

# Projet du cours « Compilation »

## Jalon 2 : Interprétation et typage de HOPIX

### (VERSION PRÉLIMINAIRE)

version numéro Thu Dec 3 15 :32 :48 CET 2015

## 1 Syntaxe abstraite

Dans les sections suivantes, lorsque l'on utilisera de la syntaxe concrète dans les spécifications, on fera référence implicitement aux arbres de syntaxe abstraite sous-jacents définis par la grammaire suivante :

p	:=	$\bar{d}$	Programme
d	:=	type T $\bar{\alpha} := d_t$	Définition de type
		extern x : $\tau$	Déclaration d'une valeur externe
		$d_v$	Définition de valeur(s)
d <sub>t</sub>	:=	{ $\ell : \tau$ }	Type d'enregistrement
		{ $K : \bar{\tau}$ }	Type somme
		•	Type abstrait
d <sub>v</sub>	:=	val x $\bar{m} : \tau^? := e$	Valeur simple
		rec x $\bar{m} : \tau^? := e$	Valeurs mutuellement récursives
$\tau$	:=	T[ $\bar{\tau}$ ]	Type construit
		$\alpha$	Variable de type
e	:=	n	Entier
		c	Caractère
		s	Chaîne de caractères
		x	Variable
		K( $\bar{e}$ )	Construction d'une donnée étiquetée
		{ $\ell := e$ }	Construction d'un enregistrement
		(e : $\tau$ )	Annotation de type
		$d_v ; e$	Définition locale
		ee	Application
		\m => e	Fonction anonyme
		e# $\ell$	Accès à un champ
		e# $\ell <- e$	Modification d'un champ
		e ? $\bar{b}$	Analyse de motifs
		if e then e else e fi	Conditionnelle
b	:=	m => e	Cas d'analyse
m	:=	K	Étiquette
		x	Motif universel liant
		-	Motif universel non liant
		(m : $\tau$ )	Annotation de type
		{ $\ell = m$ }	Enregistrement
		n	Entier
		c	Caractère
		s	Chaîne de caractères
		K( $\bar{m}$ )	Valeurs étiquetées
		m   m	Disjonction
		m & m	Conjonction

Dans cette grammaire, on a utilisé les *métavariabes* suivantes :

- T pour représenter un constructeur de type.
- x pour représenter un identificateur de valeur.
- $\ell$  pour représenter une étiquette de champs d'enregistrement.
- K pour représenter le nom d'un constructeur de données.
- $\tau$  pour représenter un type.

- $e$  pour représenter une expression.
- $m$  pour représenter un motif.
- $n$  pour représenter un littéral de type entier.
- $c$  pour représenter un littéral de type caractère.
- $s$  pour représenter un littéral de type chaîne de caractères.
- $\alpha$  pour représenter une variable de type.
- $b$  pour représenter une branche d'analyse de motifs.

Par ailleurs, on a aussi écrit  $\bar{X}$  pour représenter une liste potentiellement vide de  $X$ .

## 2 Interprétation

### 2.1 Valeurs

Les valeurs du langage sont définies par la grammaire suivante :

$v ::=$	$n$	<i>Entier</i>
	$c$	<i>Caractère</i>
	$s$	<i>Chaîne de caractères</i>
	$()$	<i>Unité</i>
	$K(\bar{v})$	<i>Valeur étiquetée</i>
	$a$	<i>Adresse d'un enregistrement</i>
	$(m \Rightarrow e)[E]$	<i>Fermeture</i>

**Environnement d'évaluation** Les identificateurs de programme sont associés à des valeurs à l'aide d'un environnement d'évaluation  $E$ . On écrit «  $E[x]$  » pour parler de la valeur de  $x$  dans l'environnement  $E$ . On écrit «  $E + x \mapsto v$  » pour parler de l'environnement  $E$  étendu par l'association entre l'identificateur  $x$  et la valeur  $v$ .

**Valeur d'enregistrement** Une valeur d'enregistrement  $\mathcal{L}$  est écrite «  $\{\ell = v\}$  ». On écrit «  $\mathcal{L}[\ell := v]$  » pour représenter l'enregistrement similaire en tout champ à  $\mathcal{L}$  sauf pour le champ  $\ell$  qui vaut  $v$ .

Les valeurs d'enregistrement sont allouées dynamiquement dans une mémoire  $M$ . On écrit «  $M + a \mapsto \mathcal{L}$  » pour représenter la mémoire qui étend la mémoire  $M$  avec une nouvelle adresse  $a$  où se trouve stockée la valeur d'enregistrement  $\mathcal{L}$ . On écrit «  $M[a \leftarrow \mathcal{L}]$  » pour représenter la mémoire  $M$  modifiée seulement à l'adresse  $a$  en y stockant la valeur d'enregistrement  $\mathcal{L}$ . Enfin, on écrit «  $(a \mapsto \mathcal{L}) \in M$  » pour indiquer que la valeur d'enregistrement  $\mathcal{L}$  est stockée à l'adresse  $a$  de la mémoire  $M$ .

### 2.2 Évaluation des programmes

Un programme  $p$  s'évalue à partir d'une mémoire vide et d'un environnement vide en évaluant successivement les définitions de valeurs dans leur ordre d'apparition dans le programme.

