Option Informatique
Expressions Algébriques Totalement Parenthésées

Sujet

6 novembre 2006

1 Expressions Algébriques totalement parenthésées

1.1 Definition [Expressions algébriques totalement parenthésées]: Soient C et V deux ensembles où Cest l'ensemble des constantes et V celui des variables. Soit F (resp. O) un ensemble d'applications de C dans C (resp. de $C \times C$ dans C).

On pose $X = C \cup V$ et $A = X \cup F \cup O \cup \{(,)\}$.

On appelle expression algébrique totalement parenthésée (EATP) tout mot sur A défini inductivement par :

- Tout élément de X est une EATP.
- si e est une EATP et f un élément de F alors fe est une EATP.
- si e et e' sont deux EATP et c est un élément de O alors (ece') est une EATP.

1.2 Example: On choisit :

- $C = \mathbb{R}$ l'enemble des réels.
- $V = \{x, y, z\}$ où x, y, z, sont les noms de trois variables.
- $O = \{+, *, \}$
- $F = \{-, \cos, \sin, \ln, \exp\}$

Les expressions suivantes sont alors totalement parenthésées :

- $e_1 = \cos(x + 2)$
- $e_2 = \ln(y + cos(x + 2))$
- $e_3 = -\cos(x + 2)$
- $e_4 = \exp(3 * y - \cos(X + 2))$
- $e_5 = (\exp(3 * y - \cos(X + 2)) + \sin(2 * x))$

1.3 Definition [Arbre associé à une EATP]: Soit e une EATP. On associe un arbre binaire $T(e)$ à e défini par :

- Si $e = x$ élément de X alors $T(e)$ est une feuille étiquetée par x.
- Si $e = fa$ alors l'arbre $T(e)$ admet f pour racine et $T(a)$ pour unique fils.
- Si $e = (acb)$ alors $T(e)$ admect c pour racine, $T(a)$ pour fils gauche et $T(b)$ pour fils droit.

1.4 Remark: Les arbres manipulés ici sont définis inductivement par :

- T est réduit à une feuille étiquetée par $X \in X$.

– T est un couple (f, T') où T' est un arbre et $f \in F$.
– T et un triplet (G, o, D) o* G, D sont des arbres et o un élément de O.

1.5 Example: Donner les arbres $t_1 \dots t_5$ associés aux expressions $e_1 \dots e_5$.

2 Implémentation en Caml des expressions réelles

Définition des expressions de l'exemple On supposera qu'une EATP est une chaîne de caractères. Les éléments sont séparés par un espace et la chaîne **se termine par un espace**. Définir les expressions $e_1 \dots e_5$. **Attention :** l'entier n est représenté par le réel 0.0.

Définition du type arbre On définit le type arbre par :
`type arbre = f of string | n1 of string * arbre | n2 of arbre * string * arbre ; ;` Définir le type arbre en Caml.

Extraction du premier élément de la chaîne et de la chaîne retante Écrire une fonction caml `(extrait s)` telle que si `s` représente l'expression ss' où x est dans A alors `(extrait s)` renvoie le couple (x, s').

2.1 Remark: – x et s' sont séparés par un blanc dans `s`.
– `(extrait "cos ( x +. 2.0 ) "` renvoie le couple `("cos","( x +. 2.0 ) \"`
– le type de la fonction doit être `string → (string * string)`
– Si le premier élément de s est blanc, la fonction renvoie le couple `("",s')` où "" est la chaîne vide.

```
let res extrait s =

let premier =                          and suite =
in
match premier with
" " →                                                ;
|_ → let deb,suite1 = extrait                         in
;;
```

Identification des objets *Cas particulier : expressions réelles avec constantes réelles* mÉcrire les fonctions booléennes `est_constante s`, `est_variable s`, `est_fonction s`, `est_operateur s` de type `string → bool`.

```
let est_constante s= ((string_of_float(float_of_string s))=s) ; ;

let est_variable s =

let est_fonction s =

let est_operateur s =
```

3 Algorithmes de transformation d'une EATP en arbre

Principe de l'algorithme Soit s une expression pas nécessairement totalement parenthésée.
On veut écrire une fonction `(analyse_eatp s)` dont le résultat est un couple `(t,suite)` o* `t` est un arbre et `suite` est une chaîne de caractères tels que $s = s'$`suite` où s' est le plus long préfixe de s qui soit une EATP et `t` est l'arbre associé à s'.

