

UTBM

Automne 2004

IA41

INTELLIGENCE ARTIFICIELLE

représentation des connaissances et résolution de problèmes

Responsable: Jean César

Intervenants en TD: Olivier Grunder, Yassine Ruichek

en TP : Mohamed Hariti

Travaux Dirigés n° 2

Semaine du 27 septembre 2004

(corrigé)

SUJET: ADRESSE DANS UNE LISTE ARBORESCENTE (suite et fin)

Recherche & illustration d'une adresse

- 1) Spécifier et définir formellement la fonction **indice** qui rend l'indice de la première occurrence d'un terme dans une liste (au premier niveau). On rappelle la convention selon laquelle les termes d'une liste sont indicés à partir de 0.

Spécification: la fonction **indice** rend l'indice de la première occurrence du terme, s'il existe, et sinon rend la valeur logique faux: la liste vide.

De façon à rester fonctionnel, où stocker l'indice en cours de calcul?

L'indice est stocké dans un paramètre supplémentaire, initialisé à 0, pour pouvoir rendre au choix cet indice ou nil.

Profil: Expr x List x N --> N

Jeu d'essais:

indice c,(),0 = liste vide

indice (u v),(a b (u v) z),0 = 2

indice c,.(a b (u v) z),0 = liste vide

Définition formelle:

Pour tout (e,L,n) appartenant à Expr x List x N

$L = () \Rightarrow \text{indice } e, L, n = ()$

$L \# () \text{ et } e = \text{tête } L \Rightarrow \text{indice } e, L, n = n$

$L \# () \text{ et } e \neq \text{tête } L \Rightarrow \text{indice } e, L, n = \text{indice } e, \text{reste } L, n+1$

- 2) Spécifier et définir formellement la fonction **treeaddress** qui rend l'adresse arborescente de la première occurrence d'une expression dans une liste.

2-a) Spécifier l'énoncé selon que l'expression cherchée existe ou non dans la liste.

Spécification: la fonction **treeaddress** rend l'adresse arborescente de la première occurrence du terme, s'il existe, et sinon rend la valeur logique faux: la liste vide.

2-b) Donner un exemple de réussite, un exemple d'échec, et le cas limite:

$\text{treeaddress } (d), (z y x (w v u) (t s r (q p o n m (l ((k j i h g f) e) (d))) c b)a) = (4 3 5 2)$

$\text{treeaddress } 2, (z y x (w v u) (t s r (q p o n m (l ((k j i h g f) e) d)) c b)a) = \text{nil}$

$\text{treeaddress } x, () = \text{nil}$

2-c) Montrer que l'utilisation d'**indice** conduirait à parcourir la liste deux fois. On va donc chercher une solution qui n'utilise pas **indice**.

indice parcourt une fois la liste pour tester l'égalité avec chaque terme. Il faudrait la parcourir une seconde fois pour tester l'atomicité de chaque terme avant de descendre en profondeur.

2-d) De façon à rester fonctionnel, où stocker l'adresse en cours de calcul?

L'adresse en cours de calcul est stockée dans un paramètre supplémentaire, initialisé à (0), pour pouvoir rendre au choix ce paramètre ou nil.

2-e) Dans quel ordre stocker cette adresse? (on suppose que l'on dispose de la fonction **reverse** permettant de la remettre dans l'ordre, le cas échéant)

Pour faciliter les calculs avec des ajouts en tête, l'adresse est stockée à l'envers, et est inversée (à plat) au moment de rendre le résultat.

2-f) Calcul de l'adresse arborescente:

-Comment évolue cette adresse avec le parcours de la liste en largeur?

On incrémente de 1 la tête de l'adresse.

-Comment évolue cette adresse avec le parcours de la liste en profondeur?

On ajoute un 0 en tête de l'adresse.

-En déduire sa valeur initiale.

Valeur initiale: (0)

2-g) Donner le profil et la partition par dichotomies successives.

Profil avec le paramètre supplémentaire: $S_Expr \times List \times ListN \rightarrow ListN$

Partition: -La liste argument est vide.

-La liste argument n'est pas vide:

-Sa tête est égale à l'expression cherchée.

-Sa tête n'est pas égale à l'expression cherchée:

-Sa tête est un atome.

-Sa tête n'est pas un atome.

NB : Le test d'égalité doit précéder le test d'atomicité, pour ne pas rater une liste cherchée.

