?y))
(cave (x ?x2)(y ?y2)(visited TRUE)(safe UNKNOWN))
(not (adj ?x ?y ?x2 ?y2))
=>
(printout t ?agent " barely considers going to (" ?x2 "," ?y2 ")." crlf)
(assert (desire (agent ?agent) (strength ?*extremelylow*) (action go)(x ?x2)(y  ?y2)(priority ?*go-risky-distant* ))))
*/
```

- **Update of the choose-desire function to consider strength and priority: In the planning phase, this rule evaluates all possible desires the hunter will encounter and therefore an action being considered.** We then select the action with the highest priority and among these actions, we select the one with the highest strength while making sure that we focus on one objective at a time. The outcome is an action plan with the goal dictating the hunter's next action.



```
;;task 4
(defrule choose-desire
  "Pick the best desire available based on priority and strength."
  (task plan)
  ?f <- (desire (agent ?agent) (strength ?s) (action ?act) (x ?x) (y ?y) (priority ?prio))
  (not (desire (agent ?agent) (priority ?p2&:(> ?p2 ?prio))))
  (not (desire (agent ?agent) (strength ?s2&:(> ?s2 ?s)) (action ?act) (priority ?prio)))
  (not (goal (agent ?agent)))
  =>
  (printout t ?agent " has chosen to " ?act " at (" ?x "," ?y ") with priority " ?prio " and strength " ?s "." crlf)
  (retract ?f)
  (assert (goal (agent ?agent) (action ?act) (x ?x) (y ?y))))

;;end task 4
```



```
SIMULATING...
SENSING...
THINKING...
Xena slightly wants to go to (2,3).
Xena moderately wants to go to (3,2).
Xena strongly wants to go to (1,1).
Xena slightly wants to go to (2,1).
Xena slightly wants to go to (1,2).
PLANNING...
Xena has chosen to go at (1,1) with priority 5 and strength 5.
ACTING...
Xena goes to (1,1).
Congratulations! You have reached the exit with the gold and after defeating the Wumpus. You won the game!
```

This selection is illustrated here. The hunter has the choice between different locations it can go to with different degrees of priority and strength. In the end, the hunter will opt for a movement to (1,1) as it has the highest priority and strength. Here therefore, the hunter will leave the cave.

<u>IV – Conclusion</u>

The Wumpus World Game is an example of knowledge-based agents in action, demonstrating how we can implement reasoning, decision-making and planning to solve specific problems. This project provides insights into knowledge-based agents and rule-based systems and how we can encode the actions and decisions while showcasing how challenging it is at the same time. This showcases the progress that is being made when it comes to creating autonomous agents capable of reasoning and navigating the real world.