UPMC

Master informatique 2 – STL

NI503 – Conception de langages

Notes III

2012

1 Fonctions

En exercice

On veut définir des fonctions f et les utiliser : $(fx) + 1$.

Syntaxe

Dec	::=	DecP | DecF
DecP	::=	...
DecF	::=	`FUN` ident Idents = Expr
	|	`FUNREC` ident Idents = Expr
...		
Expr	::=	...
	|	ident Exprs

Sémantique Ensemble des fonctions

- $FFun_1 = \mathbb{R} \to \mathbb{R}^3 \to \mathbb{R}$
- $FFun_{n+1} = \mathbb{R} \to FFun_n$
- $FFun = \bigcup_n FFun_n$

$$EnvFun = Ident \to FFun$$

Signature des fonctions sémantiques :

P	:	$Prog \to (\mathbb{R}^3)^\star$
Cs	:	$Cmds \to (\mathbb{R}^3)^\star \to Env \to EnvProc \to EnvFun \to (\mathbb{R}^3)^\star$
C	:	$Cmd \to (\mathbb{R}^3)^\star \to Env \to EnvProc \to EnvFun \to (\mathbb{R}^3)^*$
Dp	:	$Dec \to Env \to EnvProc \to EnvProc$
Df	:	$Dec \to Env \to EnvFun \to EnvFun$
E	:	$Expr \to Env \to EnvFun \to \mathbb{R}$

$c \in Cmd$, $dp \in DecP$, $df \in DecF$

$$
\begin{array}{lcl}
\mathbf{P}[[\texttt{[}cs\texttt{]}]] & = & \mathbf{Cs}[[cs]]\{\frac{pi}{2}, 0, 0\}\ \emptyset\ \emptyset\ \emptyset \\
\mathbf{Cs}[[c\ ;\ cs]]es\ \rho_v\ \rho_p\ \rho_f & = & \mathbf{Cs}[[cs]]es'\ \rho_v\ \rho_p\ \rho_f \\
 & & \text{avec } es' = \mathbf{C}[[c]]es\ \rho_v\ \rho_p\ \rho_f \\
\mathbf{Cs}[[dp\ ;\ cs]]es\ \rho_v\ \rho_p\ \rho_f & = & \mathbf{Cs}[[cs]]es\ \rho_v\ \rho'_p\ \rho_f \\
 & & \text{avec } \rho'_p = \mathbf{Dp}[[dp]]\rho_v\ \rho_p\ \rho_f \\
\mathbf{Dp}[[\texttt{PROC}\ f\ x \ldots\ \texttt{=}\ p]]\rho_v\ \rho_p\ \rho_f & = & \rho_p[f \mapsto \lambda v \ldots \lambda es.(\mathbf{P}[[p]]es\ \rho'_v\ \rho_p\ \rho_f)] \\
 & & \text{avec } \rho'_v = \rho_v[x \mapsto v; \ldots] \\
\mathbf{Dp}[[\texttt{PROCREC}\ f\ x \ldots\ \texttt{=}\ p]]\rho_v\ \rho_p\ \rho_f & = & \rho_p[f \mapsto !w.\lambda v \ldots \lambda es.(\mathbf{P}[[p]]es\ \rho'_v\ \rho'_p\ \rho_f)] \\
 & & \text{avec } \rho'_v = \rho_v[x \mapsto v; \ldots] \text{ et } \rho'_p = \rho_p[f \mapsto w] \\
\mathbf{Cs}[[df\ ;\ cs]]es\ \rho_v\ \rho_p\ \rho_f & = & \mathbf{Cs}[[cs]]es\ \rho_v\ \rho_p\ \rho'_f \\
 & & \text{avec } \rho'_f = \mathbf{Df}[[dp]]\rho_v\ \rho_p\ \rho_f \\
\mathbf{Dp}[[\texttt{FUN}\ f\ x \ldots\ \texttt{=}\ e]]\rho_v\ \rho_p\ \rho_f & = & \rho_f[f \mapsto \lambda v \ldots (\mathbf{E}[[e]]\rho'_v\ \rho_f)] \\
 & & \text{avec } \rho'_v = \rho_v[x \mapsto v; \ldots] \\
\mathbf{Dp}[[\texttt{FUNREC}\ f\ x \ldots\ \texttt{=}\ e]]\rho_v\ \rho_p\ \rho_f & = & \rho_f[f \mapsto !w.\lambda v \ldots (\mathbf{E}[[e]]\rho'_v\ \rho'_f)] \\
 & & \text{avec } \rho'_v = \rho_v[x \mapsto v; \ldots] \text{ et } \rho'_f = \rho_f[f \mapsto w] \\
 & & \\
\mathbf{E}[[f\ e \ldots]]\rho_v\ \rho_f & = & (\rho_f\ f)\ (\mathbf{E}[[e]]\rho_v\ \rho_f) \ldots
\end{array}
$$

