

# Sémantique élémentaire des langages: sémantiques d'évaluation d'un langage avec fonctions

Didier Buchs

Université de Genève

13 avril 2018

# Sémantique d'évaluation d'un langage

Langage étendu par rapport aux expressions arithmétiques en différentes étapes

- Variables
- Structures de contrôles :
  - IF THEN ELSE
  - Affectation
  - Sequence
  - WHILE DO
- Fonctions

# Langage avec Variables

Nous désirons connaître le résultat de l'évaluation d'expressions contenant des variables, syntaxiquement les variables sont un genre de constantes.

## Definition (Expressions arithmétiques avec variables)

- Les expressions doivent être construites sur les nombres et sur les opérateurs usuels.
- Soit  $V$  l'ensemble des variables
- $Exp_V = T_{\{+,-,*,/\}}(\mathbb{N} \cup \underline{V})$

Exemple :

$$(3 * v) + 2 \in T_{\{+,-,*,/\}}(\mathbb{N} \cup \{v\})$$

$$(3 * v) + w \in T_{\{+,-,*,/\}}(\mathbb{N} \cup \{v, w\})$$

Sémantiquement les variables doivent être interprétées différemment.

# Contexte d'évaluation : assignation

Un contexte d'évaluation est ici un ensemble de substitution de variables par des valeurs. Il faut indiquer les valeurs que vont prendre chaque variables dans son domaine (ici Entier).

## Definition (Assignation)

- Soit  $V$  l'ensemble des variables et  $Exp_V = T_{\{+, -, *, /\}}(\mathbb{N} \cup V)$
- Les assignation sont des fonctions des variables dans les valeurs :  $assign : V \rightarrow \mathbb{N}$

domaine des valeurs de variable  
dans notre cas  $\mathbb{N}$



# Contexte d'évaluation : substitution

## Definition (Substitution de variables)

- Soit  $V$  l'ensemble des variables et  $Exp_V = T_{\{+,-,*,/\}}(\mathbb{N} \cup V)$
- Les substitutions sont des fonctions des termes et assignation dans les termes :

$$subs : T_{\{+,-,*,/\}}(\mathbb{N} \cup V) \times assign \rightarrow T_{\{+,-,*,/\}}(\mathbb{N} \cup V)$$

Notation :  $\underline{S} / [v = n]$  signifie que l'assignation de la variable  $v$  prend la valeur  $n$ , la substitution  $S$  est enrichie de cette assignation.

ajouter à  $S$  le fait que  $v$  prend la valeur  $n$

# Exemple de substitution

Exemple de substitution :  $(\epsilon/[v=2])/[w=5]((3 * v) + w) = (3 * 2) + 5$

Les substitutions peuvent être définies inductivement :

## Definition (Substitutions)

soit un ensemble d'opérations  $OP$  et  $V$  un ensemble de variables, une substitution est une relation satisfaisant les propriétés suivantes :

$$\frac{}{\epsilon \in Subs_{OP,C,V}} \quad \frac{s \in Subs_{OP,C,V}, v \in V, e \in T_{OP}(C \cup V)}{s/[v = e] \in Subs_{OP,C,V}}$$

Propriétés :

- $(S/[x = n])/[x = m] = (S/[x = m])$
  - $(S/[x = n])/[y = m] = (S/[y = m])/[x = n]$  si  $x \neq y$
  - $Dom(S/[x = n]) = Dom(S) \cup \{x\}$  et  $Dom(\epsilon) = \emptyset$
- exposant* (pointing to  $v$  in the denominator)  
*efface les arbitraires précédentes* (pointing to  $x$  in the denominator)  
*indifférent* (pointing to the condition  $x \neq y$ )

# Evaluation d'expressions avec variables :

- Relation d'évaluation :  $eval : (Exp_V \times Subs) \times \mathbb{N}$
- Notation :  $e \in Exp_V = T_{\{+, -, *, /\}}(\mathbb{N} \cup V)$  et  $n \in \mathbb{N}$  on utilise :  
 $S \vdash e \Rightarrow n$  pour  $(e, S, n) \in eval$ .

