PARADIGMES ET LANGAGES DE PROGRAMMATION

Rapport

Projet

Christen & Mottier

1. Objectif

Produire un programme Haskell exécutable permettant de lire des expressions et des définitions de fonction et de les interpréter.

2. Fonctionnalités

Notre langage de programmation propose :

- Des opérations arithmétiques habituels et originales,
- Des opérations relationnelles (comparaisons),
- La notion de variable et des énoncés de type « let »,
- La notion de fonction (dont les fonctions « factorielle » et « Fibonacci »)

3. Résultats du calcul des fonctions

Exemple de nos fonctions :

- Factorielle :

```
SPL>f_fact(5)
[TFct "f_fact",TLeftParenthesis,TInt 5,TRightParenthesis]
Exp (App "f_fact" [Cst 5])
120
```

- Fibonacci :

```
SPL>f_fibo(6)
[TFct "f_fibo",TLeftParenthesis,TInt 6,TRightParenthesis]
Exp (App "f_fibo" [Cst 6])
13
```

- Définition et utilisation d'une fonction :

```
SPL>def f_max(v_x,v_y) = if v_x > v_y then v_x else v_y
[TDef,TFct "f_max",TLeftParenthesis,TVar "v_x",TSym ",",TVar "v_y",TRightP
Def (DefFunc "f_max" ["v_x","v_y"] (If (Bin ">" (Var "v_x") (Var "v_y")) (
"Environnement expanded"
SPL>f_max(3,10)
[TFct "f_max",TLeftParenthesis,TInt 3,TSym ",",TInt 10,TRightParenthesis]
Exp (App "f_max" [Cst 3,Cst 10])
10
```

4. Résultats intermédiaires

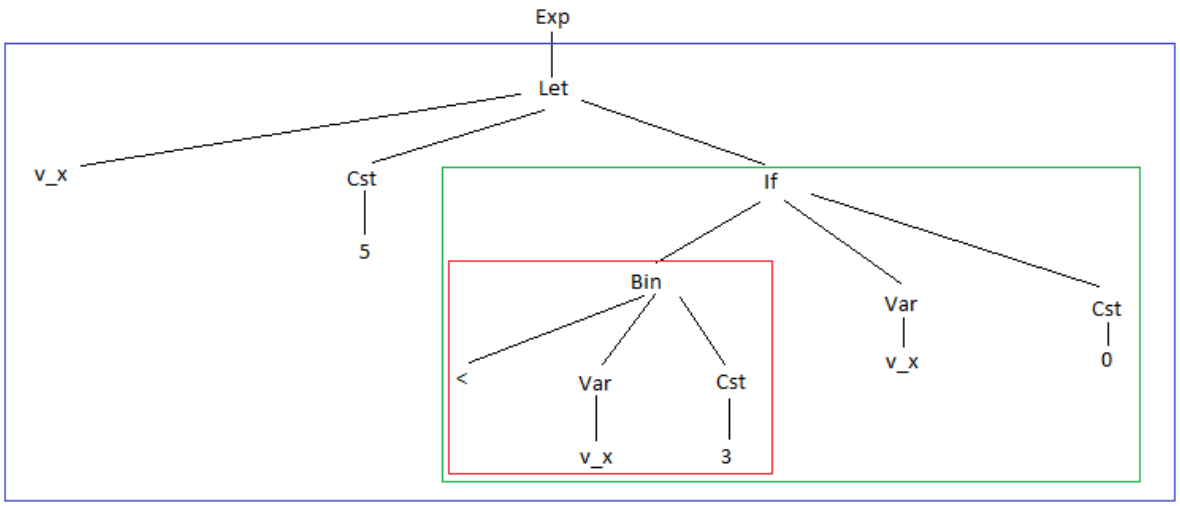
Pour la ligne « let v_x = 5 in if v_x < 3 then v_x else 0 », nous avons :

- Liste des lexèmes :

TLet	TVar « v_x »	TSym « = »	TInt 5	TIn	TIf	TVar « v_x »	TComp « < »	TInt 3	TThen	TVar « v_x »	TElse	TInt 0

- Arbre syntaxique :

```
Exp (Let « v_x » (Cst 5) (If (Bin « < » (Var « v_x ») (Cst 3)) (Var « v_x ») (Cst 0)))
```



- Résultat de l'expression : **0**

```
SPL>let v_x = 5 in if v_x < 3 then v_x else 0
[TLet,TVar "v_x",TSym "=",TInt 5,TIn,TIf,TVar "v_x",TComp "<",TInt 3,TThen,TVar "v_x",TElse,TInt 0]
Exp (Let "v_x" (Cst 5) (If (Bin "<" (Var "v_x") (Cst 3)) (Var "v_x") (Cst 0)))
0
```

i - Exemple de résultat d'un "let"