2 Synthèse

Un environnement polymorphe : les valeurs réelles, les fonctions et les procédures.

$$Val = I\!R \oplus FProc \oplus FFun$$

$$Env = Ident \to Val$$

Fonctions d'injection :
- $inR : I\!R \to Val$
- $inP : FProc \to Val$
- $inF : FFun \to Val$

Projections :
- $outR : Val \to I\!R$
- $outP : Val \to FProc$
- $outF : Val \to FFun$

Rappel unions disjointes $A \oplus B = \{(0, x) | x \in A\} \cup \{(1, x) | x \in B\}$

Définitions :
- $inR(x) = (0, x)$ etc.
- $outR(x) = (\pi_2\ x)$ etc.
- $isR(x) = ((pi_1\ x) = 0)$ etc.

Pattern matching (macro)

```
case t :
  inR(x) → (f1 x)   /* if isR(x) (f1(outR(x))) */
| inP(x) → (f2 x)   /* if isP(x) (f2(outP(x))) */
| inF(x) → (f3 x)   /* if isF(x) (f3(outF(x))) */
| _ → ⊥
```

Sémantique revisitée

Signatures

$$
\begin{array}{lcl}
\mathbf{P} & : & Prog \to (I\!R^3)^\star \\
\mathbf{Cs} & : & Cmds \to (I\!R^3)^\star \to Env \to (I\!R^3)^\star \\
\mathbf{C} & : & Cmd \to (I\!R^3)^\star \to Env \to (I\!R^3)^* \\
\mathbf{D} & : & Dec \to Env \to Env \\
\mathbf{E} & : & Expr \to Env \to I\!R
\end{array}
$$

