

TP5 – Prolog & la recherche dans un graphe d'états

Arthur Aubret, Hugo Castaneda, Rémy Chaput,
Nathalie Guin, Marie Lefevre

LA MISE EN PLACE DE LA RECHERCHE

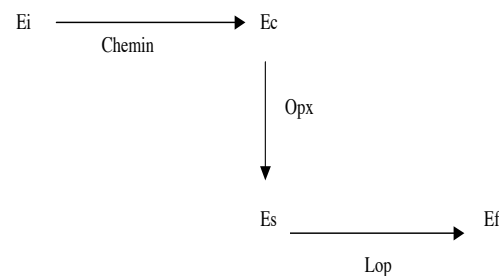
On considère des problèmes du type recherche d'un chemin entre un état initial E_i et un état final E_f , avec des opérateurs de transition pour passer d'un état à un autre. Par développement des nœuds au fil de la recherche, un graphe orienté est explicité : les sommets sont associés aux états du problème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

On se propose d'écrire un programme de recherche général -en profondeur d'abord-, que l'on appliquera par la suite à deux problèmes concrets.

Pour un problème, l'état initial est connu par le fait `initial(E_i)`. L'état final est connu par le fait `final(E_f)`. Les opérateurs sont connus par le prédicat `opérateur(E_c , O_{px} , E_s)`, E_c désignant l'état courant, O_{px} un opérateur de transition et E_s l'état de sortie, atteint par application de O_{px} .

La figure ci-contre résume le principe de résolution :

- On atteint l'état E_c à partir de E_i en passant par une liste d'états mémorisés dans la liste `Chemin`.
- L'application de l'opérateur O_{px} à l'état courant E_c fait passer à un état E_s .
- `Lop` est la liste des opérateurs qu'il faut appliquer depuis E_s pour atteindre E_f .



Question 1 : Définir le prédicat de recherche en profondeur `rechPf` tel que :

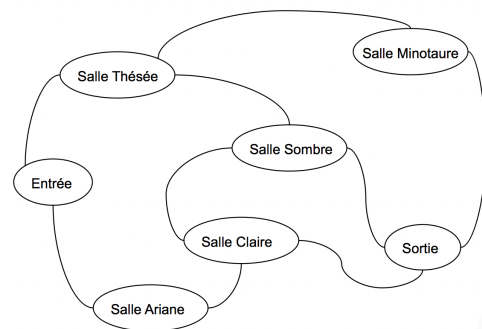
`rechPf(E_c , E_f , $L1$, $L2$) :- opérateur(E_c , O_{px} , E_s), ..., rechPf(E_s , E_f , ..., ...).`

sachant que $L1$ est la liste des états déjà parcourus et $L2$ la liste des opérateurs depuis E_c jusqu'à E_f .

Pour tester votre prédicat, utilisez **le problème du labyrinthe** illustré à droite.

Un labyrinthe possède une entrée et une sortie. Le but du jeu est de trouver la sortie sans vous faire manger par le Minotaure.

Ce problème peut être décrit de la manière suivante :



```

initial(entree).
final(sortie).
  
```

```

interdit(minotaure).
opérateur(E1,[E1,E2],E2) :- couloir(E1,E2).
opérateur(E1,[E1,E2],E2) :- couloir(E2,E1).

couloir(entree, thesee).
couloir(entree, ariane).
couloir(thesee, minotaure).
couloir(thesee, sombre).
couloir(claire, sombre).
couloir(claire, sortie).
couloir(minotaure, sortie).
couloir(ariane, claire).
couloir(sombre, sortie).

```

Question 2 : Définir le prédicat `resoudre(S)` indépendant du problème qui donne la liste `S` des opérateurs à appliquer pour passer de l'état initial à l'état final. Le résultat de résoudre sur le problème du labyrinthe donne :

```

?- resoudre(S).
S = [[entree, thesee], [thesee, sombre], [sombre, sortie]] ;
S = [[entree, thesee], [thesee, sombre], [sombre, claire], [claire, sortie]] ;
S = [[entree, ariane], [ariane, claire], [claire, sombre], [sombre, sortie]] ;
S = [[entree, ariane], [ariane, claire], [claire, sortie]] ;

false.

```

LE PROBLEME DES CRUCHES

On va maintenant appliquer le prédicat `rechPf` sur un autre problème, en définissant l'état initial, l'état final et les opérateurs : **le problème des cruches**.

**J'ai deux cruches : une cruche de 8 litres et une cruche de 5 litres,
je souhaite avoir exactement 4 litres dans l'une d'elles.
La fontaine coule indéfiniment.**

Question 3 : Pour ce nouveau problème, définir les prédicats `initial`, `final` et `opérateur`.

Le résultat du prédicat résoudre sur le problème des cruches donne comme solution la plus « courte » celle ci-dessous, mais ce n'est pas la seule !

```

?- resoudre(S),write(S).
[remplir5, transvaser5vers8, remplir5, transvaser5vers8, vider8,
transvaser5vers8, remplir5, transvaser5vers8, remplir5, transvaser5vers8]
L = [remplir5, transvaser5vers8, remplir5, transvaser5vers8, vider8,
transvaser5vers8, remplir5, transvaser5vers8|...].

```

Question 4 : Combien d'opérateurs avez-vous défini ? Si vous en avez défini plus de 6, améliorez votre programme...

(BONUS) LE SINGE ET LES BANANES

Pour ceux qui veulent encore jouer, reprenons le problème du singe et des bananes vu en TD.

Ce problème fait intervenir un singe dans un laboratoire et des bananes qui pendent au plafond, hors de portée de l'animal, qui peut cependant grimper sur une caisse pour atteindre les fruits.

Au départ, le singe se trouve au point A, les bananes au point B et la caisse au point C. Le singe et la caisse ont une hauteur de 1 mètre. Le plafond est à 2 mètres de haut. Le singe peut aller d'un point à un autre, déplacer un objet d'un point à un autre, grimper sur un objet ou en descendre, saisir ou lâcher un objet. Pour saisir un objet, le singe doit être sur le même emplacement que l'objet et à la même hauteur.

Le singe veut tromper les chercheurs pendant qu'ils sont allés boire un café. Il veut saisir les bananes tout en remettant la caisse à l'emplacement initial.

Question 5 : Pour ce nouveau problème, définir les prédicats `initial`, `final` et `opérateur`.

Le résultat du prédicat résoudre sur le problème du singe donne comme solution la plus « courte » celle ci-dessous :

```
?- findall(S, (resoudre(S),length(S,N), N<8), L).
```

```
L = [[allerEnC,  deplacerCaisseVersB,  grimper,  saisirBanane,  descendre,
deplacerCaisseVersC, allerEnA]].
```