UPMC/master/info/4I503 APS
Formulaire

P. Manoury

Janvier 2018

Contents

1 APS0

1.1 Syntaxe

1.1.1 Lexique

Symboles réservés `[ ] ( ) ; , * ->`

Mots clef

```
CONST FUN REC  ECHO
bool int
true false not and or
eq lt add sub mul div
if
```

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.

- oprim l'ensemble de mots clef: `not and or eq lt add sub mul div`

- tprim l'ensemble de mots clefs `bool int`

Séparateurs: l'espace, la tabulation, le passage à la ligne et le retour chariot.

1.1.2 Grammaire

PROG	::=	`[` CMDS `]`
CMDS	::=	STAT
	|	DEC `;` CMDS
	|	STAT `;` CMDS
TYPE	::=	tprim
	|	`(` TYPES `->` 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
STAT	::=	`ECHO` EXPR
EXPR	::=	bool | num | ident
	|	`( if` EXPR EXPR EXPR `)`
	|	`(` oprim EXPRS `)`
	|	`[` ARGS `]` EXPR
	|	`(` EXPR EXPRS `)`
EXPRS	::=	EXPR
	|	EXPR EXPRS

1.2 Typage

1.2.1 Contextes de typage

- sym = bool ∪ oprim ∪ ident

- $G = \mathsf{sym} \to \textsc{Type}$
- Extension
  - $\Gamma[x:t](x) = t$
  - $\Gamma[x:t](y) = \Gamma(y)$ lorsque x et y sont des symboles différents.

Abréviation: $\Gamma[x_1 : t_1; \ldots; x_n : t_n]$ pour $\Gamma[x_1 : t_1] \ldots [x_n : t_n]$.

Contexte initial:

$$
\begin{array}{lcl}
\Gamma_0(\texttt{true}) & = & \texttt{bool} \\
\Gamma_0(\texttt{false}) & = & \texttt{bool} \\
\Gamma_0(\texttt{not}) & = & \texttt{bool -> bool} \\
\Gamma_0(\texttt{and}) & = & \texttt{bool * bool -> bool} \\
\Gamma_0(\texttt{or}) & = & \texttt{bool * bool -> bool} \\
\Gamma_0(\texttt{eq}) & = & \texttt{int * int -> bool} \\
\Gamma_0(\texttt{lt}) & = & \texttt{int * int -> bool} \\
\Gamma_0(\texttt{add}) & = & \texttt{int * int -> int} \\
\Gamma_0(\texttt{sub}) & = & \texttt{int * int -> int} \\
\Gamma_0(\texttt{mul}) & = & \texttt{int * int -> int} \\
\Gamma_0(\texttt{div}) & = & \texttt{int * int -> int}
\end{array}
$$

1.2.2 Jugements de typages

- Commande vide: ε
- Suites de commandes terminées par la commande vide : $\textsc{Cmds}_\varepsilon$

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

1.2.3 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 $\Gamma \vdash_{\textsc{Expr}} e_1 : t_1, \ldots$ 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 $\Gamma \vdash_{\textsc{Expr}} \texttt{(if } e_1\ e_2\ e_3\texttt{)} : t$

1.2.4 Instruction

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

1.2.5 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 -> $t]$

1.2.6 Suite de commandes

(DECS) si $d \in$ DEC, si $\Gamma \vdash_{\text{Dec}} d : \Gamma'$ et si $\Gamma' \vdash_{\text{Cmds}} cs : void$ alors $\Gamma \vdash_{\text{Cmds}} (d;\ cs) : void$.

(STATS) si $s \in$ STAT, si $\Gamma \vdash_{\text{Stat}} s : void$ et si $\Gamma \vdash_{\text{Cmds}} cs : void$ alors $\Gamma \vdash_{\text{Cmds}} (s;\ cs) : void$.