Équations

$$
\begin{array}{lcl}
\mathbf{P}[[\texttt{[}cs\texttt{]}]] & = & \mathbf{Cs}[[cs]]\{\frac{pi}{2}, 0, 0\}\ \emptyset \\
\mathbf{Cs}[[d\texttt{;}cs]]es\ \rho & = & \mathbf{Cs}[[cs]]es\ \mathbf{D}[[d]]\rho \\
\mathbf{Cs}[[c\texttt{;}cs]]es\ \rho & = & \mathbf{Cs}[[cs]](\mathbf{C}[[c]]es\ \rho)\ \rho \\
\mathbf{D}[[\texttt{PROC}\ f\ x\ \texttt{=}\ p]]\rho & = & \rho[f \mapsto inP(\lambda v \lambda es.(\mathbf{P}[[p]]es(\rho[x \mapsto v])))] \\
\mathbf{D}[[\texttt{PROCREC}\ f\ x\ \texttt{=}\ p]]\rho & = & \rho[f \mapsto inP(!w.\lambda v \lambda es.(\mathbf{P}[[p]]es(\rho[x \mapsto v; f \mapsto inP(w)])))] \\
\mathbf{D}[[\texttt{FUN}\ f\ x\ \texttt{=}\ e]]\rho & = & \rho[f \mapsto inF(\lambda v \lambda es.(\mathbf{E}[[e]](\rho[x \mapsto v])))] \\
\mathbf{D}[[\texttt{FUNREC}\ f\ x\ \texttt{=}\ e]]\rho & = & \rho[f \mapsto inF(!w.\lambda v \lambda es.(\mathbf{E}[[e]](\rho[x \mapsto v; f \mapsto inF(w)])))] \\
\mathbf{C}[[\texttt{MOVE}\ e]](a,x,y) :: es\ \rho & = & (a, x + k\cos(a), y + k\sin(a)) :: (a,x,y) :: es \\
 & & \text{avec } k = \mathbf{E}[[e]]\rho \\
\mathbf{C}[[\texttt{TURN}\ e]](a,x,y) :: es\ \rho & = & (a + d\frac{\pi}{180}, x, y) :: (a,x,y) :: es \\
 & & \text{avec } d = \mathbf{E}[[e]]\rho \\
\mathbf{C}[[\texttt{CALL}\ f\ e]]es\ \rho & = & \mathsf{case}\ (\rho\ f) : \\
 & & \quad inP(p) \to (p\ (\mathbf{E}[[e]]\rho)\ es) \\
 & & \mid \_ \to \bot \\
\vdots & & \\
\mathbf{E}[[x]]\rho & = & \mathsf{case}\ (\rho\ x) : \\
 & & \quad inR(v) \to v \\
 & & \mid \_ \to \bot \\
\mathbf{E}[[e_1\texttt{+}e_2]]\rho & = & (\mathbf{E}[[e_1]]\rho) + (\mathbf{E}[[e_2]]\rho) \\
\vdots & & \\
\mathbf{E}[[(f\ e)]]\rho & = & \mathsf{case}\ (\rho\ f) : \\
 & & \quad inF(t) \to (t\ (\mathbf{E}[[e]]\rho)) \\
 & & \mid \_ \to \bot
\end{array}
$$

Question : fonctionnelles ?

3 Affectation

Déclarer des variables, modifier leur valeur.

Syntaxe

$$
\begin{array}{lcl}
\textsc{Dec} & ::= & \ldots \mid \textsc{DecV} \\
\textsc{DecV} & ::= & \texttt{VAR}\ \mathsf{ident} \\
 & & \\
\textsc{Cmd} & ::= & \ldots \\
 & \mid & \mathsf{ident}\ \texttt{:=}\ \textsc{Expr}
\end{array}
$$

Exemple :

```
[
  VAR x;
  PROC f = [ MOVE x; TURN 30; x := x+1; CALL f ]
]
```

Sémantiques

Une «mémoire» dynamique : un domaine *abstrait* d'adresses ; $Mem = Adr \to \mathbb{R}_\perp$

Opérations mémoire

- Allocation $newM : Mem \to Adr \times Mem$
- Modification $setM : Mem \to Adr \to \mathbb{R} \to Mem_\perp$
- Accès $getM : Mem \to Adr \to \mathbb{R}_\perp$

Axiomatisation des opérations mémoire : soit $\mu \in Mem$

- si $a, \mu' = newM(\mu)$ alors $(getM\ \mu\ a) = \perp$ et $(getM\ \mu'\ a) = 0$.
- si $\mu' = (setM\ \mu\ a\ v)$ alors $(getM\ \mu'\ a) = v$.

Environnement contient aussi des adresses :

$$Val = \mathbb{R} \oplus FProc \oplus FFun \oplus Adr$$

avec inA, $outA$ et isA.

