/SU/FSI/MASTER/INFO/MU4IN503
APS
Formulaire*

P.M.

Janvier 2021

1 APS0

1.1 Syntaxe

Lexique

Symboles réservés
`[ ] ( ) ; : , * ->`

Mots clef
`CONST FUN REC`
`ECHO`
`if`
`bool int`

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.

Remarque : les symboles d'opérateurs primitifs
`true false not and or eq lt add sub mul div`
sont des identificateurs.

Grammaire

Programme
PROG ::= `[` CMDS `]`

Suite de commandes
CMDS ::= STAT
| DEF `;` CMDS

Définition
DEF ::= `CONST` ident TYPE EXPR
| `FUN` ident TYPE `[` ARGS `]` EXPR
| `FUN REC` ident TYPE `[` ARGS `]` EXPR

*Que W.S. soit remercié pour sa relecture attentive.

Type

TYPE	: :=	bool | int
	|	(TYPES -> TYPE)
TYPES	: :=	TYPE
	|	TYPE * TYPES

Paramètre formel

ARGS	: :=	ARG
	|	ARG , ARGS
ARG	: :=	ident : TYPE

Instruction

STAT	: :=	ECHO EXPR

Expression

EXPR	: :=	num
	|	ident
	|	(if EXPR EXPR EXPR)
	|	(EXPR EXPRS)
	|	[ARGS] EXPR

Suite d'expressions

EXPRS	: :=	EXPR
	|	EXPR EXPRS

1.2 Typage

Contexte initial

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

Programmes

$$\vdash [cs] : \texttt{void}$$

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

Suite de commandes

$$\Gamma \vdash_{\text{CMDS}} cs : \texttt{void}$$

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

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

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

Définitions

$$\Gamma \vdash d : \Gamma'$$

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

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

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

Intruction

$$\Gamma \vdash_{\textsc{Stat}} s : \texttt{void}$$

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

Expressions

$$\Gamma \vdash_{\textsc{Expr}} e : t$$

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

(ID) si $x \in \mathsf{ident}$, si $\Gamma(x) = t$
alors $\Gamma \vdash_{\textsc{Expr}} x : t$

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

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

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

1.3 Sémantique

Fonctions primitives

$$
\begin{array}{rcll}
\pi_1(\texttt{not})(0) & = & 1 & \\
\pi_1(\texttt{not})(1) & = & 0 & \\
\\
\pi_2(\texttt{eq})(n_1, n_2) & = & 1 & \text{si } n_1 = n_2 \\
 & = & 0 & \text{sinon} \\
\pi_2(\texttt{lt})(n_1, n_2) & = & 1 & \text{si } n_1 < n_2 \\
 & = & 0 & \text{sinon} \\
\\
\pi_2(\texttt{add})(n_1, n_2) & = & n_1 + n_2 & \\
\pi_2(\texttt{sub})(n_1, n_2) & = & n_1 - n_2 & \\
\pi_2(\texttt{mul})(n_1, n_2) & = & n_1 \times n_2 & \\
\pi_2(\texttt{div})(n_1, n_2) & = & n_1 \div n_2 &
\end{array}
$$

Programmes

$$\vdash [cs] \rightsquigarrow \omega$$

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

Suites de commandes

$$\rho, \omega \vdash_{\text{Cmds}} cs \rightsquigarrow \omega'$$

(DECS) si $\rho \vdash_{\text{Dec}} d \rightsquigarrow \rho'$ 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) $\rho, \omega \vdash_{\text{Cmds}} \varepsilon \rightsquigarrow \omega$

Définitions

$$\rho \vdash_{\text{Dec}} d \rightsquigarrow \rho'$$

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

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

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

Instruction

$$\rho, \omega \vdash_{\text{Stat}} s \rightsquigarrow \omega'$$

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

Expressions

$$\rho \vdash_{\text{Expr}} e \rightsquigarrow v$$

(TRUE) $\rho \vdash_{\text{Expr}} \texttt{true} \rightsquigarrow inZ(1)$

(FALSE) $\rho \vdash_{\text{Expr}} \texttt{false} \rightsquigarrow inZ(0)$

(NUM) si $n \in \textsf{num}$ alors $\rho \vdash_{\text{Expr}} n \rightsquigarrow inZ(\nu(n))$

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

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

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

(AND1) si $\rho \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(1)$ et si $\rho \vdash_{\text{Expr}} e_2 \rightsquigarrow v$
alors $\rho \vdash_{\text{Expr}} \texttt{(and } e_1\ e_2\texttt{)} \rightsquigarrow v$.

(AND0) si $\rho \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(0)$
alors $\rho \vdash_{\text{Expr}} \texttt{(and } e_1\ e_2\texttt{)} \rightsquigarrow inZ(0)$.

(OR1) si $\rho \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(1)$
alors $\rho \vdash_{\text{Expr}} \texttt{(or } e_1\ e_2\texttt{)} \rightsquigarrow inZ(1)$.

(OR0) si $\rho \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(0)$ et si $\rho \vdash_{\text{Expr}} e_2 \rightsquigarrow v$
alors $\rho \vdash_{\text{Expr}} \texttt{(or } e_1\ e_2\texttt{)} \rightsquigarrow v$.

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

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

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

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

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

2 APS1

2.1 Syntaxe

Lexique

Symboles réservés

`[ ] ( ) ; : , * ->`

Mots clef

```
 CONST FUN REC VAR PROC
ECHO SET IF WHILE CALL
if
bool int
```

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.

Remarque : les symboles d'opérateurs primitifs

`not and or eq lt   add sub mul div`

sont des identificateurs.

Grammaire

Programme

```
PROG    ::=  BLOCK
```

Bloc

```
BLOCK   ::=  [ CMDS ]
```

Suite de commandes

```
CMDS    ::=  STAT
          |  DEF ; CMDS
          |  STAT ; CMDS
```

Définition

```
DEF     ::=  CONST ident TYPE EXPR
          |  FUN ident TYPE [ ARGS ] EXPR
          |  FUN REC ident TYPE [ ARGS ] EXPR
          |  VAR ident TYPE
          |  PROC ident [ ARGS ] BLOCK
          |  PROC REC ident [ ARGS ] BLOCK
```

Type

```
TYPE    ::=  bool | int
          |  ( TYPES -> TYPE )
TYPES   ::=  TYPE
          |  TYPE * TYPES
```

Paramètre formel

```
ARGS    ::=  ARG
          |  ARG , ARGS
ARG     ::=  ident : TYPE
```

Instruction

```
STAT    ::=  ECHO EXPR
          |  SET ident EXPR
          |  IF EXPR BLOCK BLOCK
          |  WHILE EXPR BLOCK
          |  CALL ident EXPRS
```

Expression

```
EXPR    ::=  num
          |  ident
          |  (if EXPR EXPR EXPR )
          |  ( EXPR EXPRS )
          |  [ ARGS ] EXPR
```

Suite d'expressions

```
EXPRS   ::=  EXPR
          |  EXPR EXPRS
```

2.2 Typage

Programmes

(PROG) si $\Gamma_0 \vdash_{\text{Block}} bk : \texttt{void}$
alors $\vdash bk : \texttt{void}$

