Option Informatique
Expressions Algébriques totalement parenthésées

Corrigé

6 novembre 2006

1 Expressions Algébriques Totalement Parenthésées

1.1 Example: Voir les figures 1,2,3,4,5.

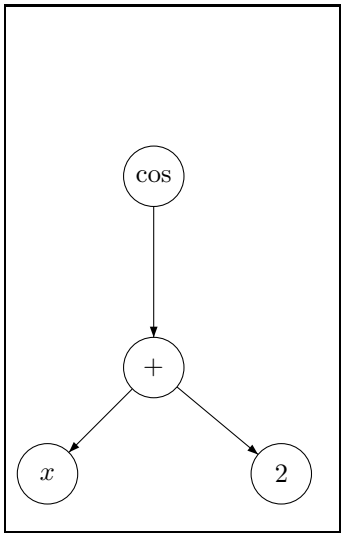

FIG. 1 – t_1

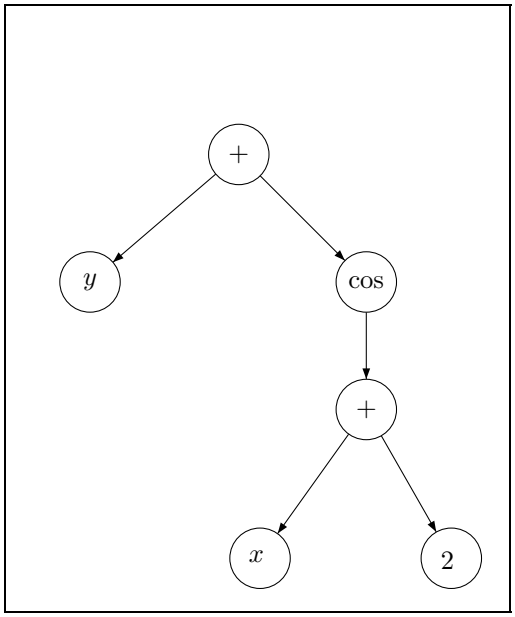

FIG. 2 – t_2

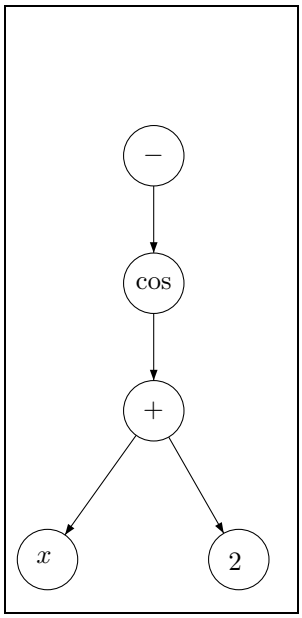

FIG. 3 – t_3

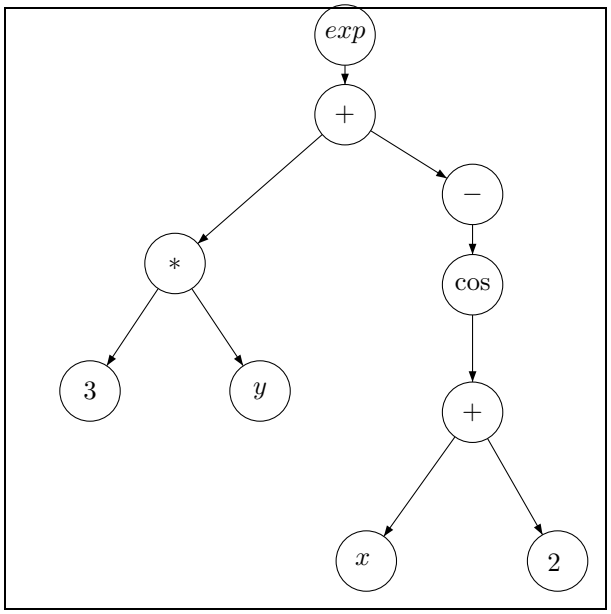

FIG. 4 – t_4

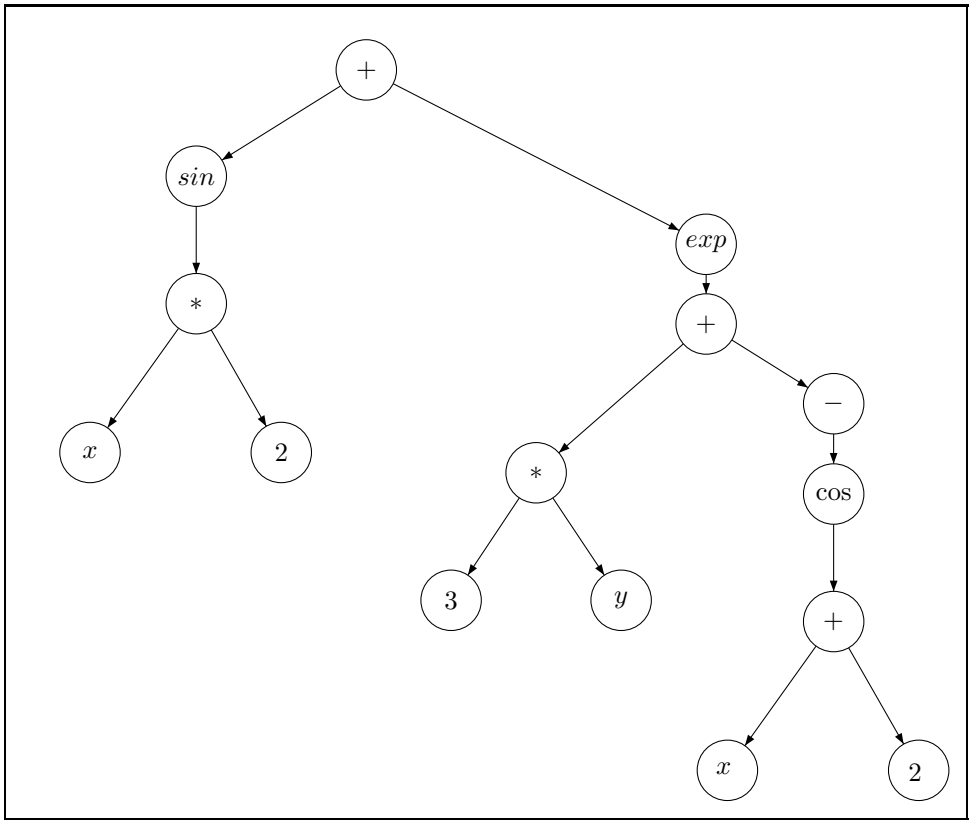

FIG. 5 – t_5

2 Implémentation en Caml des expressions réelles

Définition des expressions de l'exemple

```
# let e1 = "cos ( x + 2.0 ) ";;
val e1 : string = "cos ( x + 2.0 ) "
# let e2 = "ln ( y + cos ( x + 2.0 ) ) ";;
val e2 : string = "ln ( y + cos ( x + 2.0 ) ) "
# let e3 = "- cos ( x + 2.0 ) ";;
val e3 : string = "- cos ( x + 2.0 ) "
# let e4 = "exp ( ( 3.0 * y ) + - cos ( x + 2.0 ) ) ";;
val e4 : string = "exp ( ( 3.0 * y ) + - cos ( x + 2.0 ) ) "
# let e5 = "( exp ( ( 3.0 * y ) + - cos ( x + 2.0 ) ) + sin ( 2.0 * x ) ) ";;
val e5 : string =
  "( exp ( ( 3.0 * y ) + - cos ( x + 2.0 ) ) + sin ( 2.0 * x ) ) "
```

Définition du type arbre

```
type arbre = f of string | n1 of string * arbre | n2 of arbre * string * arbre;;
```

Extraction du premier élément de la chaîne et de la chaîne retante

```
let rec extrait s =
  let premier = sub_string s 0 1 and
      suite = sub_string s 1 ((string_length s)-1) in
  match premier with
      " "→ ("",suite);
      |_ → let deb,suite1 = extrait suite in (premier^deb,suite1);;
```

Identification des objets

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

let est_variable s = (s = "x" or s= "y" or s="z");;

let est_fonction s = (s = "-" or s = "cos" or s= "sin"
                      or s = "exp" or s="ln");;

let est_operateur s = (s="+" or s="*" or s ="/");;
```

3 Algorithmes de transformation d'une EATP en arbre

Analyse d'une EATP

```
let rec analyse_eatp = function s →
  let premier,reste = (extrait s) in
  if (est_constante premier) or (est_variable premier) then
    ((f premier), reste)
  else
    if (est_fonction premier) then