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

1.2.7 Programme

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

1.3 Sémantique

1.3.1 Domaines sémantiques

- Valeurs: $V = N \oplus F \oplus FR$
- Valeurs immédiates: N (entiers)
- Fermetures: $F = \text{Expr} \times (V^* \to E)$
- Fermetures récursives: $FR = F \to F$
- Environnements: $E = \mathsf{ident} \to V$
- Flux de sortie: $O = N^*$

1.3.2 Opérations sémantiques

- Extension des environnements: $\rho[x = v](x) = v$ et $\rho[x = v](y) = \rho(y)$ lorsque x et y sont des symboles différents.
- Ajout au flux de sortie: $n \cdot \omega$

Opérateurs primitifs:

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

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

1.3.3 Relations sémantiques

	Symbole	Domaine	Notation
Programme	$\vdash$	$\textsc{Prog} \times O$	$\vdash p \rightsquigarrow \omega$
Suite de commandes	$\vdash_{\textsc{Cmds}}$	$E \times O \times \textsc{Cmds}_\varepsilon \times O$	$\rho, \omega \vdash_{\textsc{Cmds}} cs \rightsquigarrow \omega'$
Déclaration	$\vdash_{\textsc{Dec}}$	$E \times \textsc{Dec} \times E$	$\rho \vdash_{\textsc{Dec}} d : \rho'$
Instruction	$\vdash_{\textsc{Stat}}$	$E \times O \times \textsc{Stat} \times O$	$\rho, \omega \vdash_{\textsc{Stat}} s \rightsquigarrow \omega'$
Expression	$\vdash_{\textsc{Expr}}$	$E \times \textsc{Expr} \times V$	$\rho \vdash_{\textsc{Expr}} e \rightsquigarrow v$

1.3.4 Expression

(TRUE) $\rho \vdash_{\textsc{Expr}} \texttt{true} \rightsquigarrow inN(1)$

(FALSE) $\rho \vdash_{\textsc{Expr}} \texttt{false} \rightsquigarrow inN(0)$

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

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

(PRIM) si $x \in \mathsf{oprim}$, si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inN(n_1)$, ..., si $\rho \vdash_{\textsc{Expr}} e_k \rightsquigarrow inN(n_k)$ et si $\pi(x)(n_1, \ldots, n_k) = n$
alors $\rho \vdash_{\textsc{Expr}} \texttt{(}x\ e_1 \ldots e_n\texttt{)} \rightsquigarrow inN(n)$

(IF1) si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inN(1)$ et si $\rho \vdash_{\textsc{Expr}} e_2 \rightsquigarrow v$ alors $\rho \vdash_{\textsc{Expr}} \texttt{(if}\ e_1\ e_2\ e_3\texttt{)} \rightsquigarrow v$

(IF0) si $\rho \vdash_{\textsc{Expr}} e_1 \rightsquigarrow inN(0)$ et si $\rho \vdash_{\textsc{Expr}} e_3 \rightsquigarrow v$ alors $\rho \vdash_{\textsc{Expr}} \texttt{(if}\ e_1\ e_2\ e_3\texttt{)} \rightsquigarrow v$

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

(APP) si $\rho \vdash_{\textsc{Expr}} e \rightsquigarrow inF(e', r)$, 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 $\rho \vdash \texttt{(}e\ e_1 \ldots e_n\texttt{)} \rightsquigarrow v$

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

1.3.5 Instruction

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

1.3.6 Déclaration

(CONST) si $\rho \vdash_{\text{EXPR}} e \rightsquigarrow v$ alors $\rho \vdash_{\text{DEC}}$ (CONST $x\ t\ e$) $\rightsquigarrow \rho[x = v]$

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

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

1.3.7 Suite de commandes

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

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

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

1.3.8 Programme

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

2 APS1

2.1 Syntaxe

2.1.1 Lexique

Mots clef VAR PROC
SET IF WHILE CALL

2.1.2 Grammaire

DEC	::=	...
	|	VAR ident TYPE
	|	PROC ident [ARGS] PROG
	|	PROC REC ident [ARGS] PROG
STAT	::=	...
	|	SET ident EXPR
	|	IF EXPR PROG PROG
	|	WHILE EXPR PROG
	|	CALL ident EXPRS
TYPE	::=	...
	|	void

2.2 Typage

2.2.1 Déclaration

(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$]$

2.2.2 Instruction

(SET) si $\Gamma(x) = t$ et si $\Gamma \vdash_{\text{Expr}} e : t$ alors $\Gamma \vdash_{\text{Stat}}$ (SET x e) : void

(IF) si $\Gamma \vdash_{\text{Expr}} e$: bool, si $\Gamma \vdash_{\text{Cmds}} (cs_1;\varepsilon)$: void et si $\Gamma \vdash_{\text{Cmds}} (cs_2;\varepsilon)$: void
alors $\Gamma \vdash_{\text{Stat}}$ (IF e [cs_1] [cs_2]) : void

