Faculté sciences & sciences de l'ingénieur
Université Bretagne Sud

Projet : Agent logique pour la chasse au Wumpus

Travail présenté à Salim Lardjane
INF1604

réalisé par
Jean Sevaux, Aude Le Boutouiller,
Rafael Massampu Evora Tavares

25 Avril 2021

Table des matières

1 Introduction

Le jeu «`Hunt the Wumpus`», est sorti initialement en 1972. Le but de ce dernier est de guider le chasseur vers le trésor puis vers la case départ, tout en faisant attention aux obstacles, dont le Wumpus, mais aussi des puits. Il est possible d'éliminer le monstre, même s'il est préférable de l'esquiver, par contre, les puits ne peuvent être détruits ou bouchés, ils seront obligatoirement contournés par le chasseur.

2 Rappel de la problématique

Dans le cadre de l'enseignement de Logique en ce sixième semestre de Licence, il nous est proposé un projet, qui consiste à programmer le jeu «`Hunt the Wumpus`», permettant de mettre en pratique nos connaissances sur la logique des propositions (LP0).

Le but du projet est de construire la base de connaissance de l'agent ainsi que l'implémentation de ses routines de recherches. Il s'agit donc de la réalisation d'une intelligence artificielle, programmée en Python, qui créer un agent capable de résoudre le jeu de «`La chasse au Wumpus`».

Avec cela, le but du projet n'est pas de coder seulement le jeu mais aussi un programme, une intelligence artificielle donc, capable de résoudre toute seule le jeu. Le jeu doit être codé en Python, avec une carte du jeu pour qu'on soit capable de se repérer pendant le jeu et que le programme n'ait aucune aide de l'humain pour résoudre. Donc avec tout cela en tête nous avons apporté la solution décrite ci-dessous.

3 Mise en place et configuration du projet

- CryptoMiniSat

  Parmi les membres de cette équipe, deux possèdent des ordinateurs sous Windows 10, le 3ème un ordinateur sous Debian. Bien que le solveur SAT CryptoMiniSat peut être compilé sous Windows, nous avons tous décidé de l'installer sous une configuration Unix/Linux. D'une part pour faciliter les échanges et le travail d'équipe en partant tous de la même base, d'autre part puisque nous avons l'habitude de programmer et de compiler sous cette distribution.

  Cela nous a été possible, grâce à un sous-système Ubuntu pour Windows (WSL). Avec nos configurations Linux opérationnelles, nous n'avions plus qu'à suivre les instructions de compilation disponibles dans le README présent sur le GitHub du projet CryptoMiniSat.

- Le jeu et les tests

La deuxième partie de la mise en place portait sur l'installation du projet. Pour cela, nous avons téléchargé l'archive disponible sur le site web de présentation du projet.

Afin de vérifier que notre projet était opérationnel, c'est-à-dire que le solveur CryptoMiniSat et le jeu avaient bien été installés et configurés, une série de tests y est intégrée.

Le lancement de ces tests se fait depuis le dossier racine du projet en lançant la commande `python3.9 wumpus.py -t`.

L'installation fut correcte dès que nous avons obtenu ce résultat :

```
Running simple CryptoMiniSat test:
-------------------------------------------------------
Test 1
  Query:      '(P | ~P)'
  Query CNF:  [(P | ~P)]
  Result:     True   (Expected: True)
  Variable Assignment: {P: False}
-------------------------------------------------------
Test 2
  Query:      '(P & ~P)'
  Query CNF:  [P, ~P]
  Result:     False   (Expected: False)
-------------------------------------------------------
Test 3
  Query:      '(P | R) <=> (~(Q | R) & (R >> ~(S <=> T)))'
  Query CNF:  [(S | ~T | P | R | Q | R), (T | ~S | P | R | Q | R), (R | P | R | Q | R), (~P | ~Q), (~R | ~Q), (~P | ~R), (~R | ~R), (~P | ~T | ~S | ~R),
(~R | ~T | ~S | ~R), (~P | S | ~S | ~R), (~R | S | ~S | ~R), (~P | ~T | T | ~R), (~R | ~T | T | ~R), (~P | S | T | ~R), (~R | S | T | ~R)]
  Result:     True   (Expected: True)
  Variable Assignment: {P: False, Q: True, R: False, S: False, T: False}
-------------------------------------------------------
Successfully passed 3 tests.
DONE.
```