      let arbre, reste1=(analyse_eatp reste) in
      (n1 (premier,arbre)),reste1;
    else
      if premier = "(" then
        let arbre_g,reste1 = (analyse_eatp reste) 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

```
let eatp2arbre s =
  match analyse_eatp s with
      (arbre, "") → arbre ;
    |_→ failwith ("ceci n'est pas une eatp") ; ;
```

Tests sur des exemples

```
let t1 = eatp2arbre e1 ; ;
let t2 = eatp2arbre e2 ; ;
let t3 = eatp2arbre e3 ; ;
let t4 = eatp2arbre e4 ; ;
let t5 = eatp2arbre e5 ; ;
```

4 Transformation d'un arbre en EATP

```
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) ; ;

#arbre2eatp(t1) ; ;
- : string = "cos ( x  + 2.0  ) "
#arbre2eatp(t2) ; ;
- : string = "ln ( y  + cos ( x  + 2.0  )  ) "
#arbre2eatp(t3) ; ;
- : string = "- ( cos ( x  + 2.0  )  ) "
#arbre2eatp(t4) ; ;
- : string = "exp ( 3.0  * y  + - ( cos ( x  + 2.0  )  )  ) "
#arbre2eatp(t5) ; ;
- : string =
 "exp ( 3.0  * y  + - ( cos ( x  + 2.0  )  )  )  + sin ( 2.0  * x  ) "
```

5 Transformation d'une EATP sous forme postfixée ou préfixée

```
let prefixe s = parcours_pref (eatp2arbre s)
where rec parcours_pref arbre =
  match arbre with
      f x → x^" " ;
    |n1 (fonc,fils) → fonc ^" "^(parcours_pref fils) ;
    |n2 (fg,o,fd) → "( "^o^" "^(parcours_pref fg)^(parcours_pref fd)^") " ; ;

let postfixe s = parcours_postf (eatp2arbre s)
where rec parcours_postf arbre =
```

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

6 Évaluation d'une EATP

6.1 Example:
```
let phi_exp = function
    "x" → 3.2;
  | "y" → 0.0;
  | _ → failwith("non défini");;
```

Évaluation des fonctions et opérateurs

```
let eval_fonc g x = match g with
    "-" → -1. *. (x);
  | "cos" → cos(x);
  |"sin" → sin(x);
  | "ln" → ln(x);
  |"exp" → exp(x)
;;

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

Programme

```
let eval s phi = eval_arbre (eatp2arbre s) phi
  where rec eval_arbre t phi =
  match t with
      f x → (phi x);
    | n1 (fonc, fils) → eval_fonc fonc (eval_arbre fils phi);
    | n2 (fg,o,fd) → eval_op o (eval_arbre fg phi) (eval_arbre fd phi)
;;
```