Pour spécifier précisément ce processus, il faut donc décrire la façon dont les définitions de valeurs s'évaluent : c'est le rôle de la section 2.3. Les expressions sont les termes qui s'évaluent en des valeurs. Leur évaluation est spécifiée dans la section 2.4. Elle s'appuie sur l'évaluation de l'analyse par cas (section 2.5) et l'analyse de motifs (section 2.6).

### 2.3 Évaluation des définitions

L'évaluation des définitions s'appuie sur le jugement :

$$E, M \vdash d_v \Rightarrow E', M'$$

qui se lit « Dans l'environnement  $E$  et la mémoire  $M$ , la définition  $d_v$  étend  $E$  et  $E'$  et modifie  $M$  en  $M'$  ».

**Cette partie de la spécification est omise volontairement.** Vous devez réfléchir à une spécification raisonnable.

### 2.4 Évaluation des expressions

Le jugement d'évaluation des expressions s'écrit :

$$E, M \vdash e \Downarrow v, M'$$

et se lit « Dans l'environnement d'évaluation  $E$ , l'expression  $e$  s'évalue en  $v$  et change la mémoire  $M$  en  $M'$ . »

Il est défini par les règles suivantes :

$$\begin{array}{c}
\frac{}{E, M \vdash n \Downarrow n, M} \quad \frac{}{E, M \vdash c \Downarrow c, M} \quad \frac{}{E, M \vdash s \Downarrow s, M} \quad \frac{E(x) = v}{E, M \vdash x \Downarrow v, M} \\
\\
\frac{M_0 = M \quad \forall i \in [1 \dots n], E, M_{i-1} \vdash e_i \Downarrow v_i, M_i}{E, M \vdash K(e_1, \dots, e_n) \Downarrow K(v_1, \dots, v_n), M_n} \\
\\
\frac{M_0 = M \quad \forall i \in [1 \dots n], E, M_{i-1} \vdash e_i \Downarrow v_i, M_i \quad M' = M_n + a \mapsto \{\ell_1 := v_1, \dots, \ell_n := v_n\}}{E, M \vdash \{\ell_1 := e_1, \dots, \ell_n := e_n\} \Downarrow a, M'} \quad \frac{E, M \vdash e \Downarrow v, M'}{E, M \vdash (e : \tau) \Downarrow v, M'} \\
\\
\frac{E, M \vdash d_v \Rightarrow E', M' \quad E', M' \vdash e \Downarrow v, M''}{E, M \vdash d_v; e \Downarrow v, M''} \\
\\
\frac{E, M \vdash e_1 \Downarrow (m \Rightarrow e)[E'], M' \quad E, M' \vdash e_2 \Downarrow v', M'' \quad E', M'' \vdash m \sim v' \Uparrow E'' \quad E'', M'' \vdash e \Downarrow v, M'''}{E, M \vdash e_1 e_2 \Downarrow v, M'''} \\
\\
\frac{E, M \vdash e \Downarrow a, M' \quad (a \mapsto \{\dots \ell := v \dots\}) \in M'}{E, M \vdash e \# \ell \Downarrow v, M'} \quad \frac{E, M \vdash e \Downarrow a, M' \quad (a \mapsto \mathcal{L}) \in M' \quad E, M' \vdash e' \Downarrow v, M''}{E, M \vdash e \# \ell \Leftarrow e' \Downarrow (), M''[a \leftarrow \mathcal{L}[\ell := v]]} \\
\\
\frac{E, M \vdash e \Downarrow v_s, M' \quad E, M' \vdash v_s \sim \bar{b} \Downarrow v, M''}{E, M \vdash e ? \bar{b} \Downarrow v, M''}
\end{array}$$

## 2.5 Analyse par cas

L'analyse par cas d'une valeur  $v$  consiste à traiter une liste de cas représentés par des branches  $\bar{b}$  de la forme «  $m \Rightarrow e$  » en évaluant la première de ces branches dont le motif capture la valeur  $v$ . Ce dernier mécanisme est représenté dans la section suivante.

$$\frac{E, M \vdash v \sim m \Uparrow E' \quad E', M \vdash e \Downarrow v', M'}{E, M \vdash v \sim (m \Rightarrow e) \bar{b} \Downarrow v', M'} \quad \frac{E, M \vdash v \not\sim m \quad E, M \vdash v \sim \bar{b} \Downarrow v', M'}{E, M \vdash v \sim (m \Rightarrow e) \bar{b} \Downarrow v', M'}$$

## 2.6 Analyse de motifs

L'évaluation des motifs s'appuie sur le jugement :

$$E, M \vdash v \sim m \Uparrow E'$$

qui se lit « Dans l'environnement  $E$  et dans la mémoire  $M$ , la valeur  $v$  est capturée par le motif  $m$  en étendant l'environnement  $E$  en  $E'$ . »

**Cette partie de la spécification est omise volontairement.** Vous devez réfléchir à une spécification raisonnable.

## 3 Travail à effectuer

La seconde partie du projet est l'écriture de l'interprète de la sémantique opérationnelle décrite plus haut.

La compilation s'effectue par la commande « **make** ».

Le projet est à rendre **avant le** :

**22 décembre 2015 à 23h59**

Pour finir, vous devez vous assurer des points suivants :

- Le projet contenu dans cette archive **doit compiler**.
  - Vous devez **être les auteurs** de ce projet.
  - Il doit être rendu **à temps**.

Si l'un de ces points n'est pas respecté, la note de 0 vous sera affectée.

## 4 Log