L'avant dernière phrase nous indique que les 3 tests ont été réalisés avec succès, nous avons donc bien installé le tout, nous pouvons maintenant commencer à implémenter les fonctions de ce projet.

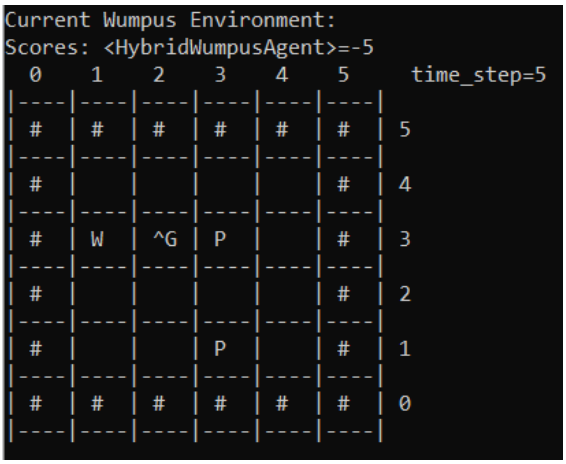
Un autre test est aussi possible, il est accessible par la commande `python3.9 wumpus.py -k`. Il permet à l'utilisateur d'avoir la main sur les mouvements du chasseur et de choisir sa stratégie ainsi que ses directions. Cette commande nous a également permis d'appréhender ce projet, comprendre son fonctionnement, les stratégies à adopter.

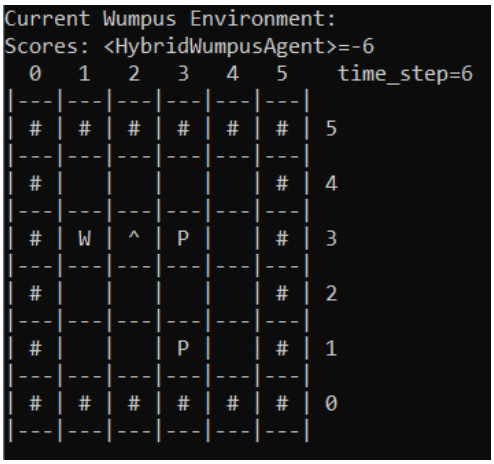
À chaque déplacement, le jeu nous indique les perceptions du chasseur, c'est alors à nous de décider de la direction qu'il doit prendre, toujours dans le but de récupérer le trésor sans croiser le chemin du Wumpus.

```
[0] You perceive: ['~Stench', '~Breeze', '~Glitter', '~Bump', '~Scream']
Enter Action ('?' for list of commands): ?
```

```
   The following are valid Hunt The Wumpus actions:
     'TurnRight', 'TurnLeft', 'Forward', 'Grab', 'Climb', 'Shoot', 'Wait'
```

Un fois le chasseur arrivé sur la case du trésor (image de gauche), il faut qu'il le saisisse avec l'instruction `"GRAB"` (image de droite) :





Mais le jeu ne s'arrête pas là, il faut que le chasseur revienne à son point de départ afin de remonter avec son trésor, sans oublier le Wumpus qu'il faut éviter, soit en le contournant, soit en le tuant. Une fois que le chasseur a réalisé l'instruction `"Climb"` depuis la case de départ, il a gagné contre le Wumpus, le jeu s'arrête et son score s'affiche :

```
Final Scores: <HybridWumpusAgent>=978
```

Étant donné que les déplacements, le tir de flèche pour tuer le Wumpus, le fait de tomber dans un puits ou de se faire manger par le monstre coûte des points, et que la victoire du chasseur en partant avec le trésor rapporte des points, le but de ce jeu est d'obtenir le plus grand nombre de points.

