/SU/FSI/MASTER/INFO/MU4IN503 (APS)

Analyse des Programmes et Sémantique

Janvier 2021

Pascal Manoury – Romain Demangeon
pascal.manoury@lip6.fr

3 : Sémantique

Sémantique

Signification des éléments du langage

Description du comportement des programmes à l'exécution

Sémantique *formelle*

- *axiomatique :* interprétation des programmes comme *système de déduction* entre formules décrivant des états mémoire
- *dénotationnelle :* interprétation des programmes comme une *fonction* sur les états mémoire
- *opérationnelle :* interprétation des programmes comme une *relation* de transition entre états mémoires ou valeurs

Sémantique opérationnelle de $APS0$

Domaines sémantiques

Valeurs sémantiques : V

- Valeurs *immédiates* : Z
- Valeurs *fonctionnelles* : *fermetures*, F
- Valeurs fonctionnelles *récursives* : *fermeture récursives*, FR

Union disjointe : $V = Z \oplus F \oplus FR$

Injections canoniques : $\left\{ \begin{array}{lcl} inZ & : & Z \to V \\ inF & : & F \to V \\ inFR & : & FR \to V \end{array} \right.$

cf. type union ou types sommes

Domaines sémantiques

Environnements d'évaluation : E

Association entre identificateurs et valeurs (fonction partielle)

$$E = \text{ident} \rightarrow V$$

Extension : $\rho \in E$, $x \in \texttt{ident}$, $v \in V$

$$\left\{ \begin{array}{lcl} \rho[x = v](x) & = & v \\ \rho[x = v](y) & = & \rho(y) \quad \text{si } y \neq x \end{array} \right.$$

Abréviation :

$$\rho[x_1 = v_1; \ldots; x_n = v_n] \text{ pour } \rho[x_1 = v_1] \ldots [x_n = v_n]$$

Les environnements sont construits par

- les définitions
- les applications de fonction

Fermeture

$$\text{Valeurs d'une fonction} \equiv \left| \begin{array}{l} \text{Ce qu'il faut pour} \\ \textit{appliquer} \text{ la fonction} \end{array} \right.$$

1. une expression (le *corps* de la fonction)
2. une *fonction sémantique* pour étendre un environnement (passage des valeurs des paramètres)

$$F = \textsc{Expr} \times (V^* \to E)$$

Notation (*lambda*)

$$inF(e, \lambda v_1, \ldots, v_n.\rho[x_1 = v_1, \ldots, x_n = v_n])$$

Réduction (exemple) :

$$\lambda v_1, v_2.\rho[\mathtt{x} = v_1, \mathtt{y} = v_2](inZ(45), inZ(67))$$
$$= \rho[\mathtt{x} = inZ(45), \mathtt{y} = inZ(67)]$$

Fermeture récursive

L'environnement d'une fermeture récursive contient
la fermeture récursive

Schématiquement

$inF(e, \lambda v_1, \dots \rho[x_1 = v_1; \dots][x = f]$

avec $f = inF(e, \lambda v_1, \dots \rho[x_1 = v_1; \dots][x = f]$

Briser le cycle : retarder la liaison

$$inFR(\lambda f.inF(e, \lambda v_1, \dots, v_n.\rho[x_1 = v_1; \dots; x_n = v_n][x = f]))$$

$$FR = V \to F$$

Domaines sémantiques

Résumé

Valeurs	$V = Z \oplus F \oplus FR$
Valeurs immédiates	Z
Fermetures	$F = \textsc{Expr} \times (V^* \to E)$
Fermetures récursives	$FR = V \to F$
Environnement	$E = \texttt{ident} \to V$

Modélisation du *flux de sortie* (instruction `ECHO`)

$$O = Z^*$$

(Suite de valeurs immédiates)

Fonctions sémantiques

Lapalissade formelle

Valeur numériques : $\nu : \texttt{num} \to Z$

Exemple : $\nu(42) = 42$

Fonctions primitives : $\pi_1 : Z \to V$, $\pi_2 : Z \times Z \to V$

$$
\begin{array}{rcl}
\pi_1(\texttt{not})(0) & = & 1 \\
\pi_1(\texttt{not})(1) & = & 0
\end{array}
$$

$$
\begin{array}{rcll}
\pi_2(\texttt{eq})(n_1, n_2) & = & 1 & \text{si } n_1 = n_2 \\
 & = & 0 & \text{sinon} \\
\pi_2(\texttt{lt})(n_1, n_2) & = & 1 & \text{si } n_1 < n_2 \\
 & = & 0 & \text{sinon}
\end{array}
$$

$$
\begin{array}{rcl}
\pi_2(\texttt{add})(n_1, n_2) & = & n_1 + n_2 \\
\pi_2(\texttt{sub})(n_1, n_2) & = & n_1 - n_2 \\
\pi_2(\texttt{mul})(n_1, n_2) & = & n_1 \times n_2 \\
\pi_2(\texttt{div})(n_1, n_2) & = & n_1 \div n_2
\end{array}
$$