Analyse d'une EATP Compléter le programme suivant et le valider en Caml :

```
let rec analyse_eatp = function s →
  let premier,reste =             in
(* premier = premier élément de s qui est dans A et reste est tq :*)
(* s = premier reste *)
  if (est_constante premier) or (est_variable premier) then

(* si premier est dans X alors s' = premier et suite = reste*)
  else
    if (est_fonction premier) then
(* si premier est une fonction, on calcule le plus long préfixe *)
(* s" de reste qui est une EATP et alors s' = premier s"*)
      let arbre, reste1=                in
```

```
    else
      if premier = "(" then
(* si premier est une "(", on calcule le plus long préfixe s'' de s *)
(* qui est une EATP et alors s' = premier s'' op s'''. sinon une *)
(* interruption s'est produite : s' n'existe pas *)
        let arbre_g,reste1 =                    in
        let premier1,reste2 = (extrait reste1) in
        if (est_operateur premier1) then
          let arbre_d,reste3 = (analyse_eatp reste2) in
          let premier3,reste4 = (extrait reste3) in
          if premier3 = ")" then
            (n2 (arbre_g,premier1,arbre_d), reste4)
          else
            failwith("s' n'existe pas")
        else
          failwith ("s' n'existe pas")
      else
        failwith("expression incorrecte");;
```

Transformation d'une EATP en arbre Ecrire une fonction `(eatp2arbre s)` qui renvoie l'arbre associé à `s` si `s` est une EATP et génère une interruption sinon.

```
let eatp2arbre s =
match analyse_eatp s with
    (arbre, "") →             ;
  |_→                      ;;
```

Tests sur des exemples Fabriquer les arbres `t1 ...t5`.

4 Transformation d'un arbre en EATP

Écrire une fonction `(arbre2eatp t)` qui renvoie la chaîne de caractères représentant l'EATP associée à l'arbre `t`.

```
let rec arbre2eatp = function t →
match t with
    f x → x^" ";
  |n1 (fonc,fils) → fonc^" ( "^arbre2eatp(fils)^" ) ";
  |n2 (fg,o,fd) → arbre2eatp(fg)^" "^o^" "^arbre2eatp(fd);;
```

Tester cette fonction avec les arbres `t1 ... t5`.

5 Transformation d'une EATP sous forme postfixée ou préfixée

Écrire une fonction `(prefixe s)` qui renvoie l'expresion préfixée équivalente à `s`. Même question poru la forme postfixée.

```
let prefixe s = parcours_pref (          )
where rec parcours_pref arbre =
  match arbre with
      f x →
    |n1 (fonc,fils) →
    |n2 (fg,o,fd) →
;;


let postfixe s = parcours_postf (eatp2arbre s)
```

```
where rec parcours_postf arbre =
  match arbre with
      f x →
    |n1 (fonc,fils) →
    |n2 (fg,o,fd) →
;;
```

6 Évaluation d'une EATP

6.1 Definition [Sémantique d'une EATP]: Soit φ une application de V dans C. On dit que φ est un environnement.

On définit une application F de l'ensemble des EATP dans C par :
- Si $e = x$, $F(e) = x$ si x est une constante ou $\varphi(x)$ si c'est une variable.
- Si $e = fa$, $F(e) = f(F(a))$
- Si $e = (acb)$, $F(e) = c(F(a), F(b))$

On peut calculer récursivement $F(c)$ à l'aide de l'arbre $T(c)$.

6.2 Example: Définir en Caml l'environnement $\texttt{phi}_{exp}$ pour lequel $\varphi(x) = 3.2$ et $\varphi(y) = 0.0$.

Évaluation des fonctions et opérateurs Écrire la fonction `(eval_fonc g x)` qui renvoie la valeur au point réel x de la fonction représentée par la chaîne g.

Écrire la fonction `(eval_op o x y)` qui renvoie la valeur x op y si la chaîne `o` représente l'opérateur op.

```
let eval_fonc g x = match g with
    "-" →
  | "cos" →
  |"sin" →
  | "ln" →
  |"exp" →
;;

let eval_op o x y  = match o with
    "+" →
  |"*" →
  |"/" →
;;
```

Programme Écrire la fonction `(eval s phi)` qui renvoie $F(s)$ pour l'environnement `phi`.

```
let eval s phi = eval_arbre (               ) phi
  where rec eval_arbre t phi =
  match t with
      f x →
    | n1 (fonc, fils) →
    | n2 (fg,o,fd) →
;;
```