Programmation des Systèmes Experts

Deuxième Année ENSIMAG 2008/2009

James L. Crowley

Travaux Pratique :1 25 Feb 2009

GRAPHSEARCH en CLIPS

L'algorithme GRAPHSEARCH pour un réseau d'emplacements peut être codé en CLIPS.

Dans cet exercice, nous allons représenter le point de départ et le but par les "structures" définies par des "templates" avec les attributs nom, x et y. L'arbre de recherche sera composé d'objets de type Noeud dont les attributs sont :

nom : Le nom de l'emplacement représenté par ce noeud.
status : l'état du noeud {Open, Closed, Active, new, etc}
père : Le noeud qui est au-dessus dans l'arbre de recherche
f : Le coût d'un chemin qui passe par cet emplacement.
g : Le coût dans le graphe entre le départ et cet emplacement.
h : Le coût estimé entre cet emplacement et le but.
x, y : Les coordonnées cartésien de l'emplacement.

Nous aurons besoins déclarer une fonction externe "distance" avec l'expression :

```
(deffunction distance (?x1 ?y1 ?x2 ?y2)
 (bind  ?dx (- ?x1 ?x2))
 (bind  ?dy (- ?y1 ?y2))
       (sqrt (+ (* ?dx ?dx) (* ?dy ?dy)))
)
```

La fonction distance peut être appelée dans une règle comme suit:

```
=>
       (modify ?E (h ( distance ?x1 ?y1 ?x2 ?y2)))
```

Pour chacune des règles que vous écrivez, donnez un nom à la règle de forme Rn où n est soit un entier, soit le nom d'une paire d'emplacements. Exemples, R1, R2 .

a) Ecrire les expressions en CLIPS pour déclarer les templates de type Départ, But et Noeud.

b) Le noeud choisi pour exploration pendant chaque cycle est étiqueté avec un status "active". Au début, un noeud "active" est créé pour le point de départ. Ecrire une règle (R1) qui crée un noeud pour le point de départ avec un status "active".

c) Ecrire une règle (R2) pour détecter si le noeud "active" est le but. Quand le système a trouvé le but, faire imprimer "but trouvé" et s'arrêter. Pour assurer que cette règle a priorité sur les règles R1 à R6, donnez luis une salience de 40.

d) Il faut tester si les noeuds avec status "new" corresponds aux chemins déjà explorés (les noeuds de status "closed"). Ecrire la règle (R3) qui détruit les noeuds "new" avec le même nom qu'un noeud "closed". Pour assurer que cette règle a priorité, donnez luis une salience de 40.

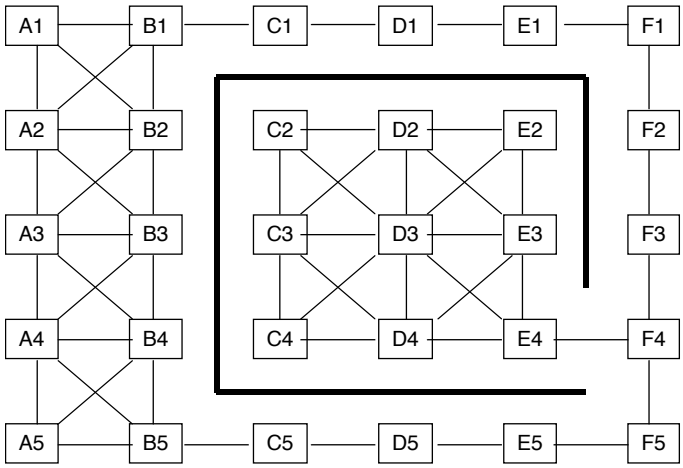
e) Il faut calculer le coût de chemin passant par chaque noeud de status "new". Ecrire la règle R4 qui calcule le coût h pour un noeud "new" en utilisant la fonction "distance". Vérifier qu'h est -1 et qu'il n'y a pas un noeud closed avec le même nom.

f) Ecrire la règle, R5, qui calcule le coût estimé f à partir de h et g. Cette règle doit changer le status à open (ouvert a l'exploration).

g) Apres le premier cycle, le noeud avec un status "open" et un coût estimé (f) minimal est rendu "Active". Ecrire une règle (R6) pour choisir un noeud avec un coût minimal et le rendre "active". Vérifier qu'il n'existe pas un noeud avec status "new" ou "active" (voir b, c).

