UPMC/master/info/4I503 APS
APS3: formulaire

P. Manoury

Mai 2018

Contents

1 Syntaxe **1**
- 1.1 Lexique . . . 1
- 1.2 Grammaire . . . 2

2 Typage **3**
- 2.1 Jugements de typages . . . 3
- 2.2 Expression . . . 3
- 2.3 Instruction . . . 3
- 2.4 Déclaration . . . 4
- 2.5 Suite de commandes . . . 4
- 2.6 Programme . . . 4

3 Sémantique **4**
- 3.1 Domaines sémantiques . . . 4
- 3.2 Opérations sémantiques . . . 5
- 3.3 Relations sémantiques . . . 6
- 3.4 Expression . . . 6
- 3.5 Valeurs gauches . . . 7
- 3.6 Instruction . . . 7
- 3.7 La commande `RETURN` . . . 8
- 3.8 Déclaration . . . 8
- 3.9 Suite de commandes . . . 8
- 3.10 Bloc . . . 8
- 3.11 Programme . . . 8

1 Syntaxe

1.1 Lexique

Symboles réservés `[ ] ( ) ; , * ->`

Mot clef

```
CONST FUN REC VAR PROC
bool int void vec
true false not and or
eq lt add sub mul div
if
```

```
len nth alloc
ECHO SET IF WHILE CALL
RETURN
```

Constantes numériques num défini par *('-'?)['0'-'9']+*

Identificateurs ident défini par *(['a'-'z''A'-'Z'])(['a'-'z''A'-'Z''0'-'9'])**
dont on exclut les mots clef.

Séparateurs: l'espace, la tabulation, le passage à la ligne et le retour chariot.
Sous ensembles utiles de mots clef:

- oprim l'ensemble de mots clef: `not and or eq lt add sub mul div`
- tprim l'ensemble de mots clefs `bool int void`

1.2 Grammaire

```
PROG   ::= [ CMDS ]
CMDS   ::= STAT
         | RET
         | DEC ; CMDS
         | STAT ; CMDS
TYPE   ::= tprim
         | ( TYPES -> TYPE )
         | (vec TYPE )
TYPES  ::= TYPE
         | TYPE * TYPES
ARG    ::= ident : TYPE
ARGS   ::= ARG
         | ARG , ARGS
DEC    ::= CONST ident TYPE EXPR
         | FUN ident TYPE [ ARGS ] EXPR
         | FUN REC ident TYPE [ ARGS ] EXPR
         | VAR ident TYPE
         | PROC ident [ ARGS ] PROG
         | PROC REC ident [ ARGS ] PROG
         | FUN ident TYPE [ ARGS ] PROG
         | FUN REC ident TYPE [ ARGS ] PROG
RET    ::= RETURN EXPR
STAT   ::= ECHO EXPR
         | SET LVAL EXPR
         | IF EXPR PROG PROG
         | WHILE EXPR PROG
         | CALL ident EXPRS
LVAL   ::= ident
         | (nth LVAL EXPR )
EXPR   ::= bool | num | ident
         | ( oprim EXPRS )
         | ( if EXPR EXPR EXPR )
         | (alloc EXPR ) | (len EXPR ) | (nth EXPR EXPR )
         | [ ARGS ] EXPR
         | ( EXPR EXPRS )
EXPRS  ::= EXPR
         | EXPR EXPRS
```

2 Typage

2.1 Jugements de typages

- Commande vide: ε
- Suites de commandes terminées par la commande vide : $\textsc{Cmds}_\varepsilon$
- Types sommes: $t + \texttt{void}$ avec $t \in \textsc{Type}$ et $\texttt{void} + \texttt{void} = \texttt{void}$
- Types étendus: $\textsc{Xtype} ::= \textsc{Type} \mid \textsc{Type} + \texttt{void}$

