# **TP 4**

Dans ce tp vous aurez besoin des fichiers suivants: **search.py**, **utils.py**, **agents.py**, **logic.py**, et **csp.py** disponibles à partir de: <a href="https://github.com/aimacode/aima-python">https://github.com/aimacode/aima-python</a>

**Remarque:** le fichier **agents.py** necessite la présence du package ipythonblocks. Vous avez deux options:

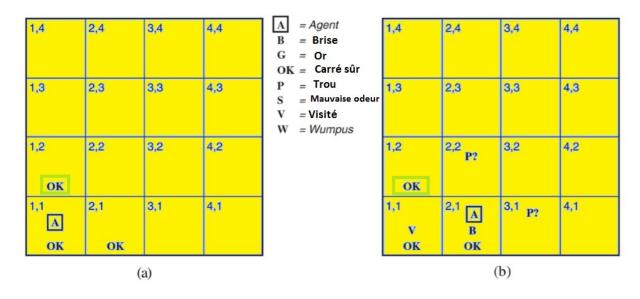
1.installer le package manquant: pip install ipythonblovks

2.mettre la ligne (from ipythonblocks import BlockGrid) en commentaire (#from ipythonblocks import BlockGrid), puisque vous n'aurez pas besoin du package dans votre TP

#### **Exercice 1:**

Utilisez un objet de la classe **PropKB** défini dans le fichier **logic.py** afin de modéliser le problème du **Monde de Wumpus** en utilisant un **agent logique.** 

Concernant le jeu, on suppose que l'agent a visité 2 cases (11 et 21) et à reçus les observations décrites dans l'image si dessous:



Les observations ainsi que les règles sont représentées en logique propositionnelle. Par exemple:

- •l'énoncé ~P11 signifie "Il n'y a pas de trou dans la case 11"
- •l'énoncé ~B11 signifie "Il n'y a pas de brise dans la case 11

## On demandera à l'agent:

- 1.S'il y a un trou dans la case 11 (P11)
- 2.S'il y a un trou dans la case 21 (P21)

## Concernant l'implémentation:

- 1. Pour créer les propositions logiques Pii et Bii, utilisez l'instruction suivante P11, P12, P21, P22, P31, B11, B21 = expr('P11, P12, P21, P22, P31, B11, B21') 2. Voici les énoncés logiques à ajouter :
  - ~P11
  - •B11 | '<=>' | ((P12 | P21))
  - •B21 | '<=>' | ((P11 | P22 | P31))
  - •~B11
  - •B21
- 3. Pour ajouter les énoncés on utilisera la méthode **tell** de l'agent créé (**agent** = **PropKB()**).
  - •Par exemple: agent.tell(~P11)
- 4.On utilisera deux méthodes d'inférences:
  - •Inférence par table de vérité: en faisant appel à la méthode **ask\_if\_true** de l'agent créé. Par exemple: agent.ask\_if\_true(~P12), et agent.ask\_if\_true(P12)
  - •Inférence par résolution: en faisant appel à la fonction **pl\_resolution.**Par exemple: pl\_resolution(agent,~P12), et pl\_resolution(agent,P12)
- 5. Pour savoir si la méthode d'inférence a pu déduire la réponse à la requête formulée ; il faut faire appel à la méthode à deux reprises : une pour la requête en question et une pour son contraire (ex : agent.ask\_if\_true(~P12), et agent.ask\_if\_true(P12)). Les deux résultats doivent être oposés (l'un doit être True ; l'autre doit être False). Si les deux sont False, sa voudra dire que la méhode n'a pas pu réaliser d'inférence.

### Exercice 2:

Dans cette partie, on encodera les connaissances en utilisant les **clauses de Horn**, avec un agent de la classe **PropDefiniteKB** (agentDC = PropDefiniteKB()). Pour la liste des clauses suivante:

• vous ajouterez les clauses à la base de connaissance de votre agent comme suit:

```
agentDC = PropDefiniteKB()
```

for c in clauses:

```
agentDC.tell(expr(c))
```

- Utilisez la méthode **pl\_fc\_entails** (inférence par chaînage avant) pour les requêtes. Par exemple:pl\_fc\_entails(agentDC, expr('G'))
- Appliquez les requêtes sur les littéraux suivants: G, H, I et J
- Refaire les mêmes étapes pour la liste des clauses suivantes:

• Appliquez une requête sur le littéral Q