## Definition (Sémantique d'évaluation)

$e \in ExpVar = T_{\{+, -, *, /\}}(\mathbb{N} \cup V)$  et  $n \in \mathbb{N}$ ,  $s \in Subs$

$+_{\mathbb{N}}, *_{\mathbb{N}}, -_{\mathbb{N}}, /_{\mathbb{N}}$  sont les fonctions sur  $\mathbb{N}$

$$R \text{ Constante : } \frac{}{S \vdash n \Rightarrow n}$$

$$R_+ : \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e + e' \Rightarrow n +_{\mathbb{N}} n'}$$

$$R_- : \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e - e' \Rightarrow n -_{\mathbb{N}} n'}$$

$$R \text{ var : } \frac{}{S/[v = n] \vdash v \Rightarrow n}$$

$$R_* : \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e * e' \Rightarrow n *_{\mathbb{N}} n'}$$

$$R/ : \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e / e' \Rightarrow n /_{\mathbb{N}} n'}$$

Remarque : la relation d'évaluation est définie pour les expressions dont les variables sont définies dans la substitution, sinon elle est indéfinie.

# Evaluation d'une expression arithmétique

Exemple :

$$\epsilon/[x = 4] \vdash 3 + x * 2 \Rightarrow$$

$$\epsilon/[y = 4] \vdash 3 + x * 2 \Rightarrow$$

pas evaluable car

$\epsilon/y = 4 \vdash x \Rightarrow 4$   
n'est  
pas  
evaluable

$$\frac{\epsilon/x = 4 \vdash x \Rightarrow 4, \vdash 2 \Rightarrow 2}{\vdash 3 \Rightarrow 8} R_*$$

$$\frac{\vdash 3 \Rightarrow 8, \vdash x * 2 \Rightarrow 8}{\vdash 3 + x * 2 \Rightarrow 11} R_+$$

$$R_+ \quad \epsilon/[x = 4] \vdash 3 + x * 2 \Rightarrow 11$$



Variable non  
définie

# Langage avec structures de contrôle et affectation

Relations sur les entiers.

Instructions avec :

- if\_then\_else :  $Rel_V \times Bloc_V \times Bloc_V \rightarrow Instr_V$
- while\_do\_ :  $Rel_V \times Bloc_V \rightarrow Instr_V$
- := (affectation) :  $V \times Expr_V \rightarrow Instr_V$

*relat*  
*? une sequence d'instructions*

## Definition (Relations et Instructions)

- Soit  $V$  l'ensemble des variables
- Les expressions  $ExprVar_V$  construites sur les nombres et sur les opérateurs usuels.
- $Rel_V = T_{\{<, \leq, >, \geq, =\}}(Expr_V)$
- $Instr_V = T_{\{if\_then\_else, while\_do, :=\}}(Expr_V \cup Rel_V)$

Les blocs sont des séquences d'instructions (ou rien) :

- $\epsilon : \rightarrow \text{Bloc}_V$
- $;\_ : \text{Instr}_V \times \text{Bloc}_V \rightarrow \text{Bloc}_V$

## Definition (Programmes)

- Soit  $V$  l'ensemble des variables
- Les instructions  $\text{Instr}_V$  construites sur les nombres et sur les opérateurs usuels.
- $\text{Bloc}_V = T_{\{;\_, \epsilon\}}(\text{Instr}_V)$

# Exemple de programmes et d'instructions

```
c  := 3;  
s  := 0;  
while c > 0 do  
    ( s:= s+ c;  
      c:= c -1;e);e
```

e : est le programme vide

# Relations d'évaluations

Nous définissons des relations d'évaluations pour chaque domaine syntaxique :