Blocs

(BLOC) si $\Gamma \vdash_{\text{Cmds}} (cs;\varepsilon) : \texttt{void}$
alors $\Gamma \vdash_{\text{Block}} \texttt{[}cs\texttt{]} : \texttt{void}$

Suite de commandes

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

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

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

Définitions

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

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

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

(VAR) si $t \in \{\texttt{int}, \texttt{bool}\}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{VAR}\ x\ t) : \Gamma[x : t]$

(PROC) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC}\ x\ \texttt{[}x_1\texttt{:}t_1, \ldots, x_n\texttt{:}t_n\texttt{]}bk) : \Gamma[x : t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ \texttt{void}]$

(PROCREC)
si $\Gamma[x_1 : t_1; \ldots; x_n : t_n; x : t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ \texttt{void}] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC REC}\ x\ \texttt{[}x_1\texttt{:}t_1, \ldots, x_n\texttt{:}t_n\texttt{]}bk) : \Gamma[x : t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ \texttt{void}]$

Intructions

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

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

(IF) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk_1 : \texttt{void}$ et si $\Gamma \vdash_{\text{Block}} bk_2 : \texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) : \texttt{void}$

(WHILE) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) : \texttt{void}$

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

Expressions

(NUM) si $n \in$ `num`
alors $\Gamma \vdash_{\text{Expr}} n :$ `int`

(ID) si $x \in$ ident, si $\Gamma(x) = t$
alors $\Gamma \vdash_{\text{Expr}} x : t$

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

(APP) si $\Gamma \vdash_{\text{Expr}} e :$ `(`t_1 `* ... *` t_n `->` t`)`,
si $\Gamma \vdash_{\text{Expr}} e_1 : t_1$, ..., si $\Gamma \vdash_{\text{Expr}} e_n : t_n$
alors $\Gamma \vdash_{\text{Expr}}$ `(`$e\ e_1 \dots e_n$`)` $: t$

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

2.3 Sémantique

Programmes

$$\vdash p \rightsquigarrow (\sigma, \omega)$$

(PROG) si $\varepsilon, \varepsilon, \varepsilon \vdash_{\text{Block}} bk \rightsquigarrow \omega$
alors $\vdash bk \rightsquigarrow (\sigma, \omega)$

Blocs

$$\rho, \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$$

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

Suites de commandes

$$\rho, \sigma, \omega \vdash_{\text{Cmds}} cs \rightsquigarrow (\sigma', \omega')$$

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

(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 $\rho, \sigma, \omega \vdash_{\text{Cmds}} (s;\ cs) \rightsquigarrow (\sigma'', \omega'')$

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

Définitions

$$\rho, \sigma \vdash_{\text{Dec}} d \rightsquigarrow (\rho', \sigma')$$

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

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

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

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

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

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

Instructions

$$\rho, \sigma, \omega \vdash_{\text{Stat}} s \rightsquigarrow (\sigma', \omega')$$

(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 $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(1)$ 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 $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(0)$ 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 $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(0)$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma, \omega)$

(LOOP1) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(1)$, 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 $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$
si $\rho' = r(v_1, \ldots, v_n)$ et $\rho', \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

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

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

Expressions

$$\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow v$$

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

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

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

(ID1) si $\rho(x) = inA(a)$
alors $\rho, \sigma \vdash_{\text{Expr}} x \rightsquigarrow inZ(\sigma(a))$

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

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

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

(AND1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(1)$ et si $\rho, \sigma \vdash_{\text{Expr}} e_2 \rightsquigarrow v$
alors $\rho, \sigma \vdash_{\text{Expr}}$ (and e_1 e_2) $\rightsquigarrow v$.

(AND0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(0)$
alors $\rho, \sigma \vdash_{\text{Expr}}$ (and e_1 e_2) $\rightsquigarrow inZ(0)$.

(OR1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(1)$
alors $\rho, \sigma \vdash_{\text{Expr}}$ (or e_1 e_2) $\rightsquigarrow inZ(1)$.

(OR0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(0)$ et si $\rho, \sigma \vdash_{\text{Expr}} e_2 \rightsquigarrow v$
alors $\rho, \sigma \vdash_{\text{Expr}}$ (or e_1 e_2) $\rightsquigarrow v$.

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

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

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

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

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

3 APS1a

3.1 Syntaxe

Lexique

Symboles réservés

`[ ] ( ) ; : , * ->`

Mots clef

`CONST FUN REC VAR PROC   ECHO SET IF WHILE CALL`
`if`
`bool int`
`var adr`

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.

Remarque : les symboles d'opérateurs primitifs
`not and or eq lt   add sub mul div`
sont des identificateurs.

Grammaire

Programme

PROG ::= BLOCK

Bloc

```
Block  ::=  [ Cmds ]
```

Suite de commandes

```
Cmds   ::=  Stat
         |  Def ; Cmds
         |  Stat ; Cmds
```

Définition

```
Def    ::=  CONST ident Type Expr
         |  FUN ident Type [ Args ] Expr
         |  FUN REC ident Type [ Args ] Expr
         |  VAR ident Type
         |  PROC ident [ ArgsP ] Block
         |  PROC REC ident [ ArgsP ] Block
```

Type

```
Type   ::=  bool | int
         |  ( Types -> Type )
Types  ::=  Type
         |  Type * Types
```

Paramètre formel (fonctions)

```
Args   ::=  Arg
         |  Arg , Args
Arg    ::=  ident : Type
```

Paramètre formel (procédure)

```
ArgsP ::=   ::=  ArgP
              |  ArgP , ArgsP
ArgP        ::=  ident : Type
              |  var ident : Type
```

Instruction

```
Stat   ::=  ECHO Expr
         |  SET ident Expr
         |  IF Expr Block Block
         |  WHILE Expr Block
         |  CALL ident ExprsP
```

Paramètres d'appel

```
ExprsP ::=  ExprP
         |  ExprP ExprsP
ExprP  ::=  Expr
         |  (adr ident)
```

Expression

```
Expr   ::=  num
         |  ident
         |  (if Expr Expr Expr )
         |  ( Expr Exprs )
         |  [ Args ] Expr
```

Suite d'expressions

```
Exprs  ::=  Expr
         |  Expr Exprs
```

3.2 Typage

Soit $p1, \ldots, p_n \in$ ARGSP.
Posons $A([p_1 : t_1, \ldots, x_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$ avec
$t'_i = \begin{cases} t_i & \text{si } p_i = x_i \\ (\texttt{ref}\ t_i) & \text{si } p_i = \texttt{var}\ x_i \end{cases}$

Programmes

(PROG) si $\Gamma_0 \vdash_{\text{Block}} bk : \texttt{void}$
alors $\vdash bk : \texttt{void}$

Blocs

(BLOC) si $\Gamma \vdash_{\text{Cmds}} (cs\texttt{;}\varepsilon) : \texttt{void}$
alors $\Gamma \vdash_{\text{Block}} \texttt{[}cs\texttt{]} : \texttt{void}$