4 Développement du projet

Lorsque les fonctions sont bien implémentées, il nous est possible de tester la version "automatisée" de cette chasse grâce à la commande `python3.9 wumpus.py -k`. Voici les résultats que nous obtenons avec notre code :

```
Scores: <HybridWumpusAgent>=0
  0   1   2   3   4   5    time_step=0
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | W | G | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # | ^ |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

```
Ask if Wumpus is Alive:
  Is Wumpus Alive? : True
```

Nous pouvons apercevoir qu'au lancement du jeu, notre chasseur se situe toujours dans la case (1,1), il est représenté par le symbole "^", et que le Wumpus est bien en vie.

```
<HybridWumpusAgent> perceives ['Stench', '~Breeze', '~Glitter', '~Bump', '~Scream']
```

Étant donné qu'à partir de cette case, les indicateurs de perceptions sont tous précédés d'un ' ', il n'y a donc aucun danger dans les cases voisines, le chasseur étend alors sa connaissance du plateau de jeu et peut sauvegarder ces informations de la manière suivante :

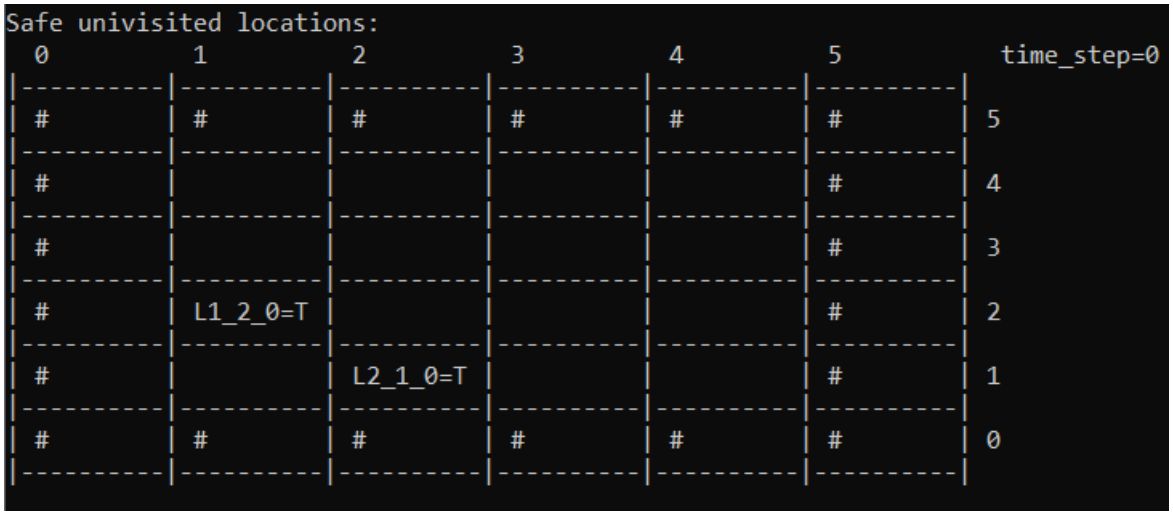
```
Find OK locations queries
  0             1             2             3             4             5             time_step=0

