M1Info/BIA/2013
TP3-4 - PROLOG
(PROgrammation LOGique)

Nadia Kabachi, Marie Lefevre, Alain Mille

14 et 21 novembre 2013

Avertissement : Ce TP sous forme de projet sera noté. Vous devez rendre un Compte Rendu (fichier : TP3_4_Nom1_Nom2.pdf) correspondant au plan proposé ***avant la séance du 28 novembre*** accompagné des codes sources commentés. Le tout sous format électronique envoyé à votre encadrant dont l'objet de l'email : TP3_4_Nom1_Nom2.

Dans le CR doivent figurer :

- Les auteurs : Noms Prénoms
- Les réponses aux questions
- Ce qui a été programmé par rapport au sujet (ce qui tourne et ce qui ne tourne pas)
- Les questions que vous vous êtes posées, les choix que vous avez faits
- Des jeux d'essais montrant les différents cas de figures traités par le programme

Objectif du TP :

1) Illustrer le cours sur la modélisation de problème
2) Illustrer le cours sur la résolution de problème
3) Pratiquer Prolog pour réaliser des « moteur » de résolution de problèmes.

Organisation du travail : La partie 1 doit être traitée lors de la première séance. Préparez la partie 2 à la maison. Tout doit être terminé à la fin de la seconde séance.

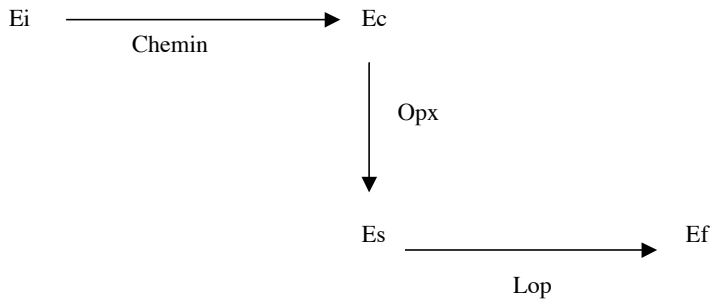
Partie 1 : Recherche dans un graphe d'états

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 deux problèmes concrets.

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.



Q1 (1 pt) : 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

Q2 (1 pt) : 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` à deux problème précis, en définissant pour chacun, l'état initial, l'état final et les opérateurs.

Le problème des flèches : définition

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

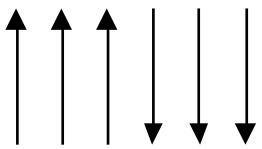


Figure 1

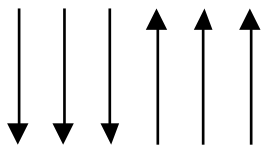


Figure 2

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 »)

Le problème des flèches : modélisation

On définit 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

On définit les prédicats `initial`, `final` et `ope` pour le problème des flèches et on utilise le prédicat `resoudre` pour trouver des solutions au problème.

Q3 (1 pt) : Faire le schéma de l'arbre d'exploration des états sur 2 niveaux (état de départ et les états accessibles par les opérateurs valides).

***Q4 (2 pts) : Définir les prédicats* `rempl(S1,S2,L1,L2)`, `initial`, `final` *et* `ope`.**

Q5 (1 pts) : Donner un exemple d'exécution du prédicat* `resoudre` *sur le problème des flèches (test à présenter en séance) et commenter son exercussion dans le CR.

Q6 (3 pts bonus) : Quelle heuristique proposez-vous pour éviter d'explorer toutes les possibilités ? Illustrer sur le schéma de l'arbre d'états en montrant les chemins qui seront ainsi écartés au profit de chemins plus prometteurs. Proposer une modification du programme des flèches qui permet de tenir compte de l'heuristique. Rendre le code commenté et le présenter en séance.

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 chacune d'elles. La fontaine coule indéfiniment.