Pour tester votre solution :

Soit que vous avez un robot mobile avec le réseau d'emplacements ci-dessus. Votre robot est à l'emplacement A3. Il a besoin d'aller à l'emplacement C3.



Le réseau est composé d'emplacement défini par le "template" :

```
;;
;;  Regles d'affecter les locations des emplacements
;;

(deftemplate emplacement
        (slot nom (type SYMBOL) (default NIL))
        (slot x (type NUMBER))
        (slot y (type NUMBER))
        (multislot voisins (default NIL))
)

(deffacts reseaux-d-emplacements
        (emplacement (nom A1) (x 0) (y 1) (voisins A2 B1 B2))
        (emplacement (nom A2) (x 0) (y 2) (voisins A1 A3 B1 B2 B3))
        (emplacement (nom A3) (x 0) (y 3) (voisins A2 A4 B2 B3 B4))
        (emplacement (nom A4) (x 0) (y 4) (voisins A3 A5 B3 B4 B5))
        (emplacement (nom A5) (x 0) (y 5) (voisins A4 B4 B5))
        (emplacement (nom B1) (x 1) (y 1) (voisins A1 A2 B2))
        (emplacement (nom B2) (x 1) (y 2) (voisins A1 A2 A3 B1 B3 C1))
        (emplacement (nom B3) (x 1) (y 3) (voisins A2 A3 A4 B2 B4))
        (emplacement (nom B4) (x 1) (y 4) (voisins A3 A4 A5 B3 B5 C5))
        (emplacement (nom B5) (x 1) (y 5) (voisins A4 A5 B4 C5))
        (emplacement (nom C1) (x 2) (y 1) (voisins B1 B2 D1))
        (emplacement (nom C2) (x 2) (y 2) (voisins C3 D2 D3))
        (emplacement (nom C3) (x 2) (y 3) (voisins C2 C4 D2 D3 D4))
        (emplacement (nom C4) (x 2) (y 4) (voisins C3 D3 D4))
        (emplacement (nom C5) (x 2) (y 5) (voisins B4 B5 D5))
        (emplacement (nom D1) (x 3) (y 1) (voisins C1 E1))
        (emplacement (nom D2) (x 3) (y 2) (voisins C2 C3 D3 E2 E3))
        (emplacement (nom D3) (x 3) (y 3) (voisins C2 C3 C4 D2 D4 E2 E3 E4))
        (emplacement (nom D4) (x 3) (y 4) (voisins C3 C4 D3 E3 E4))
```

```
        (emplacement (nom D5) (x 3) (y 5) (voisins C5 E5))
        (emplacement (nom E1) (x 4) (y 1) (voisins D1 F1 F2))
        (emplacement (nom E2) (x 4) (y 2) (voisins D2 D3 E3))
        (emplacement (nom E3) (x 4) (y 3) (voisins D2 D3 D4 E2 E4))
        (emplacement (nom E4) (x 4) (y 4) (voisins D3 D4 E3 F4))
        (emplacement (nom E5) (x 4) (y 5) (voisins D5 F4 F5))
        (emplacement (nom F1) (x 5) (y 1) (voisins E1 F2))
        (emplacement (nom F2) (x 5) (y 2) (voisins F1 F3))
        (emplacement (nom F3) (x 5) (y 3) (voisins F2 F4))
        (emplacement (nom F4) (x 5) (y 4) (voisins E4 F3 F5))
        (emplacement (nom F5) (x 5) (y 5) (voisins E5 F4))
)

;;
;;  Regles pour le connectivite des emplacements
;;  Ces regles genere les nouvelles noueds dans l'arbre de recherche
;;

(defrule Genere-voisins
        "genere les noeud pour les voisins de ?X"
        (declare (salience 20))
        (noeud (nom ?p) (status active) (g ?g))
        (emplacement (nom ?p) (voisins $? ?v $?))
=>
        (assert (noeud (nom ?v) (status new) (pere ?p) (g =(+ ?g 1))))
)

;;
;; regle pour fermer un noeud active
;;

(defrule fermer-noeud
        "passer aux status closed apres les voisins sont cree"