Suite de commandes

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

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

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

Définitions

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

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

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

(VAR) si $t \in \{\texttt{int}, \texttt{bool}\}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{VAR}\ x\ t) : \Gamma[x : (\texttt{ref}\ t)]$

(PROC) si $A([p_1 : t_1, \ldots, p_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$
si $\Gamma[x_1 : t'_1; \ldots; x_n : t'_n] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC}\ x\ \texttt{[}p_1\texttt{:}t_1, \ldots, p_n\texttt{:}t_n\texttt{]}bk) : \Gamma[x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}]$

(PROCREC)
si $A([p_1 : t_1, \ldots, x_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$
si $\Gamma[x_1 : t'_1; \ldots; x_n : t'_n; x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC REC}\ x\ \texttt{[}p_1\texttt{:}t_1, \ldots, p_n\texttt{:}t_n\texttt{]}bk) : \Gamma[x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}]$

Intructions

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

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

(IF) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk_1 : \texttt{void}$ et si $\Gamma \vdash_{\text{Block}} bk_2 : \texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) : \texttt{void}$

(WHILE) si $\Gamma \vdash_{\text{Expr}} e$: $\texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk$: $\texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) : \texttt{void}$

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

3.3 Paramètres d'appel

(REF) si $\Gamma(x) = (\texttt{ref}\ \ t)$
alors $\Gamma \vdash_{\text{Expar}} (\texttt{adr}\ \ x) : (\texttt{ref}\ \ t)$

(VAL) si $\Gamma \vdash_{\text{Expr}} e : t$
alors $\Gamma \vdash_{\text{Expar}} e : t$

Expressions

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

(IDV) si $x \in \textsf{ident}$, si $\Gamma(x) = t$ avec $t \neq (\texttt{ref}\ \ t')$
alors $\Gamma \vdash_{\text{Expr}} x : t$

(IDR) si $x \in \textsf{ident}$,
si $\Gamma(x) = (\texttt{ref}\ \ t)$
alors $\Gamma \vdash_{\text{Expr}} x : t$

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

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

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

3.4 Sémantique

Programmes

(PROG) si $\varepsilon, \varepsilon \vdash_{\text{Block}} bk \rightsquigarrow \omega$
alors $\vdash bk \rightsquigarrow (\sigma, \omega)$

Blocs

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

Suites 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 $\rho, \sigma, \omega \vdash_{\text{Cmds}} (d;\ \ cs) \rightsquigarrow (\sigma'', \omega')$

(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 $\rho, \sigma, \omega \vdash_{\text{Cmds}} (s;\ \ cs) \rightsquigarrow (\sigma'', \omega'')$

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

Définitions

Soit $p1, \ldots, p_n \in$ ARGSP.
Posons $X([p_1 : t_1, \ldots, x_n : t_n]) = [x_1, \ldots, x_n]$ avec

$$x_i = \begin{cases} x_i & \text{si } p_i = x_i \\ x_i & \text{si } p_i = \texttt{var}\ x_i \end{cases}$$

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

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

(PROC) si $X([p_1 : t_1, \ldots, x_n : t_n]) = [x_1, \ldots, x_n]$,
$\rho, \sigma \vdash_{\text{Dec}} (\texttt{PROC}\ x\ t\ \texttt{[}p_1\texttt{:}t_1\texttt{,}\ \ldots p_n\texttt{:}t_n\texttt{]}\ bk)$
$\rightsquigarrow (\rho[x = inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma)$

(PROCREC) si $X([p_1 : t_1, \ldots, x_n : t_n]) = [x_1, \ldots, x_n]$,
$\rho, \sigma \vdash_{\text{Dec}} (\texttt{PROC REC}\ x\ t\ \texttt{[}p_1\texttt{:}t_1\texttt{,}\ \ldots\ \texttt{,}p_n\texttt{:}t_n\texttt{]}bk)$
$\rightsquigarrow (\rho[x = inPR(\lambda f.inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n][x = f]), \sigma)$

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 $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(1)$ 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 $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(0)$ 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 $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(0)$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma, \omega)$

(LOOP1) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(1)$, 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 $\rho, \sigma \vdash_{\text{Expar}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expar}} e_n \rightsquigarrow v_n$
si $\rho' = r(v_1, \ldots, v_n)$ et $\rho', \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

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

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

Paramètres d'appel

(REF) si $\rho(x) = inA(a)$
alors $\rho, \sigma \vdash_{\text{Expar}} (\texttt{adr}\ x) \rightsquigarrow inA(a)$

(VAL) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow v$
alors $\rho, \sigma \vdash_{\text{Expar}} e \rightsquigarrow v$

Expressions

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

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

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

(ID1) si $\rho(x) = inA(a)$
alors $\rho, \sigma \vdash_{\text{Expr}} x \rightsquigarrow inZ(\sigma(a))$

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

(PRIM1) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inZ(n)$, et si $\pi_1(not)(n) = n'$
alors $\rho, \sigma \vdash_{\text{Expr}}$ `(not` e`)` $\rightsquigarrow inZ(n')$

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

(AND0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(0)$
alors $\rho, \sigma \vdash_{\text{Expr}}$ `(and` $e_1\ e_2$`)` $\rightsquigarrow inZ(0)$.

(AND1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(1)$ et si $\rho, \sigma \vdash_{\text{Expr}} e_2 \rightsquigarrow v$
alors $\rho, \sigma \vdash_{\text{Expr}}$ `(and` $e_1\ e_2$`)` $\rightsquigarrow v$.

(OR1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(1)$
alors $\rho, \sigma \vdash_{\text{Expr}}$ `(or` $e_1\ e_2$`)` $\rightsquigarrow inZ(1)$.

(OR0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow inZ(0)$ et si $\rho, \sigma \vdash_{\text{Expr}} e_2 \rightsquigarrow v$
alors $\rho, \sigma \vdash_{\text{Expr}}$ `(or` $e_1\ e_2$`)` $\rightsquigarrow v$.

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

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

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

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

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

4 APS2

4.1 Syntaxe

Lexique

Symboles réservés

```
[ ] ( ) ; : , * ->
```

Mots clef

```
 CONST FUN REC VAR PROC
ECHO SET IF WHILE CALL
if
bool int vec
var adr
```

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.

Remarque : les symboles d'opérateurs primitifs

`not and or eq lt   add sub mul div   alloc len nth`

sont des identificateurs.

