Université Paris Cité
L3 Informatique

Machines Virtuelles
Année 2023-2024

TP n°3

Compilation de Myrthe avec les variables

Dans ce TP, vous utiliserez les fichiers `myrtheAST.ml`, `myrthe.ml`, `compile.ml` et `mv.ml` fournis.

1 Arbre de syntaxe

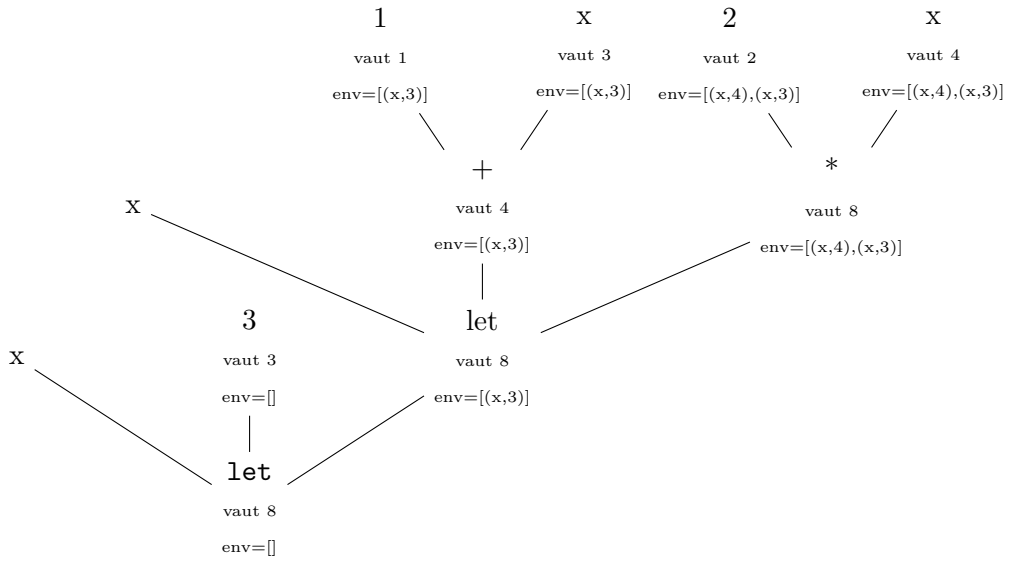
Exercice 1. Pour chacune des expressions suivantes dessinez les arbres de syntaxes en annotant pour chaque sous-expression sa valeur et son environnement comme définis par la fonction `interpr` de `myrthe.ml`.

- (((2=3) && false) || (3+4 = 1+6))
- let x = 3 in let y = true in if y then x else x + 1
- let x = 3 <= 4 in if (if x then false else true) then 3 else 4

Par exemple, l'expression

$$\text{let x = 3 in let x = 1 + x in (2*x)}$$

donne l'arbre de syntaxe annoté comme suit



Ensuite écrivez les valeurs de type `expression` correspondantes dans le fichier `myrtheAST.ml`. Dans chaque cas, quel est l'opérateur à la racine ?

2 Compilation des expressions de Myrthe (sans variables)

Exercice 2. En utilisant `compile` de `compile.ml`, pouvez-vous trouver à quelles expressions de Myrte correspondent chaque liste d'instructions de la MV ?

```
0 - Consti 3
1 - Push
2 - Consti 4
3 - Addi
4 - Push
5 - Consti 5
6 - Addi
```

```
0 - Consti 3
1 - Push
2 - Consti 4
3 - Push
4 - Consti 5
5 - Addi
6 - Addi
```

```
0 - Consti 12
1 - Push
2 - Consti 3
3 - Push
4 - Consti 4
5 - Addi
6 - Addi
7 - Push
8 - Consti 32
9 - Push
10 - Consti 3
11 - Addi
12 - Eqi
```

```
0 - Consti 1
1 - Branchif 6
2 - Consti 3
3 - Push
4 - Consti -1
5 - Addi
6 - Branch 5
7 - Consti 3
8 - Push
9 - Consti 1
10 - Addi
```

