TP02 : Problème des deux récipients

Livrables attendus : un fichier lisp + un rapport présentant et argumentant le code lisp proposé

Date de remise : lundi 14 novembre à 18H.

Enoncé

On dispose de deux récipients R1 et R2 non gradués de capacité respective 4 litres et 3 litres. Il est possible de remplir d'eau et de vider ces récipients, par terre ou l'un dans l'autre, autant de fois qu'on le désire. Au début les deux récipients sont vides.

Le but du problème est de découvrir une suite d'actions (vider et remplir les récipients d'une certaine quantité d'eau) qui permettra d'obtenir précisément 2 litres d'eau dans le récipient R1.

On se propose de résoudre ce problème à l'aide d'une recherche dans un espace d'états. On représente un état par un couple de nombres (x y), x étant le contenu du récipient R1 et y celui du récipient R2.

1) Quels sont les états initial et final ? Combien d'états peut-on avoir au plus ?
2) Les actions possibles pour un état (x y) donné peuvent être décrites par leurs conditions d'application et par leurs effets.

   Ex : (x < 4, y) -> (4 y) Si le contenu de R1 est inférieur à 4 litres, on peut le remplir. Il contiendra alors 4 litres le contenu y de R2 restant inchangé.
   (x, y > 0) -> (x 0) Si R2 n'est pas vide, on peut le vider, le contenu de R1 restant inchangé

   Décrire de la même façon les 6 autres actions envisageables.
   Les actions seront numérotées de 1 à 8.

3) Ecrire la fonction Lisp `actions(etat)` qui retourne la liste des numéros des actions autorisées pour un état donné.
4) Ecrire la fonction Lisp `successeurs(etat etatsVisites)` qui retourne la liste des successeurs non encore « visités » d'un état . On utilisera pour cela la fonction `actions` définie à la question 3.

   N.B. : Pour tester l'appartenance d'une sous-liste à une liste il faut modifier le test utilisé par la primitive `member` *de la façon suivante :*

   ```
   (member sousListe Liste :test #'equal)
   ```

5) Ecrire la fonction Lisp `rech-prof(etat)` qui explore l'espace d'états en profondeur d'abord et affiche toutes les solutions possibles. Dessiner l'arbre de recherche correspondant.
6) Ecrire la fonction Lisp `rech-larg(etat)` qui explore l'espace d'états en largeur d'abord et affiche toutes les solutions possibles. Dessiner l'arbre de recherche correspondant.
7) Comparer les deux méthodes.