|-------------|-------------|-------------|-------------|-------------|-------------|
| #           | #           | #           | #           | #           | #           | 5
|-------------|-------------|-------------|-------------|-------------|-------------|
| #           | OK1_4_0=?   | OK2_4_0=?   | OK3_4_0=?   | OK4_4_0=?   | #           | 4
|-------------|-------------|-------------|-------------|-------------|-------------|
| #           | OK1_3_0=?   | OK2_3_0=?   | OK3_3_0=?   | OK4_3_0=?   | #           | 3
|-------------|-------------|-------------|-------------|-------------|-------------|
| #           | OK1_2_0=True| OK2_2_0=?   | OK3_2_0=?   | OK4_2_0=?   | #           | 2
|-------------|-------------|-------------|-------------|-------------|-------------|
| #           | OK1_1_0=True| OK2_1_0=True| OK3_1_0=?   | OK4_1_0=?   | #           | 1
|-------------|-------------|-------------|-------------|-------------|-------------|
| #           | #           | #           | #           | #           | #           | 0
|-------------|-------------|-------------|-------------|-------------|-------------|
```

Dans ce tableau, le jeu enregistre les cases qui sont sans danger grâce à la valeur "TRUE" qui leur est attribuée. C'est lors de l'initialisation des cases à "TRUE" que la logique intervient. En effet, si le chasseur est sur une case sans aucune perception, cela signifie qu'aucune des cases voisines ne comportent un danger, elles peuvent donc être explorées librement.

Un autre tableau sert à évoluer dans le jeu, celui-ci contient également les cases qui n'ont pas de danger mais qui n'ont pas été visitées non plus. Cette base de connaissance est utile au chasseur pour qu'il soit efficace dans sa recherche.

En effet, chaque déplacement coûte 1 point, et l'objectif étant de finir le jeu avec le plus

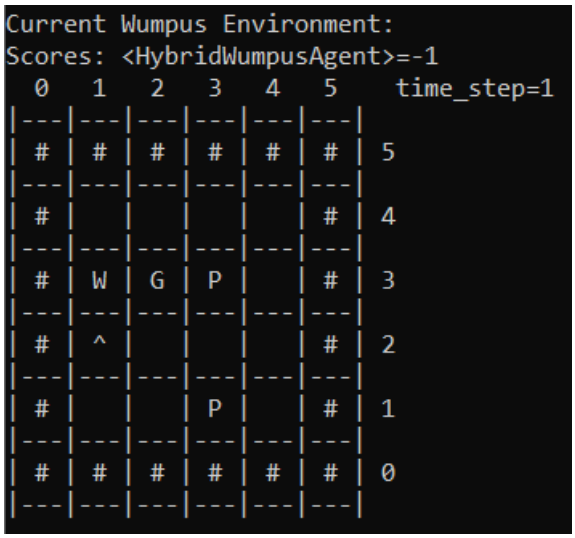
de points possibles, il est important de les optimiser et d'aller explorer des cases nouvelles qui nous apporteront de nouvelles connaissances.



À partir de cette case et de ses connaissances, le chasseur décide d'aller tout droit :

```
<HybridWumpusAgent> perceives ['~Stench', '~Breeze', '~Glitter', '~Bump', '~Scream'] and does Forward
```

Nous le retrouvons donc dans la case supérieure, puisqu'il était dirigé vers le Nord.



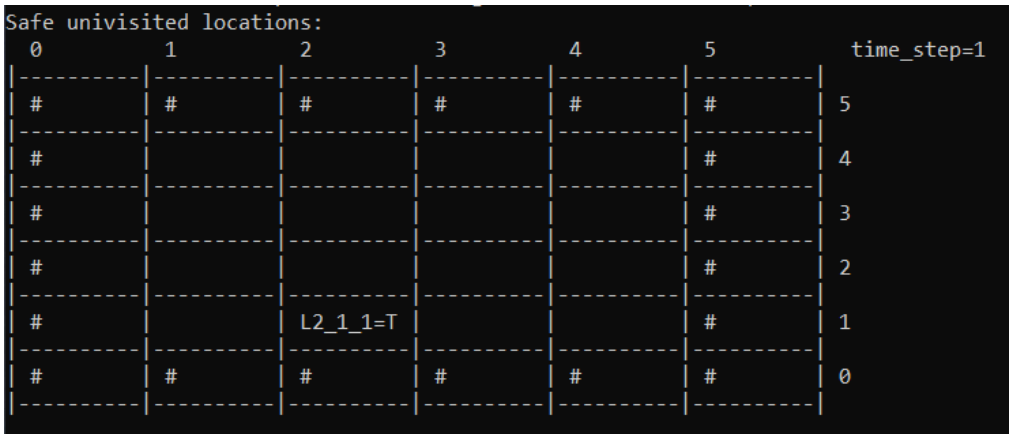
Depuis cette case, le chasseur détecte une odeur nauséabonde, représentée dans le jeu par la valeur `"Stench"` non précédée d'un `"~"` :

