/SU/FSI/MASTER/INFO/MU4IN503 (APS)

Analyse des Programmes et Sémantique

Janvier 2021

Pascal Manoury – Romain Demangeon
pascal.manoury@lip6.fr

7 : $APS3$: fonctions procédurales

(ou procédures fonctionnelles)

fonctionnel *vs* impératif

APS0 les expressions et les fonctions produisent des *valeurs*

APS1 les instructions et les procédures produisent des *effets*

$$APS3$$

1. Utiliser des instructions pour produire de valeurs.
2. Utiliser des blocs de commandes pour définir des fonctions

Un unique ajout : `RETURN`

- Un impact minime sur la syntaxe
- Un impact profond sur le typage et la sémantique

Syntaxe

Lexique : nouveau mot clef `RETURN`

Grammaire :

La *commande* `RETURN`

$$\begin{array}{lll} \textsc{Cmds} & ::= & \ldots \\ & | & \texttt{RETURN}\ \textsc{Expr} \end{array}$$

Les définitions *fonctions procédurales*

$$\begin{array}{lll} \textsc{Def} & ::= & \ldots \\ & | & \texttt{FUN}\ \text{ident}\ \textsc{Type}\ [\ \textsc{Argsp}\ ]\ \textsc{Bloc} \\ & | & \texttt{FUN}\ \texttt{REC}\ \text{ident}\ \textsc{Type}\ [\ \textsc{Argsp}\ ]\ \textsc{Bloc} \end{array}$$

L'application des fonctions procédurales est identique celle des fonctions «fonctionnelles»

La place de `RETURN`

- Syntaxe : `RETURN` en fin d'une suite de commandes
- Sémantique : cohérent
  `RETURN 1; ECHO 42` n'affiche pas 42 (code mort)

Deux problèmes

1. `IF b [RETURN 1] [RETURN 0]; ECHO 42`
   - est syntaxiquement correct
   - contient du code mort
2. `IF b [SET x 1] [RETURN 0]`
   - romp l'homogénéïté des branches d'une alternative

Une solution : analyse statique de type

Analyse de flot

Ci-dessous calcule si `f` s'annule sur [a, b] (code souhaitable)

```
SET x a;
WHILE (lt x b)
  [ IF (eq (f x) 0)
       [ RETURN true ]
       [ SET x (add x 1) ] ];
RETURN false
```

- la séquence totale est de type `bool`
- quel est le type du `IF` : `bool` ou `void` ?

Réponse : le `IF` est de type «`bool` ou `void`» !

$$\textit{Type somme} : t + \texttt{void}$$

Nota : `void`+`void` = `void`

Alternative

Typage

3 possibilités

(IF0) si $\Gamma \vdash_{\text{EXPR}} e : \texttt{bool}$,
si $\Gamma \vdash_{\text{BLOCK}} bk_1 : t$ et $\Gamma \vdash_{\text{BLOCK}} bk_2 : t$
alors $\Gamma \vdash_{\text{STAT}} (\texttt{IF}\ e\ bk_1\ bk_2) : t$

(IF1) si $\Gamma \vdash_{\text{EXPR}} e : \texttt{bool}$,
si $\Gamma \vdash_{\text{BLOCK}} bk_1 : \texttt{void}$
et $\Gamma \vdash_{\text{BLOCK}} bk_2 : t$ avec $t \neq \texttt{void}$,
alors $\Gamma \vdash_{\text{STAT}} (\texttt{IF}\ e\ bk_1\ bk_2) : t + \texttt{void}$

(IF2) si $\Gamma \vdash_{\text{EXPR}} e : \texttt{bool}$,
si $\Gamma \vdash_{\text{BLOCK}} bk_1 : t$ avec $t \neq \texttt{void}$,
et si $\Gamma \vdash_{\text{BLOCK}} bk_2 : \texttt{void}$
alors $\Gamma \vdash_{\text{STAT}} (\texttt{IF}\ e\ bk_1\ bk_2) : t + \texttt{void}$

Boucle

Typage

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

- $+\texttt{void}$ si le corps de la boucle
  - n'est pas exécuté
  - ou n'exécute pas de `RETURN`
- $(\texttt{WHILE}\ e\ bk) : \texttt{void}$ si $t = \texttt{void}$ $\qquad (\texttt{void} + \texttt{void} = \texttt{void})$

`ECHO`, `SET` et `CALL` restent `void`

Définitions

Typage

Fonctions procédurales

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

(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
$\Gamma \vdash_{\text{DEC}} (\texttt{FUN REC}\ x\ t\ [x_1 : t_1, \ldots, x_n : t_n]\ bk)$
$: \Gamma[x : t_1\ \texttt{*}\ \ldots\ \texttt{*}\ t_n\ \texttt{->}\ t]$

Remarque : t ne peut être `void` (restriction syntaxique)

Code mort

Analyse de flux

Syntaxe : `RETURN` en fin de suite (bloc) pour éléminer

```
RETURN 1; ECHO 42
```