Grammaire

Programme

PROG ::= BLOCK

Bloc

BLOCK ::= [CMDS]

Suite de commandes

```
CMDS  ::= STAT
       |  DEF ; CMDS
       |  STAT ; CMDS
```

Définition

```
DEF  ::= CONST ident TYPE EXPR
      |  FUN ident TYPE [ ARGS ] EXPR
      |  FUN REC ident TYPE [ ARGS ] EXPR
      |  VAR ident STYPE
      |  PROC ident [ ARGSP ] BLOCK
      |  PROC REC ident [ ARGSP ] BLOCK
```

Type

```
TYPE   ::= STYPE
        |  ( TYPES -> TYPE )
TYPES  ::= TYPE
        |  TYPE * TYPES
```

SType

```
STYPE  ::= bool | int
        |  ( vec STYPE )
```

Paramètre formel (fonctions)

```
ARGS  ::= ARG
       |  ARG , ARGS
ARG   ::= ident : TYPE
```

Paramètre formel (procédures)

```
ARGSP ::=    ::= ARGP
              |  ARGP , ARGSP
ARGP         ::= ident : TYPE
              |  var ident : TYPE
```

Instruction

```
STAT  ::= ECHO EXPR
       |  SET LVALUE EXPR
       |  IF EXPR BLOCK BLOCK
       |  WHILE EXPR BLOCK
       |  CALL ident EXPRSP
```

lvalue

$\text{LVALUE} ::= \mathsf{ident}$
$\mid$ `( nth` LVALUE EXPR `)`

Paramètres d'appel

$\text{EXPRSP} ::= \text{EXPRP}$
$\mid \text{EXPRP EXPRSP}$

$\text{EXPRP} ::= \text{EXPR}$
$\mid$ `(adr ident)`

Expression

$\text{EXPR} ::= \mathsf{num}$
$\mid \mathsf{ident}$
$\mid$ `(if` EXPR EXPR EXPR `)`
$\mid$ `(` EXPR EXPRS `)`
$\mid$ `[` ARGS `]` EXPR

Suite d'expressions

$\text{EXPRS} ::= \text{EXPR}$
$\mid \text{EXPR EXPRS}$

4.2 Typage

Soit $p1, \ldots, p_n \in \text{EXPRP}$.
Posons $A([p_1 : t_1, \ldots, x_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$ avec

$$t'_i = \begin{cases} t_i & \text{si } p_i = x_i \\ (\texttt{ref}\ \ t_i) & \text{si } p_i = \texttt{var}\ \ x_i \end{cases}$$

Programmes

(PROG) si $\Gamma_0 \vdash_{\text{BLOCK}} bk : \texttt{void}$
alors $\vdash bk : \texttt{void}$

Blocs

(BLOC) si $\Gamma \vdash_{\text{CMDS}} (cs;\varepsilon) : \texttt{void}$
alors $\Gamma \vdash_{\text{BLOCK}} \texttt{[}cs\texttt{]} : \texttt{void}$

Suite de commandes

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

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

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

Définitions

(CONST) si $\Gamma \vdash_{\text{EXPR}} e : t$
alors $\Gamma \vdash_{\text{DEC}} (\texttt{CONST}\ x\ t\ e) : \Gamma[x : t]$

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

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

(VAR) si $t \in \{\texttt{int}, \texttt{bool}\}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{VAR}\ x\ t) : \Gamma[x : (\texttt{ref}\ \ t)]$

(PROC) si $A([p_1 : t_1, \ldots, p_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$
si $\Gamma[x_1 : t'_1; \ldots; x_n : t'_n] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC}\ x\ \texttt{[}p_1 \texttt{:} t_1, \ldots, p_n \texttt{:} t_n\texttt{]}\, bk) : \Gamma[x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}]$

(PROCREC)
si $A([p_1 : t_1, \ldots, x_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$
si $\Gamma[x_1 : t'_1; \ldots; x_n : t'_n; x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC REC}\ x\ \texttt{[}p_1 \texttt{:} t_1, \ldots, p_n \texttt{:} t_n\texttt{]}\, bk) : \Gamma[x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}]$

Instructions

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

(SET) si $\Gamma \vdash_{\text{Lval}} e_1 : t$ et si $\Gamma \vdash_{\text{Expr}} e_2 : t$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{SET}\ e_1\ e_2) : \texttt{void}$

(IF) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk_1 : \texttt{void}$ et si $\Gamma \vdash_{\text{Block}} bk_2 : \texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) : \texttt{void}$

(WHILE) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) : \texttt{void}$

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

lvalue

(LVAR) si $\Gamma(x) = \texttt{(ref}\ \ t\texttt{)}$
alors $\Gamma \vdash_{\text{Lval}} x : t$

(LNTH) si $\Gamma \vdash_{\text{Expr}} e_1 : \texttt{(vec}\ \ t\texttt{)}$ et $\Gamma \vdash_{\text{Expr}} e_2 : \texttt{int}$
alors $\Gamma \vdash_{\text{Lval}} \texttt{(nth}\ \ e_1\ e_2\texttt{)} : t$

Expressions

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

(IDV) si $x \in \mathsf{ident}$, si $\Gamma(x) = t$ avec $t \neq (\texttt{ref}\ \ t')$
alors $\Gamma \vdash_{\text{Expr}} x : t$

(IDR) si $x \in \mathsf{ident}$, si $\Gamma(x) = (\texttt{ref}\ \ t)$
alors $\Gamma \vdash_{\text{Expr}} x : t$

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

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

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

(ALLOC) si $\Gamma \vdash_{\text{Expr}} e : \texttt{int}$
alors $\Gamma \vdash_{\text{Expr}} \texttt{(alloc}\ \ e\texttt{)} : \texttt{(vec}\ \ t\texttt{)}$

(LEN) si $\Gamma \vdash_{\text{Expr}} e : (\texttt{vec}\ t)$
alors $\Gamma \vdash_{\text{Expr}} (\texttt{len}\ e) : \texttt{int}$

(NTH) si $\Gamma \vdash_{\text{Expr}} e_1 : (\texttt{vec}\ t)$ et si $\Gamma \vdash_{\text{Expr}} e_2 : \texttt{int}$
alors $\Gamma \vdash_{\text{Expr}} (\texttt{nth}\ e_1\ e_2) : t$

4.3 Sémantique

Programmes

(PROG) si $\varepsilon, \varepsilon, \varepsilon \vdash_{\text{Block}} bk \rightsquigarrow \omega$
alors $\vdash bk \rightsquigarrow (\sigma, \omega)$

Blocs

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

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

Définitions

Soit $p1, \ldots, p_n \in$ ARGSP.
Posons $X([p_1 : t_1, \ldots, x_n : t_n]) = [x_1, \ldots, x_n]$ avec
$$x_i = \begin{cases} x_i & \text{si } p_i = x_i \\ x_i & \text{si } p_i = \texttt{var}\ x_i \end{cases}$$

(CONST) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow (v, \sigma')$
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) $\rho, \sigma \vdash_{\text{Dec}} (\texttt{FUN REC}\ x\ t\ \texttt{[}x_1\texttt{:}t_1\texttt{,} \ldots \texttt{,}x_n\texttt{:}t_n\texttt{]}\ e)$
$\rightsquigarrow (\rho[x = inFR(\lambda f.inF(e, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n][x = f]), \sigma)$

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

(PROC) si $X([p_1 : t_1, \ldots, x_n : t_n]) = [x_1, \ldots, x_n]$,
$\rho, \sigma \vdash_{\text{Dec}} (\texttt{PROC}\ x\ t\ \texttt{[}p_1\texttt{:}t_1\texttt{,} \ldots p_n\texttt{:}t_n\texttt{]}\ bk)$
$\rightsquigarrow (\rho[x = inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma)$

