# Sémantique des Langages de Programmation (SemLP)
# Projet : A Machine for CBPV

Le projet est à rendre sur Moodle et à soutenir le jeudi 23 mai. La soutenance prendra la forme de 15 minutes de présentation avec démonstration du code et explication d'une preuve de simulation.

| | |
|---|---|
| Positive types | $\varphi, \psi := \iota \mid !\sigma$ |
| General types | $\sigma, \tau := \varphi \mid \varphi \multimap \sigma$ |

(a) Types of $\Lambda_{\mathsf{HP}}$

$$M, N := x \mid \underline{n} \mid M^! \mid \mathsf{der}(M) \mid \mathsf{succ}(M) \mid \lambda x^\varphi\, M \mid \langle M \rangle N \mid \mathsf{fix}\, x^{!\sigma}\, M \\ \mid \mathsf{if}(M, N, [z]P)$$

(b) Terms of $\Lambda_{\mathsf{HP}}$

$$\frac{}{\mathcal{P} \vdash \underline{n} : \iota} \qquad \frac{}{\mathcal{P}, x : \varphi \vdash x : \varphi} \qquad \frac{\mathcal{P} \vdash M : \sigma}{\mathcal{P} \vdash M^! : !\sigma} \qquad \frac{\mathcal{P} \vdash M : !\sigma}{\mathcal{P} \vdash \mathsf{der}(M) : \sigma}$$

$$\frac{\mathcal{P}, x : \varphi \vdash M : \sigma}{\mathcal{P} \vdash \lambda x^\varphi\, M : \varphi \multimap \sigma} \qquad \frac{\mathcal{P} \vdash M : \varphi \multimap \sigma \qquad \mathcal{P} \vdash N : \varphi}{\mathcal{P} \vdash \langle M \rangle N : \sigma} \qquad \frac{\mathcal{P}, x : !\sigma \vdash M : \sigma}{\mathcal{P} \vdash \mathsf{fix}\, x^{!\sigma}\, M : \sigma}$$

$$\frac{\mathcal{P} \vdash M : \iota}{\mathcal{P} \vdash \mathsf{succ}(M) : \iota} \qquad \frac{\mathcal{P} \vdash M : \iota \qquad \mathcal{P} \vdash N : \sigma \qquad \mathcal{P}, z : \iota \vdash P : \sigma}{\mathcal{P} \vdash \mathsf{if}(M, N, [z]P) : \sigma}$$

A typing context is an expression $\mathcal{P} = (x_1 : \varphi_1, \dots, x_k : \varphi_k)$ where all types are positive and the $x_i$s are pairwise distinct variables.

(c) Typing system of $\Lambda_{\mathsf{HP}}$.

FIGURE 1 – Syntax of $\Lambda_{\mathsf{HP}}$.

*Values* are particular $\Lambda_{\mathsf{HP}}$ terms (they are not a new syntactic category) defined in Figure 2a. It is easy to check that they are all typed with positive types.
Figure 2 defines a deterministic *weak* reduction relation $\to_{\mathsf{w}}$. This reduction is weak in the sense that we never reduce within a "box" $M^!$ or under a $\lambda$.
The distinguishing feature of this reduction system is the role played by values in the definition of $\to_{\mathsf{w}}$. Consider for instance the case of $\mathsf{if}$, the term on which the test is made must be reduced to a value (necessarily of shape $\underline{0}$ or $\underline{n+1}$ if the expression is well typed) before the reduction is performed. This allows to "memoize" the value $\underline{n}$ for further usage : the value is passed to the relevant branch of the $\mathsf{if}$ through the variable $z$.
We say that $M$ is *weak normal* if there is no reduction $M \to_{\mathsf{w}} M'$. It is clear that any value is weak normal. When $M$ is closed, $M$ is weak normal iff it is a value or an abstraction.

$$V := x \mid \underline{n} \mid M^!.$$

(a) Values of $\Lambda_{\mathsf{HP}}$

$$\frac{}{\mathsf{der}(M^!) \to_{\mathsf{w}} M} \qquad \frac{}{\langle \lambda x^{\varphi}\, M\rangle V \to_{\mathsf{w}} M\,[V/x]} \qquad \frac{}{\mathsf{fix}\, x^{!\sigma}\, M \to_{\mathsf{w}} M\,\left[(\mathsf{fix}\, x^{!\sigma}\, M)^!/x\right]}$$

$$\frac{}{\mathsf{succ}(\underline{n}) \to_{\mathsf{w}} \underline{n+1}} \qquad \frac{}{\mathsf{if}(\underline{0}, N, [z]P) \to_{\mathsf{w}} N} \qquad \frac{}{\mathsf{if}(\underline{n+1}, N, [z]P) \to_{\mathsf{w}} P\,[\underline{n}/z]}$$

(b) Deterministic one-step reduction $\to_{\mathsf{w}}$

$$E := \mathsf{der}(E[\ ]) \mid \langle E[\ ]\rangle V \mid \langle M\rangle E[\ ] \mid \mathsf{succ}(E[\ ]) \mid \mathsf{if}(E[\ ], N, [z]P)$$

$$E[M] \to_{\mathsf{w}} E[N], \text{ whenever } M \to_{\mathsf{w}} N$$

(c) Evaluation contexts and context closure of reduction $\xrightarrow{p}$.