```
<HybridWumpusAgent> perceives ['Stench', '~Breeze', '~Glitter', '~Bump', '~Scream']
```

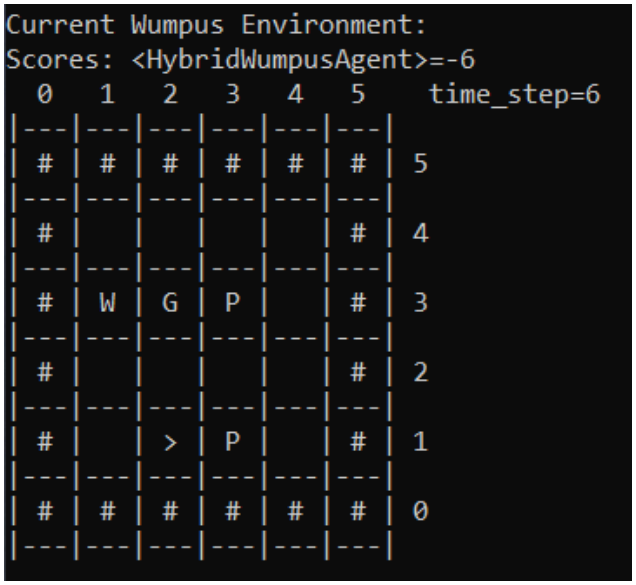
Il y a donc un danger dans au moins une des cases voisines, mais avec les éléments en sa possession le chasseur n'est pas en mesure de déduire la ou les cases concernées.

Étant donné son manque de connaissances sur la dangerosité des cases voisines et sa

volonté de gagner, le chasseur décide de revenir sur ses pas pour parcourir la case (2,1), voisine de sa case de départ et dont il sait qu'elle ne porte pas de danger et qu'il ne l'a pas visité comme son tableau l'indique :



On remarque bien que ce tableau a évolué depuis tout à l'heure, la case (1,2) sur laquelle il se situe actuellement n'y est plus renseignée puisqu'elle a été visitée. Nous retrouvons notre chasseur quelques déplacements plus tard, il est arrivé sur la case (2,1), seule case qu'il savait sans danger.



Sur cette case, il a de nouvelles sensations qui l'amènent à faire de nouvelles déductions :

En effet, l'exploration de cette nouvelle case est intéressante. Dans celle-ci, il ressent de la brise, et parmi les trois qui la jouxtent on sait déjà que celle de gauche (1,1) est sans danger. L'on connaît aussi les perceptions de la case (1,2) qui nous indiquent que le Wumpus se situe soit case (1,3) soit case (2,2). Or, dans la case courante, le chasseur ne ressent que de la brise,

si le Wumpus se situait à côté, c'est-à-dire dans la case (2,2) de la puanteur se serait faite ressentir également. Le chasseur en déduit donc que le wumpus ne peut se situer que dans la case (1,3). Une autre déduction est aussi possible, de la brise se fait ressentir dans la case courante, puisque la case de gauche (1,1) est déjà connue pour être sans danger, le puit ne peut se situer qu'au-dessus (2,2) ou à droite (3,1) de la case courante. De la même manière que précédemment, le chasseur comprend que le puit ne peut pas se situer au-dessus de lui, puisque la brise ne faisait pas partie des perceptions de la case (1,2), pourtant voisine avec cette case. Il obtient donc une nouvelle information, un puit se trouve en case (3,1). Ainsi, il en arrive à trouver une nouvelle case de sûreté, la (2,2). Il peut donc compléter en partie son tableau, en donnant aux cases comportant un danger, la valeur "`False`", et à la nouvelle case sans danger, la valeur "`True`".

