IN112 : BE Prolog

Author : Christophe Garion <garion@isae.fr>
Public : SUPAERO 1A
Date : 20/05/15

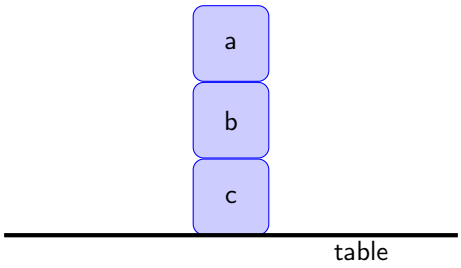
Résumé

Le but de ce BE est d'utiliser la programmation logique et le langage Prolog pour résoudre un problème.

1 Problème

Nous nous intéressons ici à un problème classique en Intelligence Artificielle, à savoir la génération de plans. Un plan est une suite d'actions qui permettent d'atteindre un but. Par exemple, pour prendre un café à SUPAERO, je sais qu'il faut que je me rende à la machine à café au BatEns, que je mette suffisamment d'argent dans la machine, que je sélectionne une boisson, que j'attende qu'elle soit prête, que je la prenne et que je récupère ma monnaie.

On suppose ici qu'un robot doit déplacer trois cubes. Le robot ne peut prendre qu'un seul cube à la fois, donc si les cubes sont empilés, le robot doit d'abord enlever les cubes au dessus du cube à déplacer. La situation initiale est la suivante : le cube *a* est sur le cube *b*, le cube *b* sur le cube *c* et le cube *c* sur la table.



2 Questions

1. représenter grâce à un prédicat la situation initiale.
2. on souhaite maintenant représenter l'action *atomique* put_on qui consiste à placer directement un objet sur un autre. Pour cela, il faut spécifier l'action. Une action peut se spécifier par :
   – ses préconditions, qui sont les formules qui doivent être vraies pour que l'on puisse réaliser l'action ;
   – ses postconditions, qui sont les formules qui seront vraies après l'exécution de l'action.

   Donner de façon informelle les préconditions et les postconditions de l'action `put_on`. Vous pourrez bien évidemment rajouter des prédicats.
3. la question que l'on peut se poser est la suivante : comment exprimer les postconditions en Prolog ? En effet, les postconditions représentent les changements effectués sur le monde et faut pouvoir retirer certaines formules représentant le monde et en ajouter d'autres. Prolog ne permet pas cela directement, car l'utilisation d'un atome dans l'algorithme de Prolog ne l'« ajoute » pas, on cherche juste à vérifier s'il est vrai ou non.

   Heureusement, il existe deux prédicats permettant d'ajouter ou de retirer un fait :
   – **`assertz`**`(f)` qui permet d'ajouter le fait `f` ;
   – **`retract`**`(f)` qui permet de retirer le fait `f` (ou tous les faits unifiables avec `f`).

   Pour un prédicat `pred/2` pouvant être utilisé par ces prédicats, il faut préciser en préambule de fichier `:- dynamic(pred/2).` (ne pas oublier le « . »). Ceci permet de spécifier à Prolog que l'on va pouvoir ajouter des clauses par rapport à ce qui existe dans le programme.

   Pour effacer tous les faits concernant un prédicat, on peut évaluer dans l'interpréteur le prédicat **`retractall`**`(pred(X1,...,Xn))`. On effacera alors de la base de faits les faits unifiables avec `pred(X1,...,Xn)`.

   Écrire la spécification de l'action `put_on` en Prolog.
4. le but de ce BE est de trouver un plan, i.e. une suite d'actions. Il faut donc pouvoir « enregistrer » les mouvements qu'effectue le robot. Modifier la définition de `put_on` pour que lorsque l'on appelle `put_on(a,b)` on enregistre un fait `move(a,b)`.
5. poser les requêtes `put_on(a,table)`, `put_on(c,a)`, `put_on(b,table)`, `put_on(c,a)` et utiliser **`listing`** sur le prédicat `move` pour vérifier que cela fonctionne.
6. `put_on` nous permet de placer un objet sur un autre s'il n'y a pas d'autre objet sur ces objets. On cherche maintenant à spécifier deux prédicats :

- `clear(a)` permet de libérer un objet, i.e. d'enlever tous les objets situés dessus en les plaçant sur la table.
- `r_put_on(a,b)` permet de placer l'objet *a* sur l'objet *b* de façon générale.

7. on cherche maintenant à spécifier des tâches par une liste de faits que l'on veut atteindre.
   (a) dans un premier temps, on introduit un prédicat `achieve` qui prend en paramètre une liste de termes représentant des buts à atteindre et affiche les actions nécessaires à la satisfaction de ces buts. Par exemple, `achieve([on(a,c), on(c,b)])` affichera la liste de mouvements nécessaires à l'obtention de la situation dans laquelle le cube *a* est sur le cube *c* et le cube *c* sur le cube *b*.
   On ne s'intéressera ici qu'à des buts du type « l'objet *x* est sur l'objet *y* », mais le prédicat `achieve` pourrait également servir pour d'autres buts/actions que le robot pourrait effectuer (se déplacer d'une pièce à l'autre etc.).
   (b) que se passe-t-il si on demande l'évaluation de `achieve([on(a,c), on(c,b)])` ?
   (c) pour pallier le problème précédent, il faut pouvoir trier correctement la liste des buts à atteindre. Proposer une solution simple permettant de réaliser ce tri *en supposant que la liste de buts est cohérente*.

3 Documents à rendre

Vous devez renvoyer pour le **20 mai 2015 à 12h15** le source d'un programme Prolog répondant aux questions ci-dessus. **Le fichier source devra impérativement s'appeler `cube-login.pl` où `login` est votre login au SI.** Les réponses aux questions pourront y être insérées sous forme de commentaires ou dans un fichier texte séparé (format texte simple, pas de OpenOffice ou PDF). Le fichier sera envoyé par mail à garion@isae-supaero.fr et **le sujet du mail devra être `[IN112] projet Prolog`**.