Q7 (4 pts) : Proposer un code Prolog sur le même modèle que précédemment (etat initial, etal final, operateur) pour le problème des cruches. Le codage doit être sans heuristique. Rendre le code commenté et le faire tester en séance.

Q8 (2 pts bonus) : Proposez une heuristique et le code correspondant. Rendre le code et le faire tester en séance.

Partie 2 : Système à base de règles

On considère des problèmes que l'on peut résoudre à l'aide d'un système à base de règles. Nous allons dans un premier temps définir un moteur d'inférence à chaînage avant, que nous appliquerons sur un problème jouet. Vous définirez ensuite la base de règles relative au problème de l'atelier de découpe de cuir. Attention, votre moteur doit être générique et fonctionner sur le problème jouet, le problème du cuir mais également sur un autre jeu de test (une autre base de règles) que nous utiliserons pour tester votre TP.

Moteur en chaînage avant

Notre problème jouet est décrit comme suit.

Nous disposons de 5 règles : R1 : A et B → C
R2 : C et D → F
R3 : F et B → E
R4 : F et A → G
R5 : G et F → B

Nous connaissons trois faits : A, C et D

Et nous voulons démonter le fait E.

Pour réaliser le moteur d'inférence en chaînage avant, il faut :

Q9 (1 pts) : Définir la base de règles sous forme de listes composées de listes de prémisses et de listes de conclusions, grâce à un prédicat `regle/1` :

```
regle(ri) :- si(Liste de prémisses),
             alors(Liste de conclusions).
```

Q10 (1 pts) : Définir un prédicat permettant à l'utilisateur d'initialiser la base de faits. On utilisera le prédicat `assert` pour ajouter `vrai(Fait)` pour les faits positifs et `faux(Fait)` pour les faits négatifs.

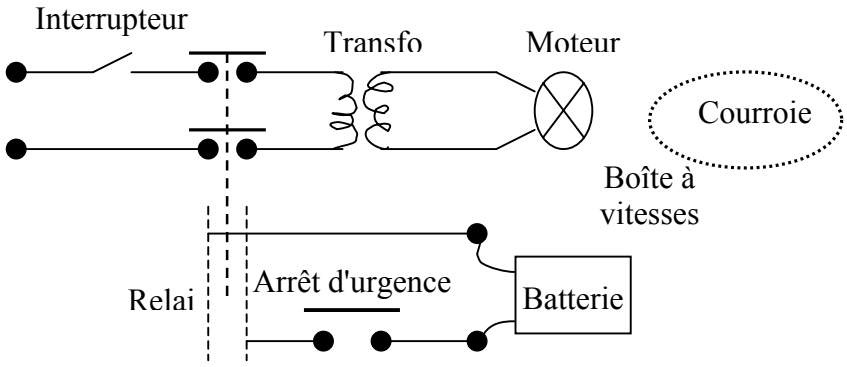
Q11 (4 pts) : Définir un prédicat `resoudre` qui sature la base de règles et produit une trace de son fonctionnement. L'algorithme utilisé pour ce moteur sera le suivant :

```
Changement <-- Vrai
Tant que Changement est Vrai
   Changement <-- Faux
   Boucle sur les règles : soit R une règle de BaseRègles
      Si R n'est pas marquée et si les prémisses de R
      appartiennent à BaseFaits
         Alors ajouter les conclusions de R à BaseFaits
         changement <-- Vrai
         marquer R
      FinSi
   FinBoucle
FinTantQue
```

Sur notre exemple jouet, le fait E est démontrable et donne la trace {R2, R4, R5, R1, R3}.

Atelier de découpe de cuir

Remerciements : *Cette partie du sujet est très fortement inspiré du cours de Guy Caplat (Département informatique – INSA de Lyon)*



Déroulement actuel de la résolution de problème :

A l'atelier de découpe de cuir, on appelle l'expert quand un problème survient :

- Si l'ouvrier veut mettre en route son installation et que ça ne marche pas : la courroie ne tourne pas. Il faut donc réparer le plus vite possible.