	Symbole	Domaine	Notation
Programme	$\vdash$	$\textsc{Prog} \times \{\texttt{void}\}$	$\vdash p : \texttt{void}$
Suite de commande	$\vdash_{\textsc{Cmds}}$	$G \times \textsc{Cmds}_\varepsilon \times \textsc{Xtype}$	$\Gamma \vdash_{\textsc{Cmds}} cs : t$
Déclaration	$\vdash_{\textsc{Dec}}$	$G \times \textsc{Dec} \times G$	$\Gamma \vdash_{\textsc{Dec}} d : \Gamma'$
Instruction	$\vdash_{\textsc{Stat}}$	$G \times \textsc{Stat} \times \textsc{Xtype}$	$\Gamma \vdash_{\textsc{Stat}} s : t$
Expression	$\vdash_{\textsc{Expr}}$	$G \times \textsc{Expr} \times \textsc{Type}$	$\Gamma \vdash_{\textsc{Expr}} e : t.$

2.2 Expression

(NUM) si $n \in \mathsf{num}$ alors $\Gamma \vdash_{\textsc{Expr}} n : \texttt{int}$

(SYM) si $x \in \mathsf{sym}$ et si $\Gamma(x) = t$ alors $\Gamma \vdash_{\textsc{Expr}} x : t$

(ABS) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n] \vdash_{\textsc{Expr}} e : t$ alors $\Gamma \vdash_{\textsc{Expr}} \texttt{[}x_1 : t_1, \ldots, x_n : t_n\texttt{]}e : t_1 \texttt{ * } \ldots \texttt{ * } t_n \texttt{ -> } t$

(APP) si $\Gamma \vdash_{\textsc{Expr}} e_1 : t_1$, ..., si $\Gamma \vdash_{\textsc{Expr}} e_n : t_n$ et si $\Gamma \vdash_{\textsc{Expr}} e : t_1 \texttt{ * } \ldots \texttt{ * } t_n \texttt{ -> } t$
alors $\Gamma \vdash_{\textsc{Expr}} \texttt{(}e\ e_1 \ldots e_n\texttt{)} : t$

(IF) si $\Gamma \vdash_{\textsc{Expr}} e_1 : \texttt{bool}$, si $\Gamma \vdash_{\textsc{Expr}} e_2 : t$ et si $\Gamma \vdash_{\textsc{Expr}} e_3 : t$
alors $\Gamma \vdash_{\textsc{Expr}} \texttt{(if } e_1\ e_2\ e_3\texttt{)} : t$

(ALLOC) pour tout $t \in \textsc{Type}$, si $\Gamma \vdash_{\textsc{Expr}} e : \texttt{int}$ alors $\Gamma \vdash_{\textsc{Expr}} \texttt{(alloc } e\texttt{)} : \texttt{(vec } t\texttt{)}$

(NTH) pour tout $t \in \textsc{Type}$, si $\Gamma \vdash_{\textsc{Expr}} e_1 : \texttt{(vec } t\texttt{)}$ et si $\Gamma \vdash_{\textsc{Expr}} e_2 : \texttt{int}$ alors $\Gamma \vdash_{\textsc{Expr}} \texttt{(nth } e_1\ e_2\texttt{)} : t$

(LEN) pour tout $t \in \textsc{Type}$, si $\Gamma \vdash_{\textsc{Expr}} e : \texttt{(vec } t\texttt{)}$ alors $\Gamma \vdash_{\textsc{Expr}} \texttt{(len } e\texttt{)} : \texttt{int}$

2.3 Instruction

(ECHO) si $\Gamma \vdash_{\textsc{Expr}} e : \texttt{int}$ alors $\Gamma \vdash_{\textsc{Stat}} (\texttt{ECHO } e) : \mathit{void}$

(SET) si $\Gamma \vdash_{\textsc{Expr}} lv : t$ et si $\Gamma \vdash_{\textsc{Expr}} e : t$ alors $\Gamma \vdash_{\textsc{Stat}} (\texttt{SET } lv\ e) : \texttt{void}$

(IF0) pour tout type t, si $G \vdash_{\textsc{Expr}} e : \texttt{bool}$ et $G \vdash_{\textsc{Block}} blk_1 : t$ et $G \vdash_{\textsc{Block}} blk_2 : t$
alors $G \vdash_{\textsc{Stat}} (\texttt{IF } e\ blk_1\ blk_2) : t$