(PROCREC) si $X([p_1 : t_1, \ldots, x_n : t_n]) = [x_1, \ldots, x_n]$,
$\rho, \sigma \vdash_{\text{Dec}} (\texttt{PROC REC}\ x\ t\ \texttt{[}p_1\texttt{:}t_1\texttt{,} \ldots \texttt{,}p_n\texttt{:}t_n\texttt{]}bk)$
$\rightsquigarrow (\rho[x = inPR(\lambda f.inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n][x = f]), \sigma)$

Instructions

(SET) si $\rho, \sigma \vdash_{\text{LVAL}} e_1 \rightsquigarrow a$ et si $\rho, \sigma \vdash_{\text{EXPR}} e_2 \rightsquigarrow (v, \sigma')$
alors $\rho, \sigma, \omega \vdash_{\text{STAT}} (\texttt{SET}\ e_1\ e_2) \rightsquigarrow (\sigma'[a := v], \omega)$

(IF1) si $\rho, \sigma \vdash_{\text{EXPR}} e \rightsquigarrow (inZ(1), \sigma')$ 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 $\rho, \sigma \vdash_{\text{EXPR}} e \rightsquigarrow (inZ(0), \sigma')$ 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 $\rho, \sigma \vdash_{\text{EXPR}} e \rightsquigarrow (inZ(0), \sigma')$
alors $\rho, \sigma, \omega \vdash_{\text{STAT}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma', \omega)$

(LOOP1) si $\rho, \sigma \vdash_{\text{EXPR}} e \rightsquigarrow (inZ(1), \sigma')$, 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 $\rho, \sigma \vdash_{\text{EXPAR}} e_1 \rightsquigarrow (v_1, \sigma_1)$, ..., si $\rho, \sigma_{n-1} \vdash_{\text{EXPAR}} e_n \rightsquigarrow (v_n, \sigma_n)$
si $\rho' = r(v_1, \ldots, v_n)$ et $\rho', \sigma_n, \omega \vdash_{\text{BLOCK}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

(CALLR) si $\rho(x) = inPR(\varphi)$, si $\varphi(inPR(\varphi)) = inP(bk, r)$,
si $\rho, \sigma \vdash_{\text{EXPAR}} e_1 \rightsquigarrow (v_1, \sigma_1)$, ..., si $\rho, \sigma_{n-1} \vdash_{\text{EXPAR}} e_n \rightsquigarrow (v_n, \sigma_n)$
et si $\rho' = r(v_1, \ldots, v_n)$ et $\rho', \sigma_n, \omega \vdash_{\text{BLOCK}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

(ECHO) si $\rho, \sigma, \omega \vdash_{\text{EXPR}} e \rightsquigarrow (inZ(n), \sigma')$
alors $\rho, \sigma, \omega \vdash_{\text{STAT}} (\texttt{ECHO}\ e) \rightsquigarrow (\sigma', n \cdot \omega)$

lvalue

(LID0) si $x \in ident$, si $\rho(x) = inA(a)$
alors $\rho, \sigma \vdash_{\text{LVAL}} x \rightsquigarrow a$

(LID1) si $x \in ident$, si $\rho(x) = inB(a)$
alors $\rho, \sigma \vdash_{\text{LVAL}} x \rightsquigarrow a + 1$

(LNTH1) si $\rho, \sigma \vdash_{\text{LVAL}} e_1 \rightsquigarrow a$, si $\rho, \sigma \vdash_{\text{EXPR}} e_2 \rightsquigarrow (inZ(i); \sigma')$ et si $\sigma'(a + i) = InZ(n)$
alors $\rho, \sigma \vdash_{\text{LVAL}} (\texttt{nth}\ e_1\ e_2) \rightsquigarrow (a + i)$

(LNTH2) si $\rho, \sigma \vdash_{\text{LVAL}} e_1 \rightsquigarrow a$, si $\rho, \sigma \vdash_{\text{EXPR}} e_2 \rightsquigarrow (inZ(i), \sigma')$ et si $\sigma'(a + i) = InB(a')$
alors $\rho, \sigma \vdash_{\text{LVAL}} (\texttt{nth}\ e_1\ e_2) \rightsquigarrow (a' + 1)$

Paramètres d'appel

(REF) si $\rho(x) = inA(a)$
alors $\rho, \sigma \vdash_{\text{EXPAR}} (\texttt{adr}\ x) \rightsquigarrow (inA(a), \sigma)$

(VAL) si $\rho, \sigma \vdash_{\text{EXPR}} e \rightsquigarrow (v, \sigma')$
alors $\rho, \sigma \vdash_{\text{EXPAR}} e \rightsquigarrow (v, \sigma')$

Expressions

(TRUE) $\rho, \sigma \vdash_{\text{EXPR}} \texttt{true} \rightsquigarrow (inZ(1), \sigma)$

(FALSE) $\rho, \sigma \vdash_{\text{EXPR}} \texttt{false} \rightsquigarrow (inZ(0), \sigma)$

(NUM) si $n \in \texttt{num}$ alors $\rho, \sigma \vdash_{\text{EXPR}} n \rightsquigarrow (inZ(\nu(n)), \sigma)$

(ID1) si $x \in \mathsf{ident}$ et $\rho(x) = inA(a)$
alors $\rho, \sigma \vdash_{\text{EXPR}} x \rightsquigarrow (inZ(\sigma(a)), \sigma)$

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

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

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

(AND0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow (inZ(0), \sigma')$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(and } e_1\ e_2\texttt{)} \rightsquigarrow (inZ(0), \sigma')$.

(AND1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow (inZ(1), \sigma')$ et si $\rho, \sigma' \vdash_{\text{Expr}} e_2 \rightsquigarrow (v, \sigma'')$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(and } e_1\ e_2\texttt{)} \rightsquigarrow (v, \sigma'')$.

(OR1) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow (inZ(1), \sigma')$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(or } e_1\ e_2\texttt{)} \rightsquigarrow (inZ(1), \sigma')$.

(OR0) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow (inZ(0)\sigma')$ et si $\rho, \sigma' \vdash_{\text{Expr}} e_2 \rightsquigarrow (v, \sigma'')$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(or } e_1\ e_2\texttt{)} \rightsquigarrow (v, \sigma'')$.

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

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

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

(APP) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inF(e', r)$, si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1)$, ..., si $\rho, \sigma_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n)$,
si $\rho' = r(v_1, \ldots, v_n)$ et si $\rho', \sigma_n \vdash_{\text{Expr}} e' \rightsquigarrow (v, \sigma')$
alors $\rho, \sigma \vdash \texttt{(}e\ e_1 \ldots e_n\texttt{)} \rightsquigarrow (v, \sigma')$

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

(ALLOC) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow (inZ(n), \sigma')$, avec $n > 0$, et si $allocb(\sigma', n) = (a, \sigma'')$, avec $\sigma'' = \sigma'[a = inZ(n)]$,
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(alloc } e\texttt{)} \rightsquigarrow (inB(a), \sigma'')$

(LEN) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow (inB(a), \sigma')$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(len } e\texttt{)} \rightsquigarrow (\sigma'(a), \sigma')$

(NTH) si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow (inB(a), \sigma')$ et si $\rho, \sigma' \vdash_{\text{Expr}} e_2 \rightsquigarrow (inZ(i), \sigma'')$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(nth } e_1\ e_2\texttt{)} \rightsquigarrow (\sigma''(a + i + 1), \sigma'')$

5 APS3

5.1 Syntaxe

Lexique

Symboles réservés

```
[ ] ( ) ; : , * ->
```

Mots clef

```
CONST FUN REC VAR PROC  ECHO SET IF WHILE CALL
RETURN
if
bool int vec
var adr
```

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.