-Quel est le résultat correspondant à la sortie par échec?

Si la liste argument est vide, **treeaddress** rend (): sortie par échec.

-Quel est le résultat correspondant à la sortie par succès?

Si la liste argument n'est pas vide et sa tête est égale (selon l'égalité entre expressions symboliques) à l'expression cherchée, **treeaddress** rend l'adresse inversée: sortie par succès.

-Quelles sont les deux façons de poursuivre la recherche?

1. Si la liste argument n'est pas vide, sa tête diffère de l'expression cherchée et cette tête est un atome, **treeaddress** est rappelé sur le reste de la liste avec le paramètre adresse dont le premier indice est incrémenté de 1.

2. Si la liste argument n'est pas vide, sa tête diffère de l'expression cherchée et cette tête n'est pas un atome, **treeaddress** rend la disjonction entre:

-**treeaddress** sur la tête de la liste avec le paramètre adresse auquel est ajouté un 0 comme premier indice,

-**treeaddress** sur le reste de la liste avec le paramètre adresse dont le premier indice est incrémenté de 1.

2-h) Donnez la définition formelle de la fonction **reverse** qui rend la liste formée des mêmes termes que son argument, mais pris dans l'ordre inverse (seulement au 1^o niveau).

Voir corrigé d'un TD ultérieur où diverses versions de cette fonction sont étudiées.

Définition Formelle:

treeaddress: $S_Expr \times List \times ListN \rightarrow ListN$

Pour tout (e,l,a) appartenant à $S_Expr \times List \times ListN$:

$l = () \Rightarrow treeaddress\ e,l,a = ()$

$l \# ()$ et $e = tête\ l \Rightarrow treeaddress\ e,l,a = inverse\ a$

$l \# ()$ et $e \# tête\ l$ et $tête\ l$ appartient à $Atom \Rightarrow$

$treeaddress\ e,l,a = treeaddress\ e,reste\ l,ajout\ l+tête\ a,reste\ a$

$l \# ()$ et $e \# tête\ l$ et $tête\ l$ n'appartient pas à $Atom \Rightarrow$

$treeaddress\ e,l,a = \{treeaddress\ e,tête\ l,ajout\ 0,a\} \text{ OU } \{treeaddress\ e,reste\ l,ajout\ l+tête\ a,reste\ a\}$

3) Spécifier et définir formellement la fonction **illustre** qui rend la liste la plus simple composée de parenthèses et, à l'adresse arborescente donnée, d'une astérisque.

3-a) Donner un exemple.

$illustre\ (1\ 3\ 4\ 2) = ((()((()())((()())()((()()*))))))$

3-b) Calcul du résultat:

-Comment évoluent adresse et résultat quand ce dernier garde la même profondeur?
Décrémenter de 1 la tête de l'adresse, et ajouter $()$ en tête du résultat.

-Comment évoluent adresse et résultat quand ce dernier augmente de profondeur?
Passer au reste de l'adresse, et mettre en liste le résultat.

-En déduire quand mettre l'astérisque.
Quand l'adresse est devenue vide.

3-c) Donner le profil et la partition par dichotomies successives.

Profil: $ListNumber \rightarrow List$

Partition: -L'adresse arborescente est vide.

-L'adresse arborescente n'est pas vide:

-La tête de l'adresse est un 0.

-La tête de l'adresse est différente de 0.

-Quel est le résultat correspondant à la sortie?
Quand l'adresse est devenue vide $\Rightarrow *$

-Quelles sont les deux façons de poursuivre la construction du résultat?

1. l'adresse est non vide et sa tête un 0 \Rightarrow

la mise en liste de **illustre** appelé récursivement sur le reste de l'adresse.

2. l'adresse est non vide et sa tête différente de 0 \Rightarrow

ajouter $()$ en tête de **illustre** rappelé récursivement sur l'adresse dont la tête est décrétementée de 1.

Définition Formelle:

illustre: $ListNumber \rightarrow List$

Pour tout aa appartenant à $ListNumber$:

$aa = () \Rightarrow illustre\ aa = *$

$aa \# ()$ et $tête\ aa = 0 \Rightarrow illustre\ aa = mise-en-liste\ illustre\ reste\ aa$

$aa \# ()$ et $tête\ aa \# 0 \Rightarrow illustre\ aa = ajout\ ()\ illustre\ ajout\ tête\ aa - 1, reste\ aa$