(IF1) pour tout $t \neq \texttt{void}$, si $G \vdash_{\textsc{Expr}} e : \texttt{bool}$ et $G \vdash_{\textsc{Block}} blk_1 : \texttt{void}$ et $G \vdash_{\textsc{Block}} blk_2 : t$
alors $G \vdash_{\textsc{Stat}} (\texttt{IF } e\ blk_1\ blk_2) : t + \texttt{void}$

(IF2) pour tout $t \neq \texttt{void}$, si $G \vdash_{\textsc{Expr}} e : \texttt{bool}$ et $G \vdash_{\textsc{Block}} blk_1 : t$ et $G \vdash_{\textsc{Block}} blk_2 : \texttt{void}$
alors $G \vdash_{\textsc{Stat}} (\texttt{IF } e\ blk_1\ blk_2) : t + \texttt{void}$

(WHILE) pour tout type t, si $G \vdash_{\textsc{Expr}} e : \texttt{bool}$ et $G \vdash_{\textsc{Block}} blk : t$ alors $G \vdash_{\textsc{Stat}} (\texttt{WHILE } e\ blk) : t + \texttt{void}$

(CALL) si $\Gamma(x) = t_1 \texttt{ * } \ldots \texttt{ * } t_n \texttt{ -> void}$, si $\Gamma \vdash_{\textsc{Expr}} e_1 : t_1$, ... et si $\Gamma \vdash_{\textsc{Expr}} e_n : t_n$
alors $\Gamma \vdash_{\textsc{Stat}} (\texttt{CALL } x\ e_1 \ldots e_n) : \texttt{void}$

2.4 Déclaration

(CONST) si $\Gamma \vdash_{\text{Expr}} e : t$ alors $\Gamma \vdash_{\text{Dec}}$ (`CONST` $x\ t\ e$) : $\Gamma[x : t]$

(FUN) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n] \vdash_{\text{Expr}} e : t$
alors $\Gamma \vdash_{\text{Dec}}$ (`FUN` $x\ t$ `[`x_1`:`t_1`,` ... `,`x_n`:`t_n`]` e) : $\Gamma[x : t_1$ `*` ... `*` t_n `->` $t]$

(FUNREC) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n; x : t_1$ `*` ... `*` t_n `->` $t] \vdash_{\text{Expr}} e : t$
alors $\Gamma \vdash_{\text{Dec}}$ (`FUN REC` $x\ t$ `[`x_1`:`t_1`,` ... `,`x_n`:`t_n`]` e) : $\Gamma[x : t_1$ `*` ... `*` t_n `->` $t]$

(VAR) $\Gamma \vdash_{\text{Dec}}$ (`VAR` $x\ t$) : $\Gamma[x : t]$

(PROC) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n] \vdash_{\text{Cmds}} (cs;\varepsilon)$: `void`
alors $\Gamma \vdash_{\text{Dec}}$ (`PROC` x `[`x_1`:`$t_1, \ldots, x_n$`:`t_n`]` `[`cs`]`) : $\Gamma[x : t_1$ `*` ... `*` t_n `-> void`$]$

(PROCREC) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n; x : t_1$ `*` ... `*` t_n `-> void`$] \vdash_{\text{Cmds}} (cs;\varepsilon)$: `void`
alors $\Gamma \vdash_{\text{Dec}}$ (`PROC REC` x `[`x_1`:`$t_1, \ldots, x_n$`:`t_n`]` `[`cs`]`) : $\Gamma[x : t_1$ `*` ... `*` t_n `-> void`$]$

(FUNP) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n] \vdash_{\text{Cmds}} (cs;\varepsilon) : t$
alors $\Gamma \vdash_{\text{Dec}}$ (`FUN` $x\ t$ `[`x_1`:`t_1`,` ... `,`x_n`:`t_n`]` `[`cs`]`) : $\Gamma[x : t_1$ `*` ... `*` t_n `->` $t]$