```
step=6
|---------------|---------------|---------------|---------------|---------------|---------------|
| #             | #             | #             | #             | #             | #             | 5
|---------------|---------------|---------------|---------------|---------------|---------------|
| #             | OK1_4_6=?     | OK2_4_6=?     | OK3_4_6=?     | OK4_4_6=?     | #             | 4
|---------------|---------------|---------------|---------------|---------------|---------------|
| #             | OK1_3_6=False | OK2_3_6=?     | OK3_3_6=?     | OK4_3_6=?     | #             | 3
|---------------|---------------|---------------|---------------|---------------|---------------|
| #             | OK1_2_6=True  | OK2_2_6=True  | OK3_2_6=?     | OK4_2_6=?     | #             | 2
|---------------|---------------|---------------|---------------|---------------|---------------|
| #             | OK1_1_6=True  | OK2_1_6=True  | OK3_1_6=False | OK4_1_6=?     | #             | 1
|---------------|---------------|---------------|---------------|---------------|---------------|
| #             | #             | #             | #             | #             | #             | 0
|---------------|---------------|---------------|---------------|---------------|---------------|
```

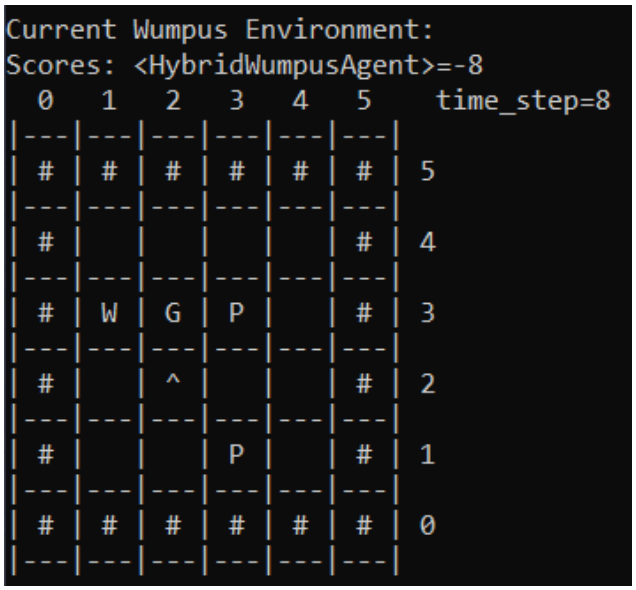
Il y a également du changement dans son tableau de lieux sains n'ayant pas été visités :

