Université Claude Bernard Lyon 1

M1Inf / BIA / novembre 2005

TP3_4 PROLOG (PROgrammation LOGique)

Résolution de problèmes par recherche dans un espace d'états

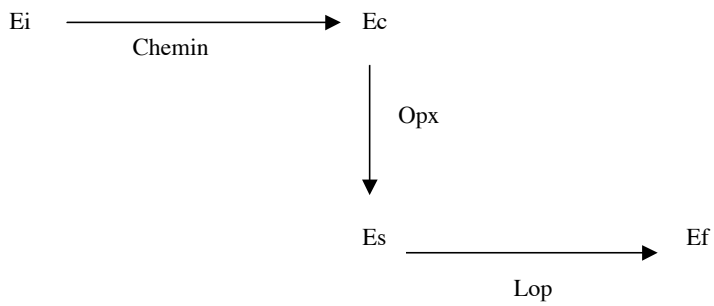
On considère des problèmes du type recherche d'un chemin entre un état initial Ei et un état final Ef, avec des opérateurs de transition pour passer d'un état à un autre. Par développement des nœuds au fil de la recherche, un graphe orienté est explicité : les sommets sont associés aux états du problème et les arêtes sont étiquetées par les opérateurs. Le graphe (arbre) obtenu s'appelle graphe (arbre) de recherche ou graphe OU.

On se propose d'écrire un programme de recherche général -en profondeur d'abord-, que l'on appliquera par la suite à des problèmes concrets (point 2 et 3).

1. Recherche en profondeur dans un espace d'états

Pour un problème, l'état initial est connu par le fait `initial(Ei)`. L'état final est connu par le fait `final(Ef)`. Les opérateurs sont connus par le prédicat `ope(Ec,Opx,Es)`, Ec désignant l'état courant, Opx un opérateur de transition et Es l'état de sortie, atteint par application de Opx.

• On atteint l'état Ec à partir de Ei en passant par une **liste d'états** mémorisés dans la liste Chemin.
• L'application de l'opérateur Opx à l'état courant Ec fait passer à un état Es.
• Lop est la **liste des opérateurs** qu'il faut appliquer depuis Es pour atteindre Ef.



1.1 Définir le prédicat de recherche en profondeur `rechPf` :

```
rechPf(Ec,Ef,.,.) :- ope(Ec,Opx,Es),. . .,rechPf(Es,Ef,. . .).
```

Liste des états depuis Ei

Liste des opérateurs jusqu'à Ef

Dans la définition de `rechPf`, on veillera à ne pas créer des boucles.

1.2 Définir le prédicat `resoudre(S)` qui donne la liste S des opérateurs à appliquer pour passer de l'état initial à l'état final.

On va maintenant appliquer le prédicat `rechPf` aux deux problèmes suivants, en définissant pour chacun d'entre eux l'état initial, l'état final et les opérateurs.

2. Le problème des flèches

2.1 Définition du problème

Six flèches sont dans la position de la figure 1. On souhaite les positionner comme dans la figure 2.

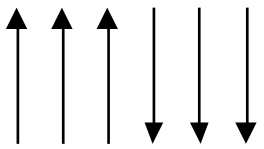


Figure 1

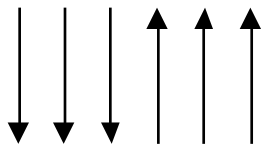


Figure 2

On symbolise ces deux états par « hhhbbb » et « bbbhhh » avec "h" pour "flèche haute" et "b" pour "flèche basse".
On définit quatre opérateurs de transition :

- R1 : retournement de deux flèches hautes adjacentes (« hh » devient « bb »)
- R2 : retournement d'une flèche haute et d'une flèche basse adjacentes (« hb » devient « bh »)
- R3 : retournement d'une flèche basse et d'une flèche haute adjacentes (« bh » devient « hb »)
- R4 : retournement de deux flèches basses adjacentes (« bb » devient « hh »)

2.2 Modéliser le problème

- Définir un prédicat `rempl(S1,S2,L1,L2)` qui est satisfait si le remplacement de la sous-liste S1 par S2 dans la liste L1 donne L2.
- Définir les prédicats `initial`, `final` et `ope` pour le problème des flèches et utiliser le prédicat `resoudre` pour trouver des solutions au problème.

3. Le problème de la simplification d'un nombre

3.1 Définition du problème

On souhaite simplifier un nombre considéré comme une liste de chiffres. On prendra l'exemple du nombre 3413242341. L'objectif est d'arriver au mot vide. Les règles de simplification sont les suivantes :

- R1 : si deux chiffres adjacents sont égaux, on peut les supprimer tous les deux.
- R2 : on peut permuter deux chiffres adjacents si la valeur absolue de leur différence est supérieure à 1.
- R3 : la séquence (n,n-1,n) peut être remplacée pas la séquence (n-1,n,n-1).

3.2 Résolution du problème

Définir les prédicats `initial`, `final` et `ope` pour ce problème et utiliser le prédicat `resoudre` pour lui trouver des solutions.

4. Amélioration de la recherche

En observant la résolution du problème des flèches, on constate que le troisième état généré était très proche de l'état final. Or l'ordre figé des opérateurs et le fait que la première substitution possible est appliquée ont conduit le programme vers une solution plus longue.
Nous allons donc améliorer la recherche en profondeur en examinant tous les états auxquels peuvent conduire les opérateurs applicables. Dans le cas où un de ces états est l'état final, c'est la règle qui y conduit qui sera appliquée. Sinon, on utilise comme précédemment le premier opérateur applicable.

• Définir le prédicat `resoudre+`, qui utilise le prédicat `rechPf+` implémentant cette stratégie. (On pourra utiliser `findall` et le prédicat prédéfini `nth1`).

5. Pour aller plus loin

Pour ceux qui ont le temps d'en faire plus : on peut encore améliorer la recherche, en définissant pour chaque problème un prédicat calculant une distance entre deux états. Dans le prédicat de recherche, on considère tous les états auxquels peuvent conduire les opérateurs applicables, et l'on choisit l'opérateur qui conduit à l'état le plus proche de l'état final (au sens de la distance précédemment définie).

6. Procédure de notation du TP

Ce TP sera noté à la fin de la deuxième séance. Un rapport de synthèse (court et individuel) et une démonstration seront demandés.