(FUNRECP) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n; x : t_1$ `*` ... `*` t_n `->` $t] \vdash_{\text{Cmds}} (cs;\varepsilon) : t$
alors $\Gamma \vdash_{\text{Dec}}$ (`FUN REC` $x\ t$ `[`x_1`:`t_1`,` ... `,`x_n`:`t_n`]` `[`cs`]`) : $\Gamma[x : t_1$ `*` ... `*` t_n `->` $t]$

2.5 Suite de commandes

(DEC) si $d \in$ DEC, si $\Gamma \vdash_{\text{Dec}} d : \Gamma'$ et si $\Gamma' \vdash_{\text{Cmds}} cs : t$ alors $\Gamma \vdash_{\text{Cmds}} (d$`;` $cs) : t$.

(STAT0) pour tout type t, si $\Gamma \vdash_{\text{Stat}} s$: `void` et $\Gamma \vdash_{\text{Cmds}} cs : t$ alors $\Gamma \vdash_{\text{Cmds}} (s$`;`$cs) : t$

(STAT1) si $t \neq$ `void`, si $\Gamma \vdash_{\text{Stat}} s : t +$ `void` et $\Gamma \vdash_{\text{Cmds}} cs : t$ alors $\Gamma \vdash_{\text{Cmds}} (s$`;`$cs) : t$

(RET) si $\Gamma \vdash_{\text{Expr}} e : t$ alors $\Gamma \vdash_{\text{Cmds}}$ (`RETURN` $e; \varepsilon) : t$

(END) $\Gamma \vdash_{\text{Cmds}} \varepsilon$: `void`

2.6 Programme

(PROG) si $\Gamma_0 \vdash_{\text{Cmds}} (cs$`;`$\varepsilon)$: `void` alors $\vdash$ `[`cs`]` : `void`

3 Sémantique

3.1 Domaines sémantiques

- Valeurs immédiates (entiers): N
- Flux de sortie: $O = N^*$
- Adresses: $A = N$
- Blocs mémoires: $B = A \times N$
- Mémoire $S = A \to N \oplus B$
- Valeurs: $V = N \oplus F \oplus FR \oplus A \oplus P \oplus PR \oplus B$
- Valeurs étendues: $V_\varepsilon = V \cup \{\varepsilon\}$

- Fermetures: $F = \textsc{Expr} \times (V^* \to E)$
- Fermetures récursives: $FR = V \to F$
- Environnements: $E = \mathsf{ident} \to V$
- Fermetures procédurales $P = \textsc{Cmds} \times (V^* \to E)$
- Fermetures procédurales récursives $PR = V \to P$

3.2 Opérations sémantiques

Constantes numériques

$$\nu : \mathsf{num} \to N$$

Opérateurs primitifs

$$\begin{array}{rcll}
\pi(\texttt{not})(0) & = & 1 & \\
\pi(\texttt{not})(1) & = & 0 & \\
\pi(\texttt{and})(0,n) & = & 0 & \\
\pi(\texttt{and})(1,n) & = & n & \\
\pi(\texttt{or})(1,n) & = & 1 & \\
\pi(\texttt{or})(0,n) & = & n & \\
\pi(\texttt{eq})(n_1,n_2) & = & 1 & \text{si } n_1 = n_2 \\
 & = & 0 & \text{sinon} \\
\pi(\texttt{lt})(n_1,n_2) & = & 1 & \text{si } n_1 < n_2 \\
 & = & 0 & \text{sinon} \\
\pi(\texttt{add})(n_1,n_2) & = & n_1 + n_2 & \\
\pi(\texttt{sub})(n_1,n_2) & = & n_1 - n_2 & \\
\pi(\texttt{mul})(n_1,n_2) & = & n_1 \cdot n_2 & \\
\pi(\texttt{div})(n_1,n_2) & = & n_1 \div n_2 &
\end{array}$$

Environnement

- Extension des environnements: $\rho[x = v](x) = v$ et $\rho[x = v](y) = \rho(y)$ lorsque x et y sont des symboles différents.