Règles sémantiques

Contexte d'évaluation : $E \times O$ notation ρ, ω

1. Programmes : production d'un flux de sortie
   $\textsc{Prog} \times O$ notation $\vdash p \leadsto \omega$.
2. Suite de commandes : idem
   $E \times O \times \textsc{Cmds} \times O$ notation $\rho, \omega \vdash_{\textsc{Cmds}} cs \leadsto \omega'$.
3. Instruction : idem
   $E \times O \times \textsc{Stat} \times O$ notation $\rho, \omega \vdash_{\textsc{Stat}} s \leadsto \omega$.
4. Déclaration : produit un environnement
   $E \times \textsc{Dec} \times E$ notation $\rho \vdash_{\textsc{Dec}} d \leadsto \rho'$.
5. Expression : produit une valeur
   $E \times \textsc{Expr} \times V$ notation $\rho \vdash_{\textsc{Expr}} e \leadsto v$.

Par induction sur la construction des programmes

Programme

Relation $\vdash$

Soit $p \in \textsc{Prog}$ tel que $p = [cs]$

$$(\textsc{prog}) \quad \text{si } \emptyset, \emptyset \vdash_{\textsc{Cmds}} (cs\,;\varepsilon) \rightsquigarrow \omega \quad \text{alors } \vdash [cs] \rightsquigarrow \omega$$

avec $\emptyset, \emptyset$ contexte d'évaluation vide.

Suite de commandes

Relation $\vdash_{\text{CMDS}}$

Hypothèse : les suites de commandes se terminent pas ε

Déclaration

(DECS) si $\rho, \omega \vdash_{\text{DEC}} d \rightsquigarrow \rho'$
et si $\rho', \omega \vdash_{\text{CMDS}} cs \rightsquigarrow \omega'$
alors $\rho, \omega \vdash_{\text{CMDS}} (d\,;\, cs) \rightsquigarrow \omega'$

Instruction

(STATS) si $\rho, \omega \vdash_{\text{STAT}} s \rightsquigarrow \omega'$
et si $\rho, \omega' \vdash_{\text{CMDS}} cs \rightsquigarrow \omega''$
alors $\rho, \omega \vdash_{\text{CMDS}} (s\,;\, cs) \rightsquigarrow \omega''$

Commande vide

(END) $\rho, \omega \vdash_{\text{CMDS}} \varepsilon \rightsquigarrow \omega$

Instruction

Relation $\vdash_{\textsc{Stat}}$

Émission d'une valeur sur le flux de sortie

$$
\begin{array}{ll}
(\textsc{echo}) & \text{si } \rho, \omega \vdash_{\textsc{Expr}} e \rightsquigarrow inZ(n) \\
 & \text{alors } \rho, \omega \vdash_{\textsc{Stat}} (\texttt{ECHO}\ e) \rightsquigarrow (n \cdot \omega)
\end{array}
$$

Déclaration de constante

Relation $\vdash_{\textsc{Dec}}$

Ajout à l'environnement de la valeur de la constante

$$(\textsc{const}) \quad \text{si } \rho \vdash_{\textsc{Expr}} e \leadsto v$$
$$\text{alors } \rho \vdash_{\textsc{Dec}} (\texttt{CONST}\ x\ t\ e) \leadsto \rho[x = v]$$

La valeur est obtenue avec $\vdash_{\textsc{Expr}}$

Déclaration de fonction

Relation $\vdash_{\textsc{Stat}}$

- Construction de la fermeture
- Capture de l'environnement de définition

Liaison statique

$$\begin{array}{l}(\textsc{Fun})\ \ \rho \vdash_{\textsc{Dec}} (\texttt{FUN}\ x\ t\ \ [x_1 : t_1\,,\ \ldots\ , x_n : t_n]\ \ e) \\ \qquad \rightsquigarrow \rho[x = inF(e, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots ; x_n = v_n])\end{array}$$

Déclaration de fonction récursive

Relation $\vdash_{\textsc{Stat}}$

Fermeture récursive

$$(\textsc{FunRec})\ \ \rho \vdash_{\textsc{Dec}} (\texttt{FUN REC}\ x\ t\ [x_1 : t_1\,, \ldots\,, x_n : t_n]\ e)$$
$$\rightsquigarrow \rho[x = inFR(\lambda f.inF(e, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots;$$
$$x_n = v_n][x = f])$$

Expressions atomiques

Relation $\vdash_{\textsc{Expr}}$

Les constantes booléennes sont interprétée par 0 et 1

$$(\textsc{true}) \quad \rho \vdash_{\textsc{Expr}} \texttt{true} \rightsquigarrow \mathit{inZ}(1)$$

$$(\textsc{false}) \quad \rho \vdash_{\textsc{Expr}} \texttt{false} \rightsquigarrow \mathit{inZ}(0)$$

La valeur d'une constante numérique est donnée par la fonction ν :

$(\textsc{num})$ si $n \in \texttt{num}$
alors $\rho \vdash_{\textsc{Expr}} n \rightsquigarrow \mathit{inZ}(\nu(n))$

La valeur d'un identificateur est celle donnée par l'environnement :

$(\textsc{id})$ si $x \in \texttt{ident}$ et $\rho(x) = v$
alors $\rho \vdash_{\textsc{Expr}} x \rightsquigarrow v$