Les instructions :  $Subs \vdash Instr_V \Longrightarrow_I Subs$

Les blocs :  $Subs \vdash Bloc_V \Longrightarrow_B Subs$

$$S \vdash e \Rightarrow n$$

---

$$S \vdash v := e \Rightarrow S \setminus [v = n]$$



# Evaluation d'un bloc

## Definition (Sémantique d'évaluation )

$i \in Instr_V$  et  $p \in Bloc_V$  ,  $S, S', S'' \in Subs$

$$R \text{ Prog vide : } \overline{S \vdash \epsilon \Longrightarrow_B S}$$

$$Rsequence : \frac{S \vdash i \Longrightarrow_I S', S' \vdash p \Longrightarrow_B S''}{S \vdash i; p \Longrightarrow_B S''}$$

# Evaluation d'une instruction : affectation

Definition (Sémantique d'évaluation : Règle affectation )

$e \in \text{ExprVar}_V$  et  $v \in V$  ,  $S, S', S'' \in \text{Subs}$

$$\text{Rassignation : } \frac{S \vdash e \Longrightarrow n}{S \vdash v := e \Longrightarrow_I S/[v = n]}$$

# Satisfaction d'une relation

Definition (Sémantique d'évaluation : Règle  $<$  )

$e, e' \in ExprVar_V$  ,  $n, n' \in \mathbb{N}$  ,  $S \in Subs$

$$R <: \frac{S \vdash e \implies n, S \vdash e' \implies n', n <_{\mathbb{N}} n'}{S \vdash e < e'}$$

Idem pour les autres relations, la non satisfaction se note

$S \not\vdash e < e'$

# Evaluation d'une instruction : if then else

Definition (Sémantique d'évaluation : Règles IFTHENELSE )

$p, p' \in \text{Bloc}_V$  et  $r \in \text{Rel}_V$ ,  $S, S' \in \text{Subs}$

$$\text{RIFTHEN} : \frac{S \vdash r, S \vdash p \Rightarrow_B S'}{S \vdash \text{if } r \text{ then } p \text{ else } p' \Rightarrow_I S'}$$

$$\text{RIFELSE} : \frac{S \not\vdash r, S \vdash p' \Rightarrow_B S'}{S \vdash \text{if } r \text{ then } p \text{ else } p' \Rightarrow_I S'}$$

R

$$\frac{S \not\vdash r}{S \vdash \text{while } r \text{ do } p \Rightarrow S}$$

$$S \vdash r, S \vdash p \Rightarrow S', S' \vdash \text{while } r \text{ do } p \Rightarrow S''$$

$$S \vdash \text{while } r \text{ do } p \Rightarrow_{\rightarrow} S''$$

16/32

# Evaluation d'une instruction : while do

Definition (Sémantique d'évaluation : Règles WHILE )

$p \in \text{Bloc}_V$  et  $r \in \text{Rel}_V$  ,  $S, S' \in \text{Subs}$

$$R\text{WHILED0} : \frac{S \vdash r, S \vdash p; \text{while } r \text{ do } p \Rightarrow_B S'}{S \vdash \text{while } r \text{ do } p \Rightarrow_I S'}$$

$$R\text{DONE} : \frac{S \not\vdash r}{S \vdash \text{while } r \text{ do } p \Rightarrow_I S}$$

# Evaluation d'un bloc de programme

```
c := 3;  
s := 0;  
while c > 0 do  
  ( s:= s+ c;  
    c:= c -1;e);e
```

e : est le programme vide

# Evaluation d'un programme

Loop2' :

$$\frac{\frac{[c=2,s=3] \vdash c \Rightarrow 3}{[c=2,s=3] \vdash c > 0}, c > \mathbb{N}^0}{\frac{[c=2,s=3] \vdash s \Rightarrow 0, [c=2,s=3] \vdash c \Rightarrow 3}{[c=2,s=3] \vdash s+c \Rightarrow 5}} \quad \frac{\frac{[c=2,s=5] \vdash c \Rightarrow 3, [c=2,s=5] \vdash 1 \Rightarrow 1}{[c=2,s=5] \vdash c-1 \Rightarrow 2}}{[c=2,s=5] \vdash c:=c-1 \Rightarrow_I [c=1,s=5]}, \text{Loop2}$$