```
Safe univisited locations:
  0          1          2          3          4          5           time_step=6
|----------|----------|----------|----------|----------|----------|
| #        | #        | #        | #        | #        | #        | 5
|----------|----------|----------|----------|----------|----------|
| #        |          |          |          |          | #        | 4
|----------|----------|----------|----------|----------|----------|
| #        |          |          |          |          | #        | 3
|----------|----------|----------|----------|----------|----------|
| #        |          | L2_2_6=T |          |          | #        | 2
|----------|----------|----------|----------|----------|----------|
| #        |          |          |          |          | #        | 1
|----------|----------|----------|----------|----------|----------|
| #        | #        | #        | #        | #        | #        | 0
|----------|----------|----------|----------|----------|----------|
```

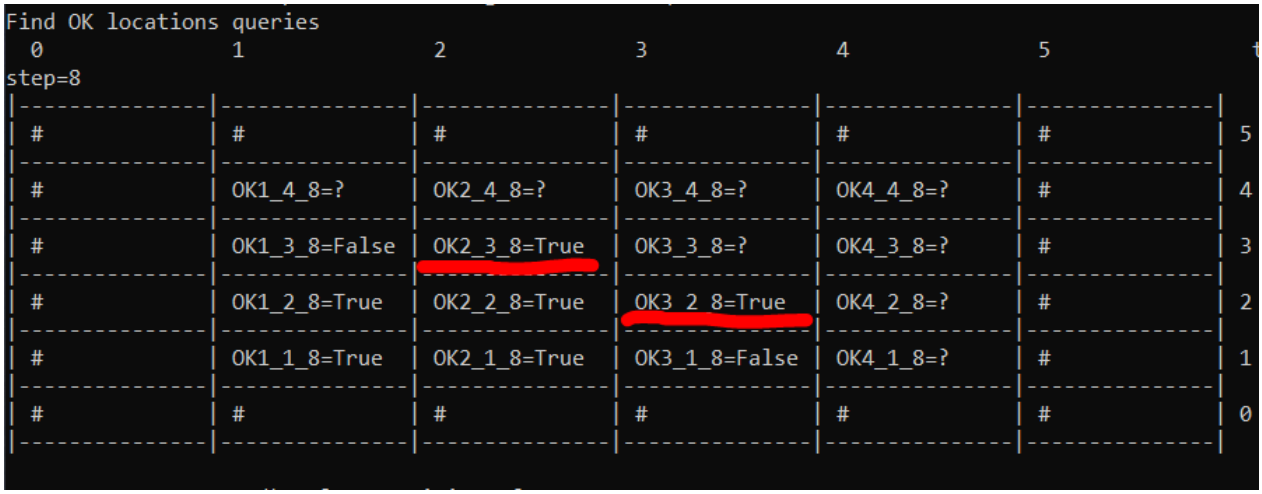
Puisque la case au-dessus de lui est sans danger et qu'il est orienté vers l'Ouest, notre chasseur décide de se tourner sur sa gauche, afin de pouvoir avancer par la suite vers la case sans danger.

```
<HybridWumpusAgent> perceives ['~Stench', 'Breeze', '~Glitter', '~Bump', '~Scream'] and does TurnLeft
```

Re-voici notre chasseur ayant avancé d'un case.



Ici, il n'a aucune sensation, il sait donc maintenant que les cases (2,3) et (3,2), les cases voisines, ne présentent aucun danger. Il met ainsi ses tableaux à jour, celui qui renseigne le niveau de chaque case une fois connu, et celui qui lui indique les cases qu'il n'a pas encore visité mais qu'il peut le faire sans soucis.



```
Safe univisited locations:
  0         1         2         3         4         5          time_step=8
|---------|---------|---------|---------|---------|---------|
| #       | #       | #       | #       | #       | #       | 5
|---------|---------|---------|---------|---------|---------|
| #       |         |         |         |         | #       | 4
|---------|---------|---------|---------|---------|---------|
| #       |         | L2_3_8=T |         |         | #       | 3
|---------|---------|---------|---------|---------|---------|
| #       |         |         | L3_2_8=T |         | #       | 2
|---------|---------|---------|---------|---------|---------|
| #       |         |         |         |         | #       | 1
|---------|---------|---------|---------|---------|---------|
| #       | #       | #       | #       | #       | #       | 0
|---------|---------|---------|---------|---------|---------|
```

À partir de là, il a 2 possibilités, il peut aller dans la case du haut ou bien celle de gauche. Mais étant donné qu'une rotation coûte un point, il est plus judicieux d'aller visiter la case du dessus, car cela ne nécessite pas que le chasseur se tourne, il va donc tout droit.

```
<HybridWumpusAgent> perceives ['~Stench', '~Breeze', '~Glitter', '~Bump', '~Scream'] and does Forward
```

Il se retrouve maintenant sur la case (2,3) et les nouvelles sont bonnes. En effet notre chasseur ressent les paillettes qui lui signifie qu'il est sur la case du trésor !

