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

P. Manoury *

Janvier 2022

2 APS1 : noyau impératif

2.1 Syntaxe

Lexique

Symboles réservés

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

Mots clef

```
CONST FUN REC VAR PROC
ECHO SET IF WHILE CALL
if and or
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 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

*Avec la précieuse relecture de W.S. et V.M. Qu'ils en soient remerciés.

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 `)`
	|	`( and` EXPR EXPR `)`
	|	`( or` 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

(BLOCK) 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 $\Gamma \vdash_{\text{DEF}} d : \Gamma'$, 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 $\Gamma \vdash_{\text{Def}}$ (`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 $\Gamma \vdash_{\text{Def}}$ (`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 $\Gamma \vdash_{\text{Def}}$ (`FUN REC` $x\ t$ `[`x_1`:`t_1`,` ... `,`x_n`:`t_n`]` e) : $\Gamma[x : t_1$ `*` ... `*` t_n `->` $t]$

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

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

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

Intructions

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

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

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

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

2.3 Sémantique

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

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

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

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

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

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

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

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

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

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

(FUNREC) $\rho \vdash_{\text{Def}} (\texttt{FUN REC}\ x\ t\ \texttt{[}x_1\texttt{:}t_1\texttt{,}\ \ldots\texttt{,}x_n\texttt{:}t_n\texttt{]}\ e) \rightsquigarrow \rho[x = inFR(e, x, (x_1; \ldots; x_n), \rho)]$

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

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

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

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 $\rho, \sigma', \omega' \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma'', \omega'')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{WHILE}\ e\ bk) \rightsquigarrow (\sigma'', \omega'')$

(CALL) si $\rho(x) = inP(bk, (x_1; \ldots; x_n), \rho')$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$
si $\rho'[x_1 = v_1; \ldots; x_n = v_n), \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
alors $\rho, \sigma, \omega \vdash_{\text{Stat}} (\texttt{CALL}\ x\ e_1 \ldots e_n) \rightsquigarrow (\sigma', \omega')$

(CALLR) si $\rho(x) = inPR(bk, x, (x_1; \ldots; \rho')$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$
et si $\rho'[x_1 = v_1; \ldots; x_n = v_n][x = inPR(bk, x, (x_1; \ldots; x_n), \rho')], \sigma, \omega \vdash_{\text{Block}} bk \rightsquigarrow (\sigma', \omega')$
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 $\pi_1(\texttt{not})(n) = n'$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(not } e\texttt{)} \rightsquigarrow inZ(n')$

(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 $\rho, \sigma \vdash_{\text{Expr}} \texttt{(}x\ e_1 e_2\texttt{)} \rightsquigarrow inZ(n)$

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

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

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

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

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

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

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

(APP) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inF(e', (x_1; \ldots; x_n), \rho')$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$,
si $\rho'[x_1 = v_1; \ldots; x_n = v_n], \sigma \vdash_{\text{Expr}} e' \rightsquigarrow v$
alors $\rho, \sigma \vdash \texttt{(}e\ e_1 \ldots e_n\texttt{)} \rightsquigarrow v$

(APPR) si $\rho, \sigma \vdash_{\text{Expr}} e \rightsquigarrow inFR(e', x, (x_1; \ldots; x_n), \rho')$,
si $\rho, \sigma \vdash_{\text{Expr}} e_1 \rightsquigarrow v_1$, ..., si $\rho, \sigma \vdash_{\text{Expr}} e_n \rightsquigarrow v_n$,
si $\rho'[x_n = v_1, \ldots, x_n = v_n, x = inFR(e', x, (x_1; \ldots; x_n), \rho')], \sigma \vdash_{\text{Expr}} e' \rightsquigarrow v$
alors $\rho, \sigma \vdash_{\text{Expr}} \texttt{(}e\ e_1 \ldots e_n\texttt{)} \rightsquigarrow v$