Flot de sortie

- Ajout au flux de sortie: $n \cdot \omega$

Mémoire

- Allocation: $alloc(\sigma) = (a, \sigma')$ si et seulement si $a \notin \mathsf{dom}(\sigma)$ et $\sigma' = \sigma[a = any]$
- Allocation multiple: $allocn(\sigma, n) = (a, \sigma')$ si et seulement si pour tout $i \in [0, n[$, $a + i \notin \mathsf{dom}(\sigma)$ et $\sigma' = \sigma[a = any; \ldots ; a + n - 1 = any]$.
- Modification: $\sigma[a = v'][a := v] = \sigma[a = v]$ et $\sigma[a' = v'][a := v] = \sigma[a := v][a' = v']$ lorsque a est différent de a'
- Restriction: soit $\alpha : V_\varepsilon \to \mathcal{P}(A)$

$$\begin{array}{lcl}
\alpha(\varepsilon) & = & \emptyset \\
\alpha(inN(n)) & = & \emptyset \\
\alpha(inV(a)) & = & \{a\} \\
\alpha(inB(a,n)) & = & \{a + i \mid i \in [0..n[\}
\end{array}$$

$Ac(\rho,\sigma) = \bigcup_{i\in I\!N} A_i$ avec

$$A_0 = \bigcup_{x\in \mathsf{dom}(\rho)} \alpha(\rho(x))$$

$$A_{n+1} = \bigcup_{a\in A_n} \alpha(\sigma(a))$$

$(\sigma/\rho)(a) = \sigma(a)$ si $a \in Ac(\rho,\sigma)$ et $(\sigma/\rho)(a)$ non définie sinon.

3.3 Relations sémantiques

	Symbole	Domaine	Notation
Programme	$\vdash$	$\textsc{Prog} \times S \times O$	$\vdash$ `[`cs`]` $\rightsquigarrow (\sigma,\omega)$
Bloc	$\vdash_{\textsc{Block}}$	$E \times S \times O \times \textsc{Cmds} \times V_\varepsilon \times S \times O$	$\rho,\sigma,\omega \vdash_{\textsc{Block}} bk \rightsquigarrow (v,\sigma',\omega')$
Suite de commandes	$\vdash_{\textsc{Cmds}}$	$E \times S \times O \times (\textsc{Cmds}_\varepsilon) \times V_\varepsilon \times S \times O.$	$\rho,\sigma,\omega \vdash_{\textsc{Cmds}} cs \rightsquigarrow (v,\sigma',\omega')$
Déclaration	$\vdash_{\textsc{Dec}}$	$E \times S \times O \times \textsc{Dec} \times E \times S \times O$	$\rho,\sigma,\omega \vdash_{\textsc{Dec}} d \rightsquigarrow (\rho',\sigma',\omega')$
Instruction	$\vdash_{\textsc{Stat}}$	$E \times S \times O \times \textsc{Stat} \times V_\varepsilon \times S \times O$	$\rho,\sigma,\omega \vdash_{\textsc{Stat}} s \rightsquigarrow (v,\sigma',\omega')$
Return	$\vdash_{\textsc{Ret}}$	$E \times S \times O \times \textsc{Ret} \times V \times S \times O$	$\rho,\sigma,\omega \vdash_{\textsc{Ret}} r \rightsquigarrow (v,\sigma',\omega')$
Valeur gauche	$\vdash_{\textsc{Lval}}$	$E \times S \times O \times \textsc{Lval} \times A \times S \times O$	$\rho,\sigma,\omega \vdash_{\textsc{Lval}} lv \rightsquigarrow (a,\sigma',\omega')$
Expression	$\vdash_{\textsc{Expr}}$	$E \times S \times O \times \textsc{Expr} \times V \times S \times O$	$\rho,\sigma,\omega \vdash_{\textsc{Expr}} e \rightsquigarrow (v,\sigma',\omega')$

3.4 Expression

(TRUE) $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `true` $\rightsquigarrow (inN(1),\sigma,\omega)$