        (declare (salience 10))
        ?N <- (noeud (status active))
=>
        (modify ?N (status closed))
)

;;
;; affecter les locations
;;

(defrule Affecter-position-emplacement
        "affecter le location de chaque emplacement"
        (declare (salience 30))
        ?N <- (noeud (nom ?n) (x -1) (y -1))
        (emplacement (nom ?n) (x ?x) (y ?y))
=>
        (modify ?N (x ?x) (y ?y))
```

```
)

(defrule Affecter-position-des-buts
        "affecter le location de chaque emplacement"
        (declare (salience 10))
        ?B <- (but (nom ?n) (x -1) (y -1))
        (emplacement (nom ?n) (x ?x) (y ?y))
=>
        (modify ?B (x ?x) (y ?y))
)



(defrule Affecter-Location-des-depart
        "affecter le location de chaque emplacement"
        (declare (salience 10))
        ?D <- (depart (nom ?n) (x nil) (y nil))
        (emplacement (nom ?n) (x ?x) (y ?y))
=>
        (modify ?D (x ?x) (y ?y))
)

;;
;; Regle d'initialiser le recherche
;;

(defrule init "Initialise avec le depart et le but"
        (initial-fact)
 (not (depart))
=>
        (printout t "Le nom de depart? ")
        (bind ?n (read))
        (assert (depart (nom ?n) ))
        (printout t "Le nom du but? ")
        (bind ?n (read))
        (assert (but (nom ?n)))
)
```

```
;;
;; regle d'areter la recherche
;;

(defrule termine-recherche  "close all open nodes"
        (chemin $?)
        ?N <- (noeud (status open))
=>
        (modify ?N (status closed))
        (halt)
)

;;
;; regle de composer le chemin
;;

(defrule compose-chemin  "compose un liste du chemin"
        ?C <- (chemin ?n $?q)
        (noeud (nom ?n) (pere ?p&:(neq ?p nil)))
        (but (nom ?b))
        (test (neq ?p ?b))
=>
        (retract ?C)
        (assert (chemin ?p ?n $?q))
)

;;
;; regle d'imprimer le chemin
;;

(defrule imprimer-chemin  "compose un liste du chemin"
        (chemin ?d $?q ?b)
        (depart (nom ?d))
=>
        (printout t "Le chemin de " ?d " a " ?b " passe par " $?q crlf)
        (halt)
)
```

Algorithme GRAPHSEARCH (Nilsson)

Symboles:

T : Arbre de Recherche

B : Ensemble des buts

s : Noeud de départ (Start)

M : Liste des successeurs

Open : Ensemble des noeuds "ouverts"
(liste des noeuds à explorer)

Closed : Ensemble des noeuds "fermés"
(liste des noeuds déjà explorés)

n, e : noeuds

Algorithme (Rappel) :

1) Créer T, Open et Closed, (tous initialisés à l'ensemble vide).
2) Mettre s dans Open et comme racine de T.
3) Loop : si OPEN est vide, Alors Exit avec échec.
4) Extraire n de Open. Open <- Open - n,
Closed <- Closed + n
5) Si n est un élément de B alors : Succès!!
Construire une pile avec les noeuds de T constituant le chemin entre n et s.
Exit en rendant la pile.
FinSi
6) Expansion de n : Créer M (les successeurs de n)
7) Pour chaque e de M,
si e est dans Closed alors M <- M - e.
sinon
a) optionel : Calculer le coût estimé f(e) pour e.
b) ajouter e dans Open (LIFO, FIFO ou Triée)
c) ajouter e à T comme successeur de n.
8) Aller à l'étape 3