TP 3 & 4 PROLOG (4h)

Problème n° 1 : une énigme

Dans 6 boîtes alignées, il y a un ver, un serpent, un lion, un canard, un chat et un pingouin. Les animaux avec des griffes sont dans les boîtes 2 et 3. Le canard est à côté du chat. Le ver est entre le serpent et le pingouin, le pingouin est entre le ver et le lion.

Soit `boite(L)` qui retourne une liste de 6 éléments, chaque élément correspondant à une boite. La valeur du $i^{ème}$ élément de la liste correspondra à l'animal contenu dans la boite i. Le prédicat `boite` aura alors la forme :

```
boite([X1, X2, X3, X4, X5, X6]) :-
```

générer les listes `[X1, X2, X3, X4, X5, X6]` dans le domaine des solutions
tester chaque solution avec le jeu de conditions de l'énigme.

Problème n° 2 : un mini master-mind

On cherche ici à simuler une version simplifiée du master mind. Dans ce jeu à deux joueurs, l'un des deux (le codeur) se fixe une suite (sans répétition) de 5 couleurs (bleu, blanc, jaune, vert, rouge) que le second (le décodeur) tente de découvrir de la manière suivante : il propose une liste de ces 5 couleurs et, en réponse, le codeur l'informe du nombre de couleurs correctement placées par rapport à son choix initial. Si les 5 couleurs sont correctement placées, la partie est terminée. Dans le cas contraire le décodeur effectue une nouvelle proposition.

On se propose de construire le programme Prolog jouant le rôle du décodeur. On adopte une méthode du type "générer et tester" décrite comme suit : on recherche une permutation de cinq couleurs compatibles avec l'historique (la liste des précédentes propositions du décodeur accompagnées de la réponse du codeur) et on la propose au codeur (l'utilisateur) pour évaluation. Une proposition est compatible avec une précédente proposition ayant n couleurs bien placées, si elle a exactement n couleurs identiques placées face à face.

Chaque proposition du décodeur est matérialisée par une liste représentant une permutation des cinq couleurs. Une proposition évaluée est un couple [V,L] où V est le nombre de couleurs correctement placées (l'évaluation) et L la proposition du décodeur. L'historique est une liste de propositions évaluées.

Concevoir le programme décrit ci-dessus. Cette première version ne tient pas compte d'éventuelles erreurs produites par le codeur lors de l'évaluation. Modifier le programme de façon à avertir le codeur dans cette éventualité.

Problème n°3 : les cannibales et les missionnaires

Trois cannibales et trois missionnaires se trouvent sur la rive gauche d'une rivière. Ils doivent atteindre la rive droite à l'aide d'une barque pouvant contenir une ou deux personnes. Que se soit sur la rive gauche, sur la rive droite ou sur la barque, si les cannibales sont plus nombreux que les missionnaires, ces derniers sont mangés. Trouvez un plan de traversée, c'est-à-dire la suite des voyages afin d'atteindre l'état désiré : tout le monde sur la rive droite et tous les missionnaires "complets".

Un état est caractérisé par un triplet *(M, C, R)* :

- *M* nombre de missionnaires sur la rive gauche ;
- *C* nombre de cannibales sur la rive gauche ;
- emplacement de la barque : *R* ive gauche ou droite.

L'état initial est donc défini par le fait : `etatInitial([3, 3, gauche])`.

L'état final est défini par le fait : `etatFinal([0, 0, droite]).`

Les actions qui décrivent les voyages autorisés en barque, sont définies par des faits où le 1er argument indique le nombre de missionnaires du `voyage` et le 2ème argument le nombre de cannibales :

```
voyage(1, 0).
voyage(2, 0).
voyage(1, 1).
voyage(0, 1).
voyage(0, 2).
```

Pour un état donné, on peut appliquer une action (un voyage), pour obtenir un nouvel état. Il faut bien sûr que celui-ci soit légal et préserve la vie des missionnaires.

Écrivez les prédicats suivants :

- `voyagePossible(E,V)` qui détermine pour un état `E` donné, un voyage possible `V` ;
- `appliquer(E1,V,E2)` qui applique un voyage `V` à un état `E1` pour donner l'état `E2` ;
- `legal(E)` qui détermine si `E` est légal pour notre système. Un état est légal si : $(0<M<3 \wedge C \leq M \wedge 3\text{-}C \leq 3\text{-}M) \vee (M = 0) \vee (M = 3)$

Le nombre d'états du graphe d'états est faible, une recherche d'une solution en utilisant un parcours en profondeur d'abord peut être mise en œuvre. Écrivez le prédicat `solution(Voyages)` qui détermine la liste des voyages successifs pour faire passer tout le monde d'un bord à l'autre. Faîtes attention de ne pas faire ramer nos amis indéfiniment…