- Chaque composant de cette installation a un rôle bien défini : la courroie, entraînée par le moteur, lui-même relayé par la boîte à vitesses interviennent dans la fonction de découpe du cuir. L'interrupteur, le secteur et le transfo servent à l'alimentation en énergie. Le bouton d'arrêt d'urgence, le relais et la batterie pour assurer la sécurité.

- On peut faire un certain nombre d'observations pour savoir si un composant est en défaut ou non. Une fonction peut ne pas être assurée pour deux raisons : soit parce qu'elle est défaillante à cause d'un des composants qui la constituent, soit parce qu'elle dépend d'une autre qui est à son tour défaillante. Par exemple, on ne peut pas avoir de découpe s'il n'y a pas d'alimentation en énergie, et la sécurité peut être amenée à interrompre l'alimentation si nécessaire.

- Quelquefois, c'est tout simplement la batterie qui n'alimente pas le relais : soit elle est morte (et il faut la changer) - la tension aux bornes dans ce cas est nulle -, soit elle est déchargée - la tension n'est pas nulle mais ne fait pas les 24 Volts nécessaires – et il suffit de la recharger. Quand le relais n'est pas alimenté, alors le transformateur ne l'est pas et le moteur non plus évidemment.

- Dans le cas où le secteur fonctionne et qu'on n'a pas procédé à un arrêt d'urgence, si la batterie est bonne et qu'il n'y a pas de courant au primaire du transformateur, c'est que le relais est foutu.

- Il se peut aussi que le moteur soit grillé. Là, ce n'est pas difficile à savoir : le moteur ne tourne pas et il y a de la tension au secondaire du transformateur. En revanche, s'il n'y en a pas au secondaire du transformateur mais qu'il y en a au primaire, c'est que le transformateur est mort. Il faut le changer. Si le moteur tourne, et que la courroie ne tourne pas, c'est que la vitesse n'est pas bien passée. Il faut ouvrir le carter de la boîte et débloquer la tringlerie à la main.

Les dirigeants de la société de découpe de cuir désirant alléger le travail de leurs experts, et surtout leur éviter des déplacements trop fréquents en atelier pour des incidents classiques, les dirigeants ont décidé de mettre en place un système d'aide au diagnostic et à la résolution des problèmes de l'atelier afin de permettre à l'ouvrier de se dépanner sans faire intervenir l'expert.
L'ouvrier rencontrant un problème n'aura alors qu'à soumettre le symptôme détecté pour que le système identifie le problème et lui propose une solution adaptée, ainsi que la catégorie du problème (découpe, énergie, sécurité). Dans le cas où le symptôme n'est pas suffisant, le système interroge l'ouvrier de manière à compléter les observations et pouvoir fournir une réponse.

Q12 (4 pts) : Définir la base de règles pour ce problème. Tester votre base et votre moteur avec au moins deux ensembles de faits différents. Mettez les traces d'exécution de ces deux exemples dans votre CR.

Exemple d'exécution souhaitée :

```
1 ?-
faits([non(relais_non_alimente),non(primaire_transformateur_non_alimente),secondaire_transform
ateur_non_alimente]).

Yes
2 ?- go.
r10
non(relais_non_alimente) non(primaire_transformateur_non_alimente)
secondaire_transformateur_non_alimente transformateur_mort
r11
non(relais_non_alimente) non(primaire_transformateur_non_alimente)
secondaire_transformateur_non_alimente transformateur_mort moteur_non_alimente
r12
non(relais_non_alimente) non(primaire_transformateur_non_alimente)
secondaire_transformateur_non_alimente transformateur_mort moteur_non_alimente moteur_arrete
r14
non(relais_non_alimente) non(primaire_transformateur_non_alimente)
secondaire_transformateur_non_alimente transformateur_mort moteur_non_alimente moteur_arrete
courroie_arretee

Yes
```