Fonction primitive

Relation $\vdash_{\text{EXPR}}$

Opérateur unaire

(PRIM1) si $\rho \vdash_{\text{EXPR}} e \leadsto inZ(n)$,
et si $\pi_1(not)(n) = n'$
alors $\rho \vdash_{\text{EXPR}} (\texttt{not}\ e) \leadsto inZ(n')$

Opérateurs binaires

(PRIM2) si $x \in \{\texttt{eq lt add sub mul div}\}$,
si $\rho \vdash_{\text{EXPR}} e_1 \leadsto inZ(n_1)$,
si $\rho \vdash_{\text{EXPR}} e_2 \leadsto inZ(n_2)$
et si $\pi_2(x)(n_1, n_2) = n$
alors $\rho \vdash_{\text{EXPR}} (x\ e_1 e_2) \leadsto inZ(n)$

Opérateurs séquentiels

Relation $\vdash_{\textsc{Expr}}$

Opérateurs booléens : évaluation *paresseuse*

$(\textsc{and0})$ si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inZ(0)$
alors $\rho \vdash_{\textsc{Expr}} (\texttt{and}\ e_1\ e_2) \rightsquigarrow inZ(0)$.

$(\textsc{and1})$ si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inZ(1)$
si $\rho \vdash_{\textsc{Expr}} e_2 \rightsquigarrow v$
alors $\rho \vdash_{\textsc{Expr}} (\texttt{and}\ e_1\ e_2) \rightsquigarrow v$.

$(\textsc{or1})$ si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inZ(1)$
alors $\rho \vdash_{\textsc{Expr}} (\texttt{or}\ e_1\ e_2) \rightsquigarrow inZ(1)$.

$(\textsc{or0})$ si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inZ(0)$
si $\rho \vdash_{\textsc{Expr}} e_2 \rightsquigarrow v$
alors $\rho \vdash_{\textsc{Expr}} (\texttt{or}\ e_1\ e_2) \rightsquigarrow v$.

Alternative

Relation $\vdash_{\textsc{Expr}}$

Opérateur *de contrôle*

1. évaluer la condition
2. selon le résultat, évaluer l'une ou l'autre alternative

$$
\begin{array}{ll}
(\textsc{if1}) & \text{si } \rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inZ(1) \\
& \text{si } \rho \vdash_{\textsc{Expr}} e_2 \rightsquigarrow v \\
& \text{alors } \rho \vdash_{\textsc{Expr}} (\texttt{if}\ e_1\ e_2\ e_3) \rightsquigarrow v \\
(\textsc{if0}) & \text{si } \rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inZ(0) \\
& \text{si } \rho \vdash_{\textsc{Expr}} e_3 \rightsquigarrow v \\
& \text{alors } \rho \vdash_{\textsc{Expr}} (\texttt{if}\ e_1\ e_2\ e_3) \rightsquigarrow v
\end{array}
$$

Abstraction

Relation $\vdash_{\textsc{Expr}}$

Capture de l'environnement courant

$$(\textsc{Abs}) \quad \rho \vdash_{\textsc{Expr}} [x_1 : t_1\,, \ldots\,, x_n : t_n]\, e \leadsto inF(e, \lambda v_1, \ldots, v_n.\rho[x_1 = v_1; \ldots; x_n = v_n])$$

Application 1

Relation $\vdash_{\textsc{Expr}}$

Expression de la forme $(e\ e_1 \ldots e_n)$

1. e s'évalue sur une *fermeture* :
   - corps de la fonction e'
   - passage des paramètres
2. évaluer les arguments
3. ajouter leurs valeurs à l'environnement *capturé*

$$
\begin{array}{ll}
(\textsc{App}) & \text{si } \rho \vdash_{\textsc{Expr}} e \rightsquigarrow inF(e', r), \\
& \text{si } \rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow v_1, \\
& \ldots, \\
& \text{si } \rho \vdash_{\textsc{Expr}} e_n \rightsquigarrow v_n \\
& \text{et si } r(v_1, \ldots, v_n) \vdash_{\textsc{Expr}} e' \rightsquigarrow v \\
& \text{alors } \rho \vdash (e\ e_1 \ldots e_n) \rightsquigarrow v
\end{array}
$$

Application 2

Relation $\vdash_{\textsc{Expr}}$

Fermeture récursive

$$\underbrace{inFR(\lambda f.inF(e, \underbrace{\lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n][x = f]}_{r}))}_{\varphi}$$

- liaison auto-référente (φ)
- passage des paramètres (r)

(APPR) si $\rho \vdash_{\textsc{Expr}} e \rightsquigarrow inFR(\varphi)$,
si $\varphi(inFR(\varphi)) = inF(e', r)$,
si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow v_1$, …, si $\rho \vdash_{\textsc{Expr}} e_n \rightsquigarrow v_n$
et si $r(v_1, \ldots, v_n) \vdash_{\textsc{Expr}} e' \rightsquigarrow v$
alors $\rho \vdash (e\ e_1 \ldots e_n) \rightsquigarrow v$