```
Current Wumpus Environment:
Scores: <HybridWumpusAgent>=-9
  0    1    2    3    4    5     time_step=9
|----|----|----|----|----|----|
| #  | #  | #  | #  | #  | #  | 5
|----|----|----|----|----|----|
| #  |    |    |    |    | #  | 4
|----|----|----|----|----|----|
| #  | W  | ^G | P  |    | #  | 3
|----|----|----|----|----|----|
| #  |    |    |    |    | #  | 2
|----|----|----|----|----|----|
| #  |    |    | P  |    | #  | 1
|----|----|----|----|----|----|
| #  | #  | #  | #  | #  | #  | 0
|----|----|----|----|----|----|
```

```
<HybridWumpusAgent> perceives ['Stench', 'Breeze', 'Glitter', '~Bump', '~Scream']
```

Il ressent également une puanteur, mais il sait déjà que le Wumpus se trouve sur sa gauche, ainsi qu'une brise, mais elle ne va pas lui poser de soucis puisqu'il sait qu'il peut revenir sur ses pas pour rentrer. Car, il ne faut pas l'oublier, le chasseur doit revenir à son point de départ et sortir de la grotte.

Avant cela, il n'oublie pas de saisir son butin, avec l'action "`Grab`", ni de renseigner dans son tableau que sa case courante est sans danger.

```
<HybridWumpusAgent> perceives ['Stench', 'Breeze', 'Glitter', '~Bump', '~Scream'] and does Grab
```

Le chasseur sait maintenant qu'il a le trésor et qu'il doit revenir à son point de départ, toujours en restant vigilant à ne pas tomber dans un puits ou se faire manger par le monstre. Il établit alors son plan de retour, le faisant passer par les cases qu'il sait sûres.

```
HWA.agent_program(): Plan:
 ['TurnLeft', 'TurnLeft', 'Forward', 'Forward', 'TurnRight', 'Forward', 'Climb']
```

Il effectue une à une les étapes de son plan et revient intacte à son point de départ, cette fois-ci avec le trésor.

```
Current Wumpus Environment:
Scores: <HybridWumpusAgent>=-16
  0   1   2   3   4   5    time_step=16
|---|---|---|---|---|---|
| # | # | # | # | # | # | 5
|---|---|---|---|---|---|
| # |   |   |   |   | # | 4
|---|---|---|---|---|---|
| # | W |   | P |   | # | 3
|---|---|---|---|---|---|
| # |   |   |   |   | # | 2
|---|---|---|---|---|---|
| # | < |   | P |   | # | 1
|---|---|---|---|---|---|
| # | # | # | # | # | # | 0
|---|---|---|---|---|---|
```

Il n'a plus qu'à sortir de la grotte avec l'instruction "`Climb`" et le voilà vainqueur du jeu !

```
<HybridWumpusAgent> perceives ['~Stench', '~Breeze', '~Glitter', '~Bump', '~Scream'] and does Climb
```

Il peut consulter son score, et constater que les 17 actions qu'il a effectué lui ont coûté

1 point chacune et que sa sortie victorieuse de la grotte avec le trésor lui a rapporté 1 000 points.

```
Final Scores: <HybridWumpusAgent>=983
```

5 Conclusion

En conclusion, nous pouvons dire, que malgré le temps qui nous était compté, et le nombre de corrections à apporter au programme, nous avons obtenu un programme capable de prendre des décisions à partir d'une base de connaissances qui s'incrémente avec la progression du personnage dans le jeu. Bien que très intéressante, l'implémentation d'une intelligence artificielle reste complexe, les décisions et déductions qui nous paraissent évidentes, sont en réalité plus complexes à programmer. De plus, ce sont des programmes qui peuvent constamment être améliorés. C'est pourquoi notre réflexion et nos tests sur ce projet continuent, dans le but de rendre notre programme, et donc le chasseur, encore plus intelligent.