Comment éliminer ?

```
IF b [RETURN 1] [RETURN 0]; ECHO 42
```

Propriété du typage

une instruction exécute toujours un `RETURN`
si et seulement si
elle n'est pas de type `void`

Typage : forcer une instruction qui n'est pas de type `void` à être en fin de suite

Suite de commandes

Typage

2 cas de continuation

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

3 cas de fin de suite

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

(RET) si $\Gamma \vdash_{\text{EXPR}} e : t$ alors $\Gamma \vdash_{\text{CMDS}} (\texttt{RETURN}\ e\,;\varepsilon) : t$

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

La règle (DEC) ne change pas

Bloc et rogramme

Typage

Un bloc peut avoir un type quelconque :

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

Un programme ne produira pas de valeur (pas de `RETURN`)
Son bloc est de type `void`

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

Sémantique

- Une suite de commandes (ou un bloc) peut produire une valeur . . .ou pas.
- Une expression peut produire des effets mémoire [1] ou sur la flux de sortie.

$$\text{Valeur ou pas : } V_\varepsilon = V \cup \{\varepsilon\}$$

Domaines des relations sémantiques :

$$\vdash_{\textsc{Cmds}} \; E \times S \times O \times \textsc{Cmds} \times V_\varepsilon \times S \times O$$
$$\vdash_{\textsc{Dec}} \; E \times S \times O \times \textsc{Dec} \times E \times S \times O$$
$$\vdash_{\textsc{Stat}} \; E \times S \times O \times \textsc{Stat} \times V_\varepsilon \times S \times O$$
$$\vdash_{\textsc{Expr}} \; E \times S \times O \times \textsc{Expr} \times V \times S \times O$$

1. Déjà dans APS2 avec `alloc`

Expressions

Sémantique

Application des fonctions procédurales : $x \in \texttt{ident}$

(AFP) 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 $\rho', \sigma_n, \omega_n \vdash_{\text{BLOCK}} bk \rightsquigarrow (v, \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (x\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma', \omega')$

(AFPR) 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 $\rho, \sigma_{n-1}, \omega_{n-1} \vdash_{\text{EXPAR}} e_n \rightsquigarrow (v_n, \sigma_n, \omega_n)$
si $\rho' = r(v_1, \ldots, v_n)$
et $\rho', \sigma_n, \omega_n \vdash_{\text{BLOCK}} bk \rightsquigarrow (v, \sigma', \omega')$
alors $\rho, \sigma, \omega \vdash (x\ e_1 \ldots e_n) \rightsquigarrow (v, \sigma', \omega')$

Autres expressions

Sémantique

Amender les règles de $APS2$ pour prendre en compte les effets.

Exemple :

(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 $\rho, \sigma, \omega \vdash_{\text{EXPR}} \texttt{(if } e_1\ e_2\ e_3\texttt{)} \rightsquigarrow (v, \sigma'', \omega'')$

(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 $\rho, \sigma, \omega \vdash_{\text{EXPR}} \texttt{(if } e_1\ e_2\ e_3\texttt{)} \rightsquigarrow (v, \sigma'', \omega'')$

Instructions

Sémantique 1

`ECHO` et `SET` ne produisent pas de valeur (ε)

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

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

Instructions

Sémantique 2

`CALL` non plus (voir typage)

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

Instructions

Sémantique 3

Boucle : 3 cas.

Sortie nominale

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

Sortie anticipée

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

Continuation

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

Autre instruction, *lvalue*, paramètres

Sémantique 4

Amender les régles pour tenir compte des éventuelles effets

Exemples :

(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 $\rho, \sigma, \omega \vdash_{\text{STAT}} (\texttt{IF}\ e\ bk_1\ bk_2) \rightsquigarrow (v, \sigma'', \omega'')$

(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 $\rho, \sigma, \omega \vdash_{\text{LVAL}} (\texttt{nth}\ e_1\ e_2) \rightsquigarrow (a' + 1, \sigma'', \omega'')$

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

Définitions

Sémantique

Définition de constante :

évaluation d'une expression $\Rightarrow$ effet possible

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

Amender les autres règles selon la nouvelle signature

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

Blocs, suites de commandes

Sémantique 1

Bloc

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

Continuation : déclaration et instruction sans valeur

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

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

Blocs, suites de commandes

Sémantique 1

3 cas d'arrêt

Instruction qui produit une valeur (voir typage)

(STATS1) si $\rho, \sigma, \omega \vdash_{\text{STAT}} s \rightsquigarrow (v, \sigma', \omega')$ avec $v \neq \varepsilon$
alors $\rho, \sigma, \omega \vdash_{\text{CMDS}} (s\,;\ \varepsilon) \rightsquigarrow (v, \sigma'', \omega'')$

Commande `RETURN` (voir syntaxe)

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

Suite vide

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

Programme

Sémantique

Un programme ne produit pas de valeur (voir typage)

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

Suites possibles

- Fuites mémoire systématique
  $\Rightarrow$ modéliser un récupérateur automatique de mémoire
  (problème assez difficile)
- Généraliser la rupture de flux d'exécution (`RETURN`)
  $\Rightarrow$ *continuations*
  (sémantique dénotationnelle)