(FALSE) $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `false` $\rightsquigarrow (inN(0),\sigma,\omega)$

(NUM) si $n \in \mathsf{num}$ alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}} n \rightsquigarrow (inN(\nu(n)),\sigma,\omega)$

(ID1) si $x \in \mathsf{ident}$ et $\rho(x) = inA(a)$ alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}} x \rightsquigarrow (inN(\sigma(a)),\sigma,\omega)$

(ID2) si $x \in \mathsf{ident}$ et $\rho(x) = v$, avec $v \neq inA(a)$ alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}} x \rightsquigarrow (v,\sigma,\omega)$

(PRIM) si $x \in \mathsf{oprim}$, si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e_1 \rightsquigarrow (inN(n_1),\sigma_1,\omega_1)$, ..., si $\rho,\sigma_{k-1},\omega_{k-1} \vdash_{\textsc{Expr}} e_k \rightsquigarrow (inN(n_k),\sigma_k,\omega_k)$
et si $\pi(x)(n_1,\ldots,n_k) = n$ alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `(`$x\ e_1 \ldots e_n$`)` $\rightsquigarrow (inN(n),\sigma_k,\omega_k)$

(ALLOC) si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e \rightsquigarrow (inN(n),\sigma',\omega')$ et si $allocn(\sigma',n) = (a,\sigma'')$
alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `(alloc` e`)` $\rightsquigarrow (inB(a,n),\sigma'',\omega')$

(NTH) si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e_1 \rightsquigarrow (inB(a,n),\sigma',\omega')$ et si $\rho,\sigma',\omega' \vdash_{\textsc{Expr}} (inN(i),\sigma'',\omega'')$
alors $\rho,\sigma \vdash_{\textsc{Expr}}$ `(nth` $e_1\ e_2$`)` $\rightsquigarrow (\sigma''(a+i),\sigma'',\omega'')$

(LEN) si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e \rightsquigarrow (inB(a,n),\sigma',\omega')$ alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `(len` e`)` $\rightsquigarrow (inN(n),\sigma',\omega')$

(IF1) si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e_1 \rightsquigarrow (inN(1),\sigma',\omega')$ et si $\rho,\sigma',\omega' \vdash_{\textsc{Expr}} e_2 \rightsquigarrow (v,\sigma'',\omega'')$
alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `(if` $e_1\ e_2\ e_3$`)` $\rightsquigarrow (v,\sigma'',\omega'')$

(IF2) si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e_1 \rightsquigarrow (inN(0),\sigma',\omega')$ et si $\rho,\sigma',\omega' \vdash_{\textsc{Expr}} e_3 \rightsquigarrow (v,\sigma'',\omega'')$
alors $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `(if` $e_1\ e_2\ e_3$`)` $\rightsquigarrow (v,\sigma'',\omega'')$

(ABS) $\rho,\sigma,\omega \vdash_{\textsc{Expr}}$ `[`x_1`:`t_1`,` ... `,`x_n`:`t_n`]`$e \rightsquigarrow (inF(e,\lambda v_1 \ldots v_n.\rho[x_1 = v_1;\ldots;x_n = v_n]),\sigma,\omega)$

(APP) si $\rho,\sigma,\omega \vdash_{\textsc{Expr}} e \rightsquigarrow (inF(e',r),\sigma',\omega')$,
si $\rho,\sigma',\omega' \vdash_{\textsc{Expr}} e_1 \rightsquigarrow (v_1,\sigma_1,\omega_1)$, ..., si $\rho,\sigma_{n-1},\omega_{n-1} \vdash_{\textsc{Expr}} e_n \rightsquigarrow (v_n,\sigma_n,\omega_n)$
et si $r(v_1,\ldots,v_n),\sigma_n,\omega_n \vdash_{\textsc{Expr}} e' \rightsquigarrow (v,\sigma'',\omega'')$
alors $\rho,\sigma,\omega \vdash$ `(`$e\ e_1 \ldots e_n$`)` $\rightsquigarrow (v,\sigma'',\omega'')$