Signatures des fonctions sémantiques :

P	:	$Prog \to Mem$
Cs	:	$Cmds \to Env \to Mem \to Mem$
C	:	$Cmd \to Env \to Mem \to Mem$
D	:	$Dec \to Env \to Mem \to Env \times Mem$
E	:	$Expr \to Env \to Mem \to \mathbb{R}$

Mémoriser l'état On ne conserve plus la trace et on alloue trois valeurs pour mémoriser direction et coordonnées : soient

- $\mu_0 = \emptyset$;
- $\mu_1 = (setM\ \mu'_1\ a_1\ \frac{\pi}{2})$ avec $a_1, \mu'_1 = (newM\ \mu_0)$;
- $\mu_2 = (setM\ \mu'_2\ a_2\ 0)$ avec $a_2, \mu'_2 = (newM\ \mu_1)$;
- $\mu_3 = (setM\ \mu'_3\ a_3\ 0)$ avec $a_3, \mu'_3 = (newM\ \mu_2)$;

Notons $M_0 = \mu_3$

Équations

$$
\begin{array}{lcl}
\mathbf{P}[[\texttt{[}cs\texttt{]}]] & = & \mathbf{Cs}[[cs]]\emptyset\ M_0 \\
\mathbf{Cs}[[d\texttt{;}cs]]\rho\ \mu & = & \mathbf{Cs}[[cs]]\rho'\ \mu' \\
 & & \text{avec } \rho', \mu' = \mathbf{D}[[d]]\rho\ \mu \\
\mathbf{Cs}[[c\texttt{;}cs]]\rho\ \mu & = & \mathbf{Cs}[[cs]]\rho\ \mu' \\
 & & \text{avec } \mu' = \mathbf{C}[[c]]\rho\ \mu
\end{array}
$$

$$
\begin{array}{lcl}
\mathbf{D}[[\texttt{PROC } f\ x\ \texttt{=}\ p]]\rho\ \mu & = & \rho', \mu \\
 & & \text{avec } \rho' = \rho[f \mapsto inP(\lambda v \lambda \mu.(\mathbf{P}[[p]](\rho[x \mapsto v])\ \mu))] \\
\mathbf{D}[[\texttt{PROCREC } f\ x\ \texttt{=}\ p]]\rho\ \mu & = & \rho', \mu \\
 & & \text{avec } \rho' = \rho[f \mapsto inP(!w.\lambda v \lambda \mu.(\mathbf{P}[[p]]es(\rho[x \mapsto v; f \mapsto inP(w)])\mu))] \\
\mathbf{D}[[\texttt{FUN } f\ x\ \texttt{=}\ e]]\rho\ \mu & = & \rho', \mu \\
 & & \text{avec } \rho' = \rho[f \mapsto inF(\lambda v \lambda \mu.(\mathbf{E}[[e]](\rho[x \mapsto v])\ \mu))] \\
\mathbf{D}[[\texttt{FUNREC } f\ x\ \texttt{=}\ e]]\rho\ \mu & = & \rho', \mu \\
 & & \text{avec } \rho' = \rho[f \mapsto inF(!w.\lambda v \lambda \mu.(\mathbf{E}[[e]](\rho[x \mapsto v; f \mapsto inF(w)])\ \mu))] \\
 & & \\
\mathbf{D}[[\texttt{VAR } x]]\rho\ \mu & = & \rho', \mu' \\
 & & \text{avec } \rho' = \rho[x \mapsto a] \text{ et } a, \mu' = (newM\ \mu) \\
\mathbf{C}[[\texttt{MOVE } e]]\rho\ \mu & = & (setM\ (setM\ \mu\ a_2\ x + k\cos(\alpha))\ a_3\ y + k\sin(\alpha)) \\
 & & \text{avec } \alpha = (getM\ \mu\ a_1), x = (getM\ \mu\ a_2), y = (getM\ \mu\ a_3) \\
 & & \text{et } k = \mathbf{E}[[e]]\rho\ \mu \\
\mathbf{C}[[\texttt{TURN } e]]\rho\ \mu & = & (setM\ \mu\ a_1\ (\alpha + d\frac{\pi}{180})) \\
 & & \text{avec } \alpha = (getM\ \mu\ a_1) \text{ et } d = \mathbf{E}[[e]]\rho\ \mu \\
\mathbf{C}[[\texttt{CALL } f\ e]]\rho\ \mu & = & \mathsf{case}\ (\rho\ f): \\
 & & \quad inP(p) \to (p\ (\mathbf{E}[[e]]\rho\ \mu)\ \mu) \\
 & & \mid \_ \to \bot \\
 & & \\
\mathbf{C}[[x\ \texttt{:=}\ e]]\rho\ \mu & = & \text{case}\ (\rho\ x): \\
 & & \quad inA(a) \to (setM\ \mu\ a\ (\mathbf{E}[[e]]\rho\ \mu)) \\
 & & \mid \_ \to \bot \\
 & & \\
\vdots & & \\
\mathbf{E}[[x]]\rho\ \mu & = & \mathsf{case}\ (\rho\ x): \\
 & & \quad inR(v) \to v \\
 & & \mid inA(a) \to (getM\ \mu\ a) \\
 & & \mid \_ \to \bot \\
\mathbf{E}[[e_1\texttt{+}e_2]]\rho\ \mu & = & (\mathbf{E}[[e_1]]\rho\ \mu) + (\mathbf{E}[[e_2]]\rho\ \mu) \\
\vdots & & \\
\mathbf{E}[[(f\ e)]]\rho\ \mu & = & \mathsf{case}\ (\rho\ f): \\
 & & \quad inF(t) \to (t\ (\mathbf{E}[[e]]\rho\ \mu)\ \mu) \\
 & & \mid \_ \to \bot
\end{array}
$$

