Sémantique des Langages de Programmation (SemLP)
TD n° 5 : Reduction Strategies

Exercice 1 :

Recall the dynamic call-by-name λ-calculus with closures given in class.

$$\frac{}{v \ \Downarrow v} \qquad \frac{\eta(x)[\eta] \Downarrow v}{x[\eta] \ \Downarrow v} \qquad \frac{M[\eta] \Downarrow_n \lambda x.M_1[\eta'] \qquad M_1[\eta'[M'[\eta]/x]] \Downarrow_n v}{(MM')[\eta] \Downarrow_n v}$$

Define a similar big-step semantics reduction rule for the call-by-value λ-calculus.

Exercice 2 :

Recall the abstract machine for the call-by-name λ-calculus using a strack. Here s is a stack of closures.

$$\begin{array}{rcl} (x[\eta], s) & \rightarrow & (\eta(x), s) \\ ((MM')[\eta], s) & \rightarrow & (M[\eta], M'[\eta] : s) \\ ((\lambda x.M)[\eta], c : s) & \rightarrow & (M[\eta[c/x]], s) \end{array}$$

Define a similar stack-based strategy to evaluate the call-by-name λ-calculus. Importantly, since in call-by-name the argument is evaluated before the substitution, you will have to store the functional in the stack while evaluating the arguments.
Hint : use markers to indicate whether the element being added to the stack is the functional or an argument.

Exercice 3 :

Assume the abstract machine for the call-by-value λ-calculus of exercise 2where we add a unary operator f and a binary operator g.

1. What are the new evaluation contexts ?

2. How is the abstract machine to be modified ?

Exercice 4 :

Define (and implement) the abstract machine for call-by-value using De Brujin variables.

Exercice 5 :

Suppose we add to the call-by-name λ-calculus two monadic operators : C for *control* and A for *abort*. If M is a term then CM and AM are λ-terms. An evaluation context E is always defined as : $E ::= [\ ] \mid EM$, and the reduction of the control and abort operators is governed by the following rules :

$$E[CM] \ \rightarrow \ M(\lambda x.AE[x]) \ , \qquad E[AM] \ \rightarrow \ M$$

Adapting the rules of Table 15 of the lecture notes, design an abstract machine to execute the terms in this extended language (a similar exercise can be carried on for call-by-value). *Hint :* assume an operator *ret* which takes a *whole stack* and *retracts* it into a closure ; then, for instance, the rule for the control operator can be formulated as :

$$((CM)[\eta], s) \ \rightarrow \ (M[\eta], ret(s))$$