Bonus

```
0 - Consti 0
1 - Branchif 3
2 - Consti 2
3 - Branch 2
4 - Consti 3
5 - Push
6 - Consti 2
7 - Eqi
8 - Branchif 13
9 - Consti 2
10 - Push
11 - Consti 3
12 - Eqi
13 - Branchif 6
14 - Consti 1
15 - Push
16 - Consti 2
17 - Addi
18 - Branch 2
19 - Consti 7
20 - Branch 6
21 - Consti 1
22 - Branchif 3
23 - Consti 10
24 - Branch 2
25 - Consti 0
```

Évaluez-les avec la fonction `exec_big` de `mv.ml` : le comportement est-il celui attendu ?

Exercice 3. Complétez la fonction `compile` dans le fichier `compile.ml` pour qu'elle traite tous les constructeurs de Myrthe sauf les variables et leur déclarations.

3 Compilation des expressions de Myrthe (avec variables)

Comment compiler les expressions avec variables ? L'idée est de fournir une fonction auxiliaire à `compile` qui prend en paramètre un environnement, c'est-à-dire une liste d'association. Les clés de cette liste seront les identificateurs des variables, mais quelles seront les valeurs ? Ils ne peuvent pas être des valeurs des expressions, comme dans `interpr`, car un compilateur ne doit pas évaluer une expression !

— Où se trouveront les valeurs des variables définies localement pendant l'exécution du code compilé ?
— Donc, quelles seront les valeurs de la liste d'association environnement de la compilation ?

Exercice 4. L'environnement de la compilation est de type `(var_id*int) list` : une association `(k,v)` indiquant que la valeur de la variable `k` se trouvera sur la pile de la MV en position `v`.

Utiliser un tel environnement pour modifier la fonction `compile` du fichier `compile.ml` afin qu'elle traite aussi la lecture et la déclaration des variables de Myrthe. [1]

Exercice 5. Testez votre compilateur sur les expressions de l'exercice 1. Est-ce que les résultats calculés par MV sur le code compilé correspondent bien à la valeur de l'expression de Myrthe donnée par `interpr` ? Si non, trouvez l'erreur, corrigez-le. Ensuite, testez le code de nouveau.
Attention : à la fin de l'exécution du code compilé, le résultat doit se trouver sur l'accumulateur et la pile doit être vide.
Testez aussi votre compilateur sur l'expression suivante :

$$\texttt{let x} = \left(\texttt{let x} = \texttt{1 in x} + \texttt{3} \right) \texttt{ in}$$
$$\left(\left(\texttt{let x} = \left(\begin{array}{l} \texttt{let x} = \texttt{x} + \texttt{4} \\ \texttt{in x} + \texttt{5} \end{array} \right) \texttt{ in x} + \texttt{2} \right) + \texttt{x} \right)$$

1. *Suggestion :*

(Ne lisez pas cette suggestion tout de suite... essayez de résoudre l'exercice vous même sans cette aide !.

Étant donné un environnement `env` et une expression `e` de Myrthe :

- si `e` est une lecture de variable, i.e. `Var "x"`, le code compilé devra mettre dans l'accumulateur la valeur de la variable `"x"` qui se trouve sur la pile de la MV (la position de cette valeur est à chercher dans l'environnement) ;
- si `e` est une déclaration locale, i.e. `Let ("x",e1,e2)`, le code compilé consistera en le code compilé de `e1`, ensuite un `Push` pour mettre la valeur de l'accumulateur sur la pile, ensuite le code compilé de l'expression `e2` sous l'environnent mis à jour par rapport aux positions des valeurs des variables dans l'environnement (attention ! le nouveau environnement aura une nouvelle association `("x",0)`, mais aussi les associations déjà présents dans `env` devront être mis à jour, car on a changé l'état de la pile) et enfin l'instruction permettant de dépiler l'élément de la pile associé à la variable `"x"` qui ne doit plus être visible en dehors de `e2`.

Attention : est-ce qu'il faut modifier les cas de `compile` associés aux autres constructeurs de Myrthe ?