Remarque : les symboles d'opérateurs primitifs

`not and or eq lt   add sub mul div   alloc len nth`

sont des identificateurs.

Grammaire

Programme

```
PROG   ::=  BLOCK
```

Bloc

```
BLOCK  ::=  [ CMDS ]
```

Suite de commandes

```
CMDS   ::=  STAT
         |  RETURN EXPR
         |  DEF ; CMDS
         |  STAT ; CMDS
```

Définition

```
DEF    ::=  CONST ident TYPE EXPR
         |  FUN ident TYPE [ ARGS ] EXPR
         |  FUN REC ident TYPE [ ARGS ] EXPR
         |  VAR ident STYPE
         |  PROC ident [ ARGSP ] BLOCK
         |  PROC REC ident [ ARGSP ] BLOCK
         |  FUN ident TYPE [ ARGSP ] BLOC
         |  FUN REC ident TYPE [ ARGSP ] BLOC
```

Type

```
TYPE   ::=  STYPE
         |  ( TYPES -> TYPE )
TYPES  ::=  TYPE
         |  TYPE * TYPES
```

SType

```
STYPE  ::=  bool | int
         |  ( vec STYPE )
```

Paramètre formel (fonctions pures)

```
ARGS   ::=  ARG
         |  ARG , ARGS
ARG    ::=  ident : TYPE
```

Paramètre formel (procédure et fonctions procédurales)

```
ARGSP ::=   ::=  ARGP
              |  ARGP , ARGSP
ARGP         ::=  ident : TYPE
              |  var ident : TYPE
```

Instruction

$$
\begin{array}{lll}
\textsc{Stat} & ::= & \texttt{ECHO}\ \textsc{Expr} \\
& | & \texttt{SET}\ \textsc{Lvalue}\ \textsc{Expr} \\
& | & \texttt{IF}\ \textsc{Expr}\ \textsc{Block}\ \textsc{Block} \\
& | & \texttt{WHILE}\ \textsc{Expr}\ \textsc{Block} \\
& | & \texttt{CALL}\ \mathsf{ident}\ \textsc{Exprsp}
\end{array}
$$

lvalue

$$
\begin{array}{lll}
\textsc{Lvalue} & ::= & \mathsf{ident} \\
& | & (\ \texttt{nth}\ \textsc{Lvalue}\ \textsc{Expr}\ )
\end{array}
$$

Paramètres d'appel

$$
\begin{array}{lll}
\textsc{Exprsp} & ::= & \textsc{Exprp} \\
& | & \textsc{Exprp}\ \textsc{Exprsp} \\
\textsc{Exprp} & ::= & \textsc{Expr} \\
& | & (\texttt{adr}\ \texttt{Lvalue})
\end{array}
$$

Expression

$$
\begin{array}{lll}
\textsc{Expr} & ::= & \mathsf{num} \\
& | & \mathsf{ident} \\
& | & (\texttt{if}\ \textsc{Expr}\ \textsc{Expr}\ \textsc{Expr}\ ) \\
& | & (\ \textsc{Expr}\ \textsc{Exprsp}\ ) \\
& | & [\ \textsc{Args}\ ]\ \textsc{Expr}
\end{array}
$$

Suite d'expressions

$$
\begin{array}{lll}
\textsc{Exprs} & ::= & \textsc{Expr} \\
& | & \textsc{Expr}\ \textsc{Exprs}
\end{array}
$$

5.2 Typage

Soit $p1, \ldots, p_n \in$ EXPRP.
Posons $A([p_1 : t_1, \ldots, x_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$ avec

$$t'_i = \begin{cases} t_i & \text{si } p_i = x_i \\ (\texttt{ref}\ \ t_i) & \text{si } p_i = \texttt{var}\ \ x_i \end{cases}$$

Programmes

(PROG) si $\Gamma_0 \vdash_{\text{Block}} bk : \texttt{void}$
alors $\vdash bk : \texttt{void}$

Blocs

(BLOC) si $\Gamma \vdash_{\text{Cmds}} (cs;\varepsilon) : t$
alors $\Gamma \vdash_{\text{Block}} [cs] : t$

Suite de commandes

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

(STATS0) si $s \in$ STAT, si $\Gamma \vdash_{\text{Stat}} s : \texttt{void}$, si $\Gamma \vdash_{\text{Cmds}} cs : t$
alors $\Gamma \vdash_{\text{Cmds}} (s;cs) : t$.

(STATS1) si $t \neq \texttt{void}$, si $s \in$ STAT, si $\Gamma \vdash_{\text{Stat}} s : t + \texttt{void}$, si $\Gamma \vdash_{\text{Cmds}} cs : t$
alors $\Gamma \vdash_{\text{Cmds}} (s;cs) : t$.

(STATS2) si $t \neq \texttt{void}$, si $s \in$ STAT, si $\Gamma \vdash_{\text{Stat}} s : t$
alors $\Gamma \vdash_{\text{Cmds}} (s;\varepsilon) : t$.

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

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

Définitions

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

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

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

(VAR) si $t \in \{\texttt{int}, \texttt{bool}\}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{VAR}\ x\ t) : \Gamma[x : (\texttt{ref}\ t)]$

(PROC) si $A([p_1 : t_1, \ldots, p_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$
si $\Gamma[x_1 : t'_1; \ldots; x_n : t'_n] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC}\ x\ \texttt{[}p_1\texttt{:}t_1, \ldots, p_n\texttt{:}t_n\texttt{]}bk) : \Gamma[x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}]$

(PROCREC)
si $A([p_1 : t_1, \ldots, x_n : t_n]) = [x_1 : t'_1, \ldots, x_n : t'_n]$
si $\Gamma[x_1 : t'_1; \ldots; x_n : t'_n; x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}] \vdash_{\text{Block}} bk : \texttt{void}$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{PROC REC}\ x\ \texttt{[}p_1\texttt{:}t_1, \ldots, p_n\texttt{:}t_n\texttt{]}bk) : \Gamma[x : t'_1\ \texttt{*}\ \ldots\ \texttt{*}\ t'_n\ \texttt{->}\ \texttt{void}]$

(FUNP) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n] \vdash_{\text{Block}} bk : t$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{FUN}\ x\ t\ \texttt{[}x_1\texttt{:}t_1\texttt{,} \ldots \texttt{,}x_n\texttt{:}t_n\texttt{]}\ bk) : \Gamma[x : \texttt{(}t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ t\texttt{)}]$