(APPR) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inFR(\varphi), \sigma', \omega()$, si $\varphi(inFR(\varphi)) = inF(e', r)$,
si $\rho, \sigma', \omega' \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, $\ldots$, si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
et si $r(v_1, \ldots, v_n), \sigma_n, \omega_n \vdash_{\text{Expr}} e' \rightsquigarrow (v, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash (e\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma'', \omega'')$

(APP') si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inP(bk, r), \sigma', \omega')$,
si $\rho, \sigma', \omega' \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, $\ldots$, si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
et si $r(v_1, \ldots, v_n), \sigma_n, \omega_n \vdash_{\text{Block}} bk \rightsquigarrow (v, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash (e\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma'', \omega'')$

(APPR') si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inPR(\varphi), \sigma', \omega')$, si $\varphi(inPR(\varphi)) = inP(bk, r)$,
si $\rho, \sigma', \omega' \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, $\ldots$, si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
et si $r(v_1, \ldots, v_n), \sigma_n, \omega_n \vdash_{\text{Block}} bk \rightsquigarrow (v, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash (e\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma'', \omega'')$

3.5 Valeurs gauches

(LIDA) si $x \in \mathsf{ident}$ et si $\rho(x) = inA(a)$ alors $\rho, \sigma, \omega \vdash_{\text{Lval}} x \rightsquigarrow (a, \sigma, \omega)$

(LIDB) si $x \in \mathsf{ident}$ et si $\rho(x) = inB(a, n)$ alors $\rho, \sigma, \omega \vdash_{\text{Lval}} x \rightsquigarrow (a, \sigma, \omega)$

(LNTH) si $\rho, \sigma, \omega \vdash_{\text{Expr}} lv \rightsquigarrow (inB(a, n), \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Expr}} e \rightsquigarrow (inN(i), \sigma'', \omega'')$
alors $\rho, \sigma \vdash_{\text{Lval}} (\texttt{nth}\ lv\ e) \rightsquigarrow (a + i, \sigma'', \omega'')$

3.6 Instruction

(ECHO) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inN(n), \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{ECHO}\ e) \rightsquigarrow (\varepsilon, \sigma', (n \cdot \omega))$

(SET) si $\rho, \sigma, \omega \vdash_{\text{Expr}} lv \rightsquigarrow a$ et si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (v, \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{SET}\ lv\ e) \rightsquigarrow (\varepsilon, \sigma'[x = v], \omega')$

(IF1) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inN(1), \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Block}} bk_1 \rightsquigarrow (v, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{IF}\ bk_1\ bk_2) \rightsquigarrow (v, \sigma'', \omega'')$

(IF2) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inN(0), \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Block}} bk_2 \rightsquigarrow (v, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{IF}\ bk_1\ bk_2) \rightsquigarrow (v, \sigma'', \omega'')$

(LOOP0) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inN(0), \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ blk) \rightsquigarrow (\varepsilon, \sigma', \omega')$

(LOOP1A) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inN(1), \sigma', \omega')$,
si $\rho, \sigma', \omega' \vdash_{\text{Block}} blk \rightsquigarrow (\varepsilon, \sigma'', \omega'')$ et si $\rho, \sigma'', \omega'' \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ blk) \rightsquigarrow (v, \sigma''', \omega''')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ blk) \rightsquigarrow (v, \sigma''', \omega''')$

(LOOP1B) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inN(1), \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Block}} blk \rightsquigarrow (v, \sigma'', \omega'')$, avec $v \neq \varepsilon$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ blk) \rightsquigarrow (v, \sigma'', \omega'')$

(CALL) si $\rho(x) = inP(bk, r)$,
si $\rho, \sigma, \omega \vdash_{\text{Expr}} (e_1, \sigma_1) \rightsquigarrow (v_1, \sigma_1, \omega_1)$, $\ldots$, si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
et si $r(v_1, \ldots, v_n), \sigma_n, \omega_n \vdash_{\text{Block}} bk \rightsquigarrow (v, \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma', \omega')$