Exercice : passage par valeur, passage par référence.

4 Boucles

Boucle *while*

Syntaxe

CMD ::= ...
| WHILE EXP PROG

Sémantique la commanda WHILE est une fonction récursive de la mémoire.

$$
\begin{array}{lcl}
\mathbf{C}[[\texttt{WHILE } e\ \texttt{[}cs\texttt{]}]]\rho\ \mu & = & (!w.\lambda\mu'.(\mathsf{if}\ (\mathbf{E}[[e]]\rho\ \mu') \\
 & & \qquad\qquad (w\ (\mathbf{Cs}[[cs]]\rho\ \mu')) \\
 & & \qquad\qquad \mu') \\
 & & \quad \mu)
\end{array}
$$

Itération bornée simple

Syntaxe

CMD ::= ...
| `REPEAT` EXP PROG

`REPEAT` n cs itère n fois la séquence cs.

Sémantique La commande `REPEAT` est une fonction récursive à paramètre entier (indice de boucle) :

$$\begin{aligned}\mathbf{C}[[\texttt{REPEAT } e\ cs]]\rho\ \mu \quad = \quad & (!w.\lambda i.\lambda\mu.(\mathsf{if}\ (i>0)\\ & \qquad (w\ (i-1)\ (\mathbf{Cs}[[cs]]\rho\ \mu))\\ & \qquad \mu)\\ & (\mathbf{E}[[e]]\ \rho\ \mu)\\ & \mu)\end{aligned}$$

Boucle *for* (la vraie)

Une itération bormée avec un indice explicite de boucle.

Syntaxe

CMD ::= ...
| `FOR` ident EXP EXP PROG

Sémantique

$$\begin{aligned}\mathbf{C}[[\texttt{FOR } i\ e_1\ e_2\ \texttt{[}cs\texttt{]}]]\rho\ \mu \quad = \quad & (!w.\lambda\rho'\lambda\mu'.(\mathsf{if}\ (outR(\rho'\ i)<(\mathbf{E}[[e_2]]\rho\ \mu))\\ & \qquad (w\ (\rho'[i\mapsto outR(\rho'\ i)+1])\ (\mathbf{Cs}[[cs]]\rho'\ \mu'))\\ & \qquad \mu')\\ & \rho[i\mapsto inR(\mathbf{E}[[e_1]]\rho\ \mu)]\\ & \mu)\end{aligned}$$

Question : *quid* de `FOR i 0 10 [i := 0]` ?