(FUNRECP) si $\Gamma[x_1 : t_1; \ldots; x_n : t_n; x : t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ t] \vdash_{\text{Block}} bk : t$
alors $\Gamma \vdash_{\text{Dec}} (\texttt{FUN REC}\ x\ t\ \texttt{[}x_1\texttt{:}t_1\texttt{,} \ldots \texttt{,}x_n\texttt{:}t_n\texttt{]}\ bk) : \Gamma[x : t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ t]$

Instructions

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

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

(IF0) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk_1 : t$ et si $\Gamma \vdash_{\text{Block}} bk_2 : t$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) : t$

(IF1) si $t \neq \texttt{void}$, si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk_1 : \texttt{void}$ et si $\Gamma \vdash_{\text{Block}} bk_2 : t$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) : t + \texttt{void}$

(IF2) si $t \neq \texttt{void}$, si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk_1 : t$ et si $\Gamma \vdash_{\text{Block}} bk_2 : \texttt{void}$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{IF}\ e\ bk_1\ bk_2) : t + \texttt{void}$

(WHILE) si $\Gamma \vdash_{\text{Expr}} e : \texttt{bool}$, si $\Gamma \vdash_{\text{Block}} bk : t$
alors $\Gamma \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) : t + \texttt{void}$

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

lvalue

(LVAR) si $\Gamma(x) = \texttt{(ref}\ t\texttt{)}$
alors $\Gamma \vdash_{\text{Lval}} x : t$

(LNTH) si $\Gamma \vdash_{\text{Expr}} e_1 : \texttt{(vec}\ t\texttt{)}$ et $\Gamma \vdash_{\text{Expr}} e_2 : \texttt{int}$
alors $\Gamma \vdash_{\text{Lval}} \texttt{(nth}\ e_1\ e_2\texttt{)} : t$

Expressions

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

(IDV) si $x \in \textsf{ident}$, si $\Gamma(x) = t$ avec $t \neq (\texttt{ref}\ t')$
alors $\Gamma \vdash_{\text{Expr}} x : t$

(IDR) si $x \in \textsf{ident}$,
si $\Gamma(x) = (\texttt{ref}\ t)$
alors $\Gamma \vdash_{\text{Expr}} x : t$

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

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

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

(ALLOC) si $\Gamma \vdash_{\text{Expr}} e : \texttt{int}$
alors $\Gamma \vdash_{\text{Expr}} (\texttt{alloc}\ e) : (\texttt{vec}\ t)$

(LEN) si $\Gamma \vdash_{\text{Expr}} e : \texttt{vec}\ t)$
alors $\Gamma \vdash_{\text{Expr}} (\texttt{len}\ e) : \texttt{int}$

(NTH) si $\Gamma \vdash_{\text{Expr}} e_1 : \texttt{vec}\ t)$ et si $\Gamma \vdash_{\text{Expr}} e_2 : \texttt{int}$
alors $\Gamma \vdash_{\text{Expr}} (\texttt{nth}\ e_1\ e_2) : t$

5.3 Sémantique

Programmes

$$\vdash p \rightsquigarrow (\sigma, \omega)$$

(PROG) si $\varepsilon, \varepsilon, \varepsilon \vdash_{\text{Block}} bk \rightsquigarrow (\varepsilon, \sigma, \omega)$
alors $\vdash bk \rightsquigarrow (\sigma, \omega)$

Blocs

$$\rho, \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (v, \sigma', \omega')$$

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

Suites de commandes

$$\rho, \sigma, \omega \vdash_{\text{Cmds}} cs \rightsquigarrow (v, \sigma', \omega')$$

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

(STATS0) 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 $\rho, \sigma, \omega \vdash_{\text{Cmds}} (s\texttt{;}\ cs) \rightsquigarrow (v, \sigma'', \omega'')$

(STATS1) si $\rho, \sigma, \omega \vdash_{\text{Stat}} s \rightsquigarrow (v, \sigma', \omega')$ avec $v \neq \varepsilon$ alors $\rho, \sigma, \omega \vdash_{\text{Cmds}} (s\texttt{;}\ cs) \rightsquigarrow (v, \sigma'', \omega'')$

(END0) $\rho, \sigma, \omega \vdash_{\text{Cmds}} \varepsilon \rightsquigarrow (\varepsilon, \sigma, \omega)$

(END1) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (v, \sigma', \omega')$ alors $\rho, \sigma, \omega \vdash_{\text{Cmds}} (\texttt{RETURN}\ e\texttt{;}\varepsilon) \rightsquigarrow (v, \sigma', \omega')$

Définitions

$$\rho, \sigma, \omega \vdash_{\text{Dec}} d \rightsquigarrow (\rho', \sigma', \omega')$$

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

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

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

(VAR) si $alloc(\sigma) = (a, \sigma')$, avec $\sigma' = \sigma[a = any]$ et $a \notin \mathsf{dom}(\sigma)$
alors $\rho, \sigma, \omega \vdash_{\text{Dec}} (\texttt{VAR}\ x\ t) \rightsquigarrow (\rho[x = inA(a)], \sigma', \omega)$

(PROC) $\rho, \sigma, \omega \vdash_{\text{Dec}} (\texttt{PROC}\ x\ t\ \texttt{[}p_1\texttt{:}t_1\texttt{,}\ \ldots p_n\texttt{:}t_n\texttt{]}\ bk)$
$\rightsquigarrow (\rho[x = inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma, \omega)$

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

(FUNP) $\rho, \sigma, \omega \vdash_{\text{Dec}} (\texttt{FUN}\ x\ t\ \texttt{[}p_1\texttt{:}t_1\texttt{,}\ \ldots p_n\texttt{:}t_n\texttt{]}\ bk)$
$\rightsquigarrow (\rho[x = inP(bk, \lambda v_1 \ldots v_n.\rho[x_1 = v_1; \ldots; x_n = v_n]), \sigma, \omega)$

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

Instructions

$$\rho, \sigma, \omega \vdash_{\text{Stat}} s \rightsquigarrow (v, \sigma', \omega')$$

(SET) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e_2 \rightsquigarrow (v, \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Lval}} e_1 \rightsquigarrow (a, \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{SET}\ e_1\ e_2) \rightsquigarrow (\varepsilon, \sigma''[a := v], \omega'')$

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

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

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

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

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

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

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

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

lvalue

$$\rho,\sigma,\omega \vdash_{\text{LVAL}} e \rightsquigarrow (a,\sigma',\omega')$$

(LID0) si $x \in \mathit{ident}$, si $\rho(x) = inA(a)$ alors $\rho,\sigma,\omega \vdash_{\text{LVAL}} x \rightsquigarrow (a,\sigma,\omega)$

(LID1) si $x \in \mathit{ident}$, si $\rho(x) = inB(a)$ alors $\rho,\sigma,\omega \vdash_{\text{LVAL}} x \rightsquigarrow (a+1,\sigma,\omega)$