(CALLR) si $\rho(x) = inPR(\varphi)$, si $\varphi(inPR(\varphi)) = inP(bk, r)$,
si $\rho, \sigma, \omega \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, $\ldots$, si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
et si $r(v_1, \ldots, v_n), \sigma_n, \omega_n \vdash_{\text{Block}} bk \rightsquigarrow (v, \sigma', \omega'))$
alors $\rho, \omega \vdash_{\text{Stat}} (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma', \omega')$

3.7 La commande RETURN

(RET) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (v, \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Ret}}$ (RETURN e) $\rightsquigarrow (v, \sigma', \omega')$

3.8 Déclaration

(CONST) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (v, \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (CONST $x\ t\ e$) $\rightsquigarrow (\rho[x = v], \sigma', \omega')$

(VAR) si $alloc(\sigma) = (a, \sigma')$ alors $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (VAR $x\ t$) $\rightsquigarrow (\rho[x = inA(a)], \sigma', \omega)$

(FUN) $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (FUN $x\ t$ [x_1:t_1, ... ,x_n:t_n] e)
$\rightsquigarrow (\rho[x = inF(e, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma, \omega)$

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

(PROC) $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (PROC $x\ t$ [x_1:t_1, ... ,x_n:t_n] bk)
$\rightsquigarrow (\rho[x = inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma, \omega)$

(PROCREC) $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (PROC REC $x\ t$ [x_1:t_1, ... ,x_n:t_n] bk)
$\rightsquigarrow (\rho[x = inPR(\lambda f.inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n][x = f]), \sigma, \omega)$

(FUNP) $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (FUN $x\ t$ [x_1:t_1, ... ,x_n:t_n] bk)
$\rightsquigarrow (\rho[x = inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma, \omega)$

(FUNPR) $\rho, \sigma, \omega \vdash_{\text{Dec}}$ (FUN REC $x\ t$ [x_1:t_1, ... ,x_n:t_n] bk)
$\rightsquigarrow (\rho[x = inPR(\lambda f.inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n][x = f]), \sigma, \omega)$

3.9 Suite de commandes

(STAT0) si $\rho, \sigma, \omega \vdash_{\text{Stat}} s \rightsquigarrow (\varepsilon, \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Cmds}} cs \rightsquigarrow (v, \sigma'', \omega'')$ alors $\rho, \sigma, \omega \vdash_{\text{Cmds}} (s;cs) \rightsquigarrow (v, \sigma'', \omega'')$

(STAT1) si $\rho, \sigma, \omega \vdash_{\text{Stat}} s \rightsquigarrow (v, \sigma', \omega')$ avec $v \neq \varepsilon$ alors $\rho, \sigma, \omega \vdash_{\text{Cmds}} (s;cs) \rightsquigarrow (v, \sigma', \omega')$

(DEC) si $\rho, \sigma, \omega \vdash_{\text{Dec}} d \rightsquigarrow (\rho', \sigma', \omega')$ et si $\rho', \sigma', \omega' \vdash_{\text{Cmds}} cs \rightsquigarrow (v, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Cmds}} (d;\ cs) \rightsquigarrow (v, \sigma'', \omega'')$

(RET) si $\rho, \sigma, \omega \vdash_{\text{Ret}} r \rightsquigarrow (v, \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Cmds}} (r;\varepsilon) \rightsquigarrow (v, \sigma', \omega')$

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

3.10 Bloc

(BLOCK0) si $\rho, \sigma, \omega \vdash_{\text{Cmds}} (cs;\varepsilon) \rightsquigarrow (\varepsilon, \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Block}}$ [cs] $\rightsquigarrow (v, (\sigma'/\rho), \omega')$

(BLOCK1) si $\rho, \sigma, \omega \vdash_{\text{Cmds}} (cs;\varepsilon) \rightsquigarrow (v, \sigma', \omega')$ avec $v \neq \varepsilon$
alors $\rho, \sigma, \omega \vdash_{\text{Block}}$ [cs] $\rightsquigarrow (v, (\sigma'/\rho[\delta = v]), \omega')$

3.11 Programme

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