(WHILE) si $\Gamma \vdash_{\text{Expr}} e$: bool et si $\Gamma \vdash_{\text{Cmds}} (cs;\varepsilon)$: void alors $\Gamma \vdash_{\text{Stat}}$ (WHILE e [cs]) : void

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

2.3 Sémantique

2.3.1 Domaines sémantiques

Adresse A

Mémoire $S = A \to N$ (fonction partielle)

Fermetures procédurales $P = \text{Cmds} \times (V^* \to E)$

Fermetures procédurales récursives $PR = P \to P$

Valeurs $V\oplus = A \oplus P \oplus PR$

2.3.2 Opérations sémantiques

- Allocation: $alloc(\sigma) = (a, \sigma')$ si et seulement si $a \notin \mathsf{dom}(\sigma)$ et $\sigma' = \sigma[a = any$
- 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: $(\sigma/\rho)(a) = \sigma(a)$ si et seulement il existe x (élément de ident) tel que $\rho(x) = inA(a)$.

2.3.3 Relations sémantiques

	Symbole	Domaine	Notation
Programme	$\vdash$	$\text{Prog} \times S \times O$	$\vdash$ [cs] $\rightsquigarrow (\sigma, \omega)$
Bloc	$\vdash_{\text{Block}}$	$E \times S \times O \times \text{Prog} \times S \times O$	$\rho, \sigma, \omega \vdash bk \rightsquigarrow (\sigma', \omega')$
Suite de commandes	$\vdash_{\text{Cmds}}$	$E \times S \times O \times (\text{Cmds}_\varepsilon) \times S \times O.$	$\rho, \sigma, \omega \vdash_{\text{Cmds}} cs \rightsquigarrow (\sigma', \omega')$
Déclaration	$\vdash_{\text{Dec}}$	$E \times S \times \text{Dec} \times E \times S$	$\rho, \sigma \vdash_{\text{Dec}} d \rightsquigarrow (\rho', \sigma')$
Instruction	$\vdash_{\text{Stat}}$	$E \times S \times O \times \text{Stat} \times S \times O$	$\rho, \sigma, \omega \vdash_{\text{Stat}} s \rightsquigarrow (\sigma', \omega')$
Expression	$\vdash_{\text{Expr}}$	$E \times S \times \text{Expr} \times V$	$\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow v$

2.3.4 Expressions

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

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

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

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

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

(PRIM) si $x \in \mathsf{oprim}$, si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inN(n_1)$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_k \rightsquigarrow inN(n_k)$ et si $\pi(x)(n_1, \ldots, n_k) = n$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(}x\ e_1 \ldots e_n\texttt{)} \rightsquigarrow inN(n)$

(IF1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inN(1)$ et si $\rho, \sigma \vdash_{\text{Expr}} e_2 \rightsquigarrow v$ alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(if}\ e_1\ e_2\ e_3\texttt{)} \rightsquigarrow v$

(IF0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inN(0)$ et si $\rho, \sigma \vdash_{\text{Expr}} e_3 \rightsquigarrow v$ alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(if}\ e_1\ e_2\ e_3\texttt{)} \rightsquigarrow v$

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

(APP) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inF(e', r)$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$ et si $r(v_1, \ldots, v_n), \sigma \vdash_{\text{Expr}} e' \rightsquigarrow v$
alors $\rho, \sigma \vdash \texttt{(}e\ e_1 \ldots e_n\texttt{)} \rightsquigarrow v$

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

2.3.5 Déclarations

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

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

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

(CONST) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow v$ alors $\rho, \sigma \vdash_{\text{Dec}} (\texttt{CONST}\ x\ t\ e) \rightsquigarrow (\rho[x = v], \sigma)$

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

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

2.3.6 Instructions

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

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

(IF0) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inN(0)$ et si $\rho, \sigma, \omega \vdash_{\text{Block}} bk_2 \rightsquigarrow (\sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) \rightsquigarrow (\sigma', \omega')$

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

(LOOP1) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inN(1)$, si $\rho, \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
et si $\rho, \sigma', \omega' \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma'', \omega'')$

(CALL) si $\rho(x) = inP(bk, r)$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$
et si $r(v_1, \ldots, v_n), \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

(CALLR) si $\rho(x) = inPR(\varphi)$, si $\varphi(\varphi) = inP(bk, r)$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$
et si $r(v_1, \ldots, v_n), \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

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

2.3.7 Suite de commandes

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

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

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

2.3.8 Bloc

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

2.3.9 Programme

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