FIGURE 2 – Operational semantics of $\Lambda_{\mathsf{HP}}$

**Exercice 1 :**

In this exercise, we consider $\Lambda_{\mathsf{HP}}$ without fixpoints of terms.

**1.** Write an Abstract Machine without environment that simulates the evaluation of $\Lambda_{\mathsf{HP}}$.

Stack Language : $K := M \mid \varphi \mid \mathsf{fun} \mid \mathsf{arg} \mid \mathsf{der} \mid \mathsf{if} \mid \mathsf{S}$ and $\pi := [\,] \mid K \cdot \pi$

Reduction : $(M, \pi) \to_k (M', \pi')$

$$\begin{array}{rcl} (\langle M\rangle N, \pi) & \to_k & (M, \mathsf{arg} \cdot N \cdot \pi) \\ (\lambda x^{\varphi}\, M, \mathsf{arg} \cdot N \cdot \pi) & \to_k & (N, \mathsf{fun} \cdot x \cdot \varphi \cdot M \cdot \pi) \\ (V, \mathsf{fun} \cdot x \cdot \varphi \cdot M \cdot \pi) & \to_k & (M\,[V/x]\,, \pi) \\ & \dots & \end{array}$$

Implement this Abstract Machine.

**2.** Prove that the reduction terminates.

**3.** Give a translation $*$ from States of the Abstract Machine to $\Lambda_{\mathsf{HP}}$ such that :

- If $(M, \pi) \to_k^* (V, [\,])$, then $(M, \pi)^* = V$.
- If $M \to_{\mathsf{w}} M'$, then $(M, [\,]) \to_k^* (M', [\,])$.

For instance,

$$\begin{array}{rcl} (M, \mathsf{arg} \cdot N \cdot \pi)^* & = & (\langle M\rangle N, \pi)^* \\ (V, \mathsf{fun} \cdot x \cdot M \cdot \pi)^* & = & (M\,[V/x]\,, \pi)^* \\ & \dots & \end{array}$$

Prove that the translation is well defined and satisfies the wanted properties.

**4.** Define a typing systems for stacks such that the translation $*$ is compatible with types, that is :
- If $\vdash M : \sigma$ and $\sigma \vdash \pi : \tau$ then $\vdash (M, \pi) : \tau$.
- If $\vdash (M, \pi) : \sigma$ and $(M, \pi) \rightarrow_k (M', \pi')$ then $\vdash (M', \pi') : \sigma$.
- If $\vdash (M, \pi) : \sigma$ then $\vdash (M, \pi)^* : \sigma$.

For instance,

$$\frac{\varphi \vdash \pi : \tau \qquad \vdash N : \varphi}{\varphi \multimap \sigma \vdash \mathsf{arg} \cdot N \cdot \pi : \tau} \qquad \frac{\sigma \vdash \pi : \tau \qquad x : \varphi \vdash N : \sigma}{\varphi \vdash \mathsf{fun} \cdot x \cdot N \cdot \pi : \tau}$$

**5.** Give a compilation $\mathcal{C}$ of CBV into $\Lambda_{\mathsf{HP}}$ which is compatible with the reductions. $\mathcal{C} : \Lambda_v \rightarrow \Lambda_{\mathsf{HP}}$ is defined on types and terms such that :
- If $\Gamma \vdash M : A$, then $\mathcal{C}(\Gamma) \vdash \mathcal{C}(M) : \mathcal{C}(A)$
- If $\Gamma \vdash M : A \Rightarrow B$, then $\mathcal{C}(\Gamma) \vdash \mathcal{C}(M) :!(\mathcal{C}(A) \multimap \mathcal{C}(B))$
- $\mathcal{C}((M)N) = (\langle \mathsf{der}(\mathcal{C}(M)) \rangle \mathcal{C}(N))$

Implement this compilation and prove the simulation theorem.

**6.** Give a compilation $\mathcal{D}$ of CBN into $\Lambda_{\mathsf{HP}}$ which is compatible with the reductions. $\mathcal{C} : \Lambda_n \rightarrow \Lambda_{\mathsf{HP}}$ is defined on types and terms such that :
- If $\Gamma \vdash M : A$, then $!\mathcal{D}(\Gamma) \vdash \mathcal{D}(M) : \mathcal{D}(A)$
- If $\Gamma \vdash M : A \Rightarrow B$, then $!\mathcal{D}(\Gamma) \vdash \mathcal{D}(M) : !\mathcal{D}(A) \multimap \mathcal{D}(B))$
- $\mathcal{D}((M)N) = \langle \mathcal{D}(M) \rangle \mathcal{D}(N)^!$

Implement this compilation and prove the simulation theorem.

# Références

[1] Thomas Ehrhard. Call-by-push-value from a linear logic point of view. In *ESOP*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228. Springer, 2016.

[2] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3) :199–207, 2007.

[3] Frédéric Lang. Explaining the lazy krivine machine using explicit substitution and addresses. *Higher-Order and Symbolic Computation*, 20(3) :257–270, 2007.

[4] Mitchell Wand. On the correctness of the krivine machine. *Higher-Order and Symbolic Computation*, 20(3) :231–235, 2007.