Intelligence Artificielle
Quiz 1
19 Mars 2018

1. Écrivez votre nom et votre numéro sciper sur chaque feuille.
2. Placez votre carte de légitimation devant vous sur la table.
3. Documents admis: le mémento Python 3.
4. Utilisez les espaces prévus sur les feuilles de réponse si possible. Si vous utilisez des feuilles supplémentaires, répondez à chaque question sur une feuille séparée.
5. Les points qui correspondent à chaque question sont indiqués. Le quiz vaut 40 points en tout et compte pour 20% de la note finale.
6. Bonne chance !

(page vide)

1 Moteur d'Inférence (Chaînage avant) [20pts]

Cette question concerne le moteur d'inférence à chaînage avant avec variables que vous avez programmé au 3e exercice. Voici une partie de code incomplète. (Certaines fonctions auxiliaires sont reproduits sur la prochaine page.)

```
    def chaine(self):
1.      queue = self.connaissances.faits[:]
2.      self.reinitialise()

3.      while len(queue) > 0:
4.          fait = queue.pop(0)

5.          if fait not in self.solutions:
6.              self.trace.append(fait)
7.              self.solutions.append(fait)

                # Vérifie si des règles sont déclenchées par le nouveau fait.
8.              for regle in self.connaissances.regles:

9.11: ?????


12.                     if len(envs) > 0:
13.                         queue.extend(self.instancie_conclusion(regle, envs))
14.                         self.trace.append(regle)

15.     return self.solutions
```

Question 1.1 [10 pts]
La partie qui met en correpondance le `fait` avec les règles (`self.connaissances.regles`) manque. Complétez cette partie!

Utilisez les fonctions auxiliaires (identiques aux exercices) suivantes:

```
class RegleAvecVariables:
    def __init__(self, conditions, conclusion):
        self.conditions = conditions
        self.conclusion = conclusion

    def depend_de(self, fait, methode):
        envs = {}

        for condition in self.conditions:
            env = methode.pattern_match(fait, condition, {})
            if env != methode.echec:
                envs[condition] = env

        return envs

    def satisfaite_par(self, faits, cond, env, methode):
        envs = [env]

        for cond1 in conditions_a_tester:
            envs_nouveaux = []

            for fait in faits:
                for env1 in envs:
                    env1 = methode.pattern_match(fait, cond1, env1)
                    if env1 != methode.echec:
                        envs_nouveaux.append(env1)

            if len(envs_nouveaux) == 0:
                return []

            envs = envs_nouveaux

        return envs

    def __repr__(self):
        return '{} => {}'.format(str(self.conditions), str(self.conclusion))
```

Question 1.2 [10 pts]
Vous souhaitez que le moteur s'arrête dès qu'il connaît un fait qui satisfait le test `is-solution(fait)`.
Quel modification devriez-vous appliquer au code?

(page vide)

2 Inférence avec incertitude (20 pts)

Considérez un système de diagnostic medical (très simple). Vous souhaitez formaliser les connaissances que

a) une toux peut être un symptôme d'une grippe ou le résultat de trop fumer (80% grippe, 20% fumeur)

b) un nez qui coule est toujours un symptôme d'une grippe.

Question 2.1 [10 pts]
Comment pouvez-vous formaliser ces connaissances sous forme de règles, en utilisant des facteurs de certitude (CF) pour caractériser l'incertitude?

Question 2.2 [10 pts]
Un patient entre dans le cabinet et vous l'entendez tousser. Quel est le diagnostic du système, et comment est-ce qu'il est obtenu? Quand il s'approche, vous sentez la fumée et vous ne voyez aucune indication d'un nez qui coule. Est-ce que cela change le diagnostic que fournira le système? Est-ce que cela *devrait* changer le diagnostic?