(LNTH1) si $\rho,\sigma,\omega \vdash_{\text{LVAL}} e_1 \rightsquigarrow (a,\sigma',\omega')$, si $\rho,\sigma',\omega' \vdash_{\text{EXPR}} e_2 \rightsquigarrow (inZ(i),\sigma'',\omega'')$ et si $\sigma''(a+i) = InZ(n)$
alors $\rho,\sigma \vdash_{\text{LVAL}}$ `(nth` e_1 e_2`)` $\rightsquigarrow (a+i,\sigma'',\omega'')$

(LNTH2) si $\rho,\sigma,\omega \vdash_{\text{LVAL}} e_1 \rightsquigarrow (a,\sigma',\omega')$, si $\rho,\sigma',\omega' \vdash_{\text{EXPR}} e_2 \rightsquigarrow (inZ(i),\sigma'',\omega'')$ et si $\sigma''(a+i) = InB(a')$
alors $\rho,\sigma,\omega \vdash_{\text{LVAL}}$ `(nth` e_1 e_2`)` $\rightsquigarrow (a'+1,\sigma'',\omega'')$

Paramètres d'appel

$$\rho,\sigma,\omega \vdash_{\text{EXPAR}} p \rightsquigarrow (u,\sigma',\omega')$$

(REF) si $\rho(x) = inA(a)$ alors $\rho,\sigma,\omega \vdash_{\text{EXPAR}}$ `(adr` x`)` $\rightsquigarrow (inA(a),\sigma,\omega)$

(VAL) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e \rightsquigarrow (v,\sigma',\omega')$ alors $\rho,\sigma,\omega \vdash_{\text{EXPAR}} e \rightsquigarrow (v,\sigma',\omega')$

Expressions

$$\rho,\sigma,\omega \vdash_{\text{EXPR}} e \rightsquigarrow (v,\sigma',\omega')$$

(TRUE) $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `true` $\rightsquigarrow (inZ(1),\sigma,\omega)$

(FALSE) $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `false` $\rightsquigarrow (inZ(0),\sigma,\omega)$

(NUM) si $n \in$ `num` alors $\rho,\sigma,\omega \vdash_{\text{EXPR}} n \rightsquigarrow (inZ(\nu(n)),\sigma,\omega)$

(ID1) si $x \in \mathsf{ident}$ et $\rho(x) = inA(a)$ alors $\rho,\sigma,\omega \vdash_{\text{EXPR}} x \rightsquigarrow (inZ(\sigma(a)),\sigma,\omega)$

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

(PRIM1) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e \rightsquigarrow (inZ(n),\sigma',\omega')$, et si $\pi_1(not)(n) = n'$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(not` e`)` $\rightsquigarrow (inZ(n'),\sigma',\omega')$

(PRIM2) si $x \in \{$`eq lt add sub mul div`$\}$,
si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e_1 \rightsquigarrow (inZ(n_1),\sigma',\omega')$, si $\rho,\sigma',\omega' \vdash_{\text{EXPR}} e_2 \rightsquigarrow (inZ(n_2),\sigma'',\omega'')$ et si $\pi_2(x)(n_1,n_2) = n$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(`x e_1e_2`)` $\rightsquigarrow (inZ(n),\sigma'',\omega'')$

(AND0) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e_1 \rightsquigarrow (inZ(0),\sigma',\omega')$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(and` e_1 e_2`)` $\rightsquigarrow (inZ(0),\sigma',\omega')$.

(AND1) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e_1 \rightsquigarrow (inZ(1),\sigma',\omega')$ et si $\rho,\sigma',\omega' \vdash_{\text{EXPR}} e_2 \rightsquigarrow (v,\sigma'',\omega'')$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(and` e_1 e_2`)` $\rightsquigarrow (v,\sigma'',\omega'')$.

(OR1) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e_1 \rightsquigarrow (inZ(1),\sigma',\omega')$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(or` e_1 e_2`)` $\rightsquigarrow (inZ(1),\sigma',\omega')$.

(OR0) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e_1 \rightsquigarrow (inZ(0)\sigma',\omega')$ et si $\rho,\sigma',\omega' \vdash_{\text{EXPR}} e_2 \rightsquigarrow (v,\sigma'',\omega'')$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(or` e_1 e_2`)` $\rightsquigarrow (v,\sigma'',\omega'')$.

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

(IF0) si $\rho,\sigma,\omega \vdash_{\text{EXPR}} e_1 \rightsquigarrow (inZ(0),\sigma',\omega')$ et si $\rho,\sigma',\omega' \vdash_{\text{EXPR}} e_3 \rightsquigarrow (v,\sigma'',\omega'')$
alors $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `(if` e_1 e_2 e_3`)` $\rightsquigarrow (v,\sigma'',\omega'')$

(ABS) $\rho,\sigma,\omega \vdash_{\text{EXPR}}$ `[`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 $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inF(e', r), \sigma_0, \omega_0)$,
si $\rho, \sigma_0, \omega_0 \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, ..., si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$,
si $\rho' = r(v_1, \ldots, v_n)$ et si $\rho', \sigma_n, \omega_n \vdash_{\text{Expr}} e' \rightsquigarrow (v, \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash \texttt{(}e\ e_1 \ldots e_n\texttt{)} \rightsquigarrow (v, \sigma', \omega')$

(APPR) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow (inFR(\varphi), \sigma_0, \omega_0)$, si $\varphi(inFR(\varphi)) = inF(e', r)$,
si $\rho, \sigma_0, \omega_0 \vdash_{\text{Expr}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, ..., si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expr}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$,
si $\rho' = r(v_1, \ldots, v_n)$ et si $\rho', \sigma_n, \omega_n \vdash_{\text{Expr}} e' \rightsquigarrow (v, \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash_{\text{Expr}} (e\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma', \omega')$

(ALLOC) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inZ(n), \sigma', \omega')$, avec $n > 0$,
et si $allocb(\sigma', n) = (a, \sigma'')$, avec $\sigma'' = \sigma'[a = inZ(n)]$,
alors $\rho, \sigma, \omega \vdash_{\text{Expr}} \texttt{(alloc } e\texttt{)} \rightsquigarrow (inB(a), \sigma'', \omega')$

(LEN) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e \rightsquigarrow (inB(a), \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash_{\text{Expr}} \texttt{(len } e\texttt{)} \rightsquigarrow (\sigma'(a), \sigma', \omega')$

(NTH) si $\rho, \sigma, \omega \vdash_{\text{Expr}} e_1 \rightsquigarrow (inB(a), \sigma', \omega')$ et si $\rho, \sigma', \omega' \vdash_{\text{Expr}} e_2 \rightsquigarrow (inZ(i), \sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Expr}} \texttt{(nth } e_1\ e_2\texttt{)} \rightsquigarrow (\sigma''(a + i + 1), \sigma'', \omega'')$

(AFP) si $x \in \textsf{ident}$ et $\rho(x) = inP(bk, r)$,
si $\rho, \sigma, \omega \vdash_{\text{Expar}} e_1 \rightsquigarrow (v_1, \sigma_1, \omega_1)$, ..., si $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{Expar}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
si $\rho' = r(v_1, \ldots, v_n)$ et $\rho', \sigma_n, \omega_n \vdash_{\text{Block}} bk \rightsquigarrow (v, \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (x\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma', \omega')$

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