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

P. Manoury*

Janvier 2022

1 APS0 : noyau fonctionnel

1.1 Syntaxe

Lexique

Symboles réservés

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

Mots clef

`CONST FUN REC ECHO 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 primitifs `true false not 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

Type

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

Types ::= Type
  | Type `*` Types

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

Paramètres formels

$$\begin{array}{lcl} \textsc{Args} & ::= & \textsc{Arg} \\ & \mid & \textsc{Arg}\ \texttt{,}\ \textsc{Args} \\ \textsc{Arg} & ::= & \mathsf{ident}\ \texttt{:}\ \textsc{Type} \end{array}$$

Instruction

$$\begin{array}{lcl} \textsc{Stat} & ::= & \texttt{ECHO}\ \textsc{Expr} \end{array}$$

Expression

$$\begin{array}{lcl} \textsc{Expr} & ::= & \mathsf{num} \\ & \mid & \mathsf{ident} \\ & \mid & \texttt{(if}\ \textsc{Expr}\ \textsc{Expr}\ \textsc{Expr}\ \texttt{)} \\ & \mid & \texttt{(and}\ \textsc{Expr}\ \textsc{Expr}\ \texttt{)} \\ & \mid & \texttt{(or}\ \textsc{Expr}\ \textsc{Expr}\ \texttt{)} \\ & \mid & \texttt{(}\ \textsc{Expr}\ \textsc{Exprs}\ \texttt{)} \\ & \mid & \texttt{[}\ \textsc{Args}\ \texttt{]}\ \textsc{Expr} \end{array}$$

Suite d'expressions

$$\begin{array}{lcl} \textsc{Exprs} & ::= & \textsc{Expr} \\ & \mid & \textsc{Expr}\ \textsc{Exprs} \end{array}$$

1.2 Typage

Contexte initial

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

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

(PROG) si $\Gamma_0 \vdash_{\textsc{Cmds}} cs : \texttt{void}$
alors $\vdash \texttt{[}cs\texttt{]} : \texttt{void}$

Suite de commandes $\Gamma \vdash_{\textsc{Cmds}} cs : \texttt{void}$

(DEFS) si $d \in \textsc{Def}$, si $\Gamma \vdash_{\textsc{Def}} d : \Gamma'$, si $\Gamma' \vdash_{\textsc{Cmds}} cs : \texttt{void}$
alors $\Gamma \vdash_{\textsc{Cmds}} (d\texttt{;}cs) : \texttt{void}$.

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

Définitions $\Gamma \vdash_{\textsc{Def}} d : \Gamma'$

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

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

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

Intruction $\Gamma \vdash_{\text{Stat}} s : \texttt{void}$

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

Expressions $\Gamma \vdash_{\text{Expr}} e : t$

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

(ID) si $x \in \mathsf{ident}$, si $\Gamma(x) = t$
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 $\Gamma \vdash_{\text{Expr}} \texttt{(if}\ e_1\ e_2\ e_3\texttt{)} : t$

(AND) si $\Gamma \vdash_{\text{Expr}} e_1 : \texttt{bool}$, si $\Gamma \vdash_{\text{Expr}} e_2 : \texttt{bool}$
alors $\Gamma \vdash_{\text{Expr}} \texttt{(and}\ e_1\ e_2\texttt{)} : \texttt{bool}$

(OR) si $\Gamma \vdash_{\text{Expr}} e_1 : \texttt{bool}$, si $\Gamma \vdash_{\text{Expr}} e_2 : \texttt{bool}$
alors $\Gamma \vdash_{\text{Expr}} \texttt{(or}\ e_1\ e_2\texttt{)} : \texttt{bool}$

(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 $\Gamma \vdash_{\text{Expr}} (e\ e_1 \ldots e_n) : t$

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

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 $\varepsilon, \varepsilon \vdash_{\text{Cmds}} cs \rightsquigarrow \omega$
alors $\vdash [cs] \rightsquigarrow \omega$

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

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

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

Définitions $\rho \vdash_{\text{Def}} d \rightsquigarrow \rho'$

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

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

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

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

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

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

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

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

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

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

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

(PRIM2) si $x \in \{$`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 $\rho \vdash_{\text{Expr}}$ ($x\ e_1\ e_2$) $\rightsquigarrow inZ(n)$

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

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

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

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

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

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

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

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

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