Loop1 :

[illegible]

## Exercise

Repeat  $p$  until  $r$

$$S \vdash p \Rightarrow S', S' \not\vdash r, S' \vdash \text{repeat } p \text{ until } r \Rightarrow S''$$

$$\hline S \vdash \text{repeat } p \text{ until } r \Rightarrow S''$$

For  $V = a$  to  $b$  do

$$\frac{S \vdash p \Rightarrow S', S' \vdash r}{S \vdash \text{repeat } p \text{ until } r \Rightarrow S'}$$

$$\frac{S \vdash p \Rightarrow S'', S'' \vdash \text{while } r \text{ until } p \Rightarrow S'}{S \vdash \text{repeat } p \text{ until } r \Rightarrow S'}$$



For  $v = a$  to  $b$  do  $P$   $\in \mathbb{N}$

$$S' \vdash \text{For } v = (a+1) \text{ to } b \text{ do } P \Rightarrow S''$$

$$\frac{S/[v=a] \vdash v < b, S/[v=a] \vdash P \Rightarrow S'}{S \vdash \text{For } v = a \text{ to } b \text{ do } P \Rightarrow S'' - \{v\}}$$

$S \vdash \text{For } v = a \text{ to } b \text{ do } P \Rightarrow S'' - \{v\}$  ← on entière  $v$

— ou —

des alternatives

$$S \not\vdash [v=a] \not\vdash v < b$$

$$S \vdash \text{For } v = a \text{ to } b \text{ do } P \Rightarrow S/[v=a]$$

$S_1$   
on veut  
 $v$

# Programmes avec fonctions

- Une fonction est un morceau de code qui peut être appelé n'importe où dans les expressions.
- Une fonction à un nom (en  $\lambda$  - calcul év. pas)
- Une fonction à des paramètres et retourne un résultat
- Les paramètres d'une fonction sont les paramètres formels
- Lors de l'appel ils deviennent les paramètres effectifs à l'intérieur de la fonction
- La visibilité des variables est limitée à la fonction (pas de variables globales !)

# Programmes avec fonctions : Syntaxe

Soit  $F$  l'espace des noms de fonctions, une arité est définie pour ces fonctions (il y a un seul type dans notre langage)

Les expressions sont étendues avec l'appel de fonctions : Soit  $V$  l'ensemble des variables et  $\underline{Exp_{F,V}} = T_{\{+,-,*,/\} \cup F}(\mathbb{N} \cup V)$  et les relations sur les entiers.

Instructions avec :

- $if\_then\_else : Rel_V \times Bloc_{F,V} \times Bloc_{F,V} \rightarrow Instr_{F,V}$
- $while\_do\_ : Rel_V \times Bloc_V \rightarrow Instr_{F,V}$
- $:= (affectation) : V \times Exp_{F,V} \rightarrow Instr_{F,V}$
- $return : Exp_{F,V} \rightarrow Instr_{F,V}$

l'expression qui calcule le résultat

# Programmes avec fonctions : Syntaxe (2)

## Definition (Relations et Instructions)

- Soit  $V$  l'ensemble des variables
- Les expressions  $ExprVar_V$  construites sur les nombres et sur les opérateurs usuels.
- $Rel_{F,V} = T_{\{<, \leq, >, \geq, =\}}(Exp_{F,V})$
- $Instr_{F,V} = T_{\{if\_then\_else, while\_do, :=, return\}}(Exp_{F,V} \cup Rel_{F,V})$

## Programmes avec fonctions : Syntaxe (3)

Les blocs sont des séquences d'instructions (ou rien) :

- $\epsilon : \rightarrow Bloc_{F,V}$
- $;- : Instr_{F,V} \times Bloc_{F,V} \rightarrow Bloc_{F,V}$

### Definition (Blocs, Fonctions et Programmes)

- Soit  $V$  l'ensemble des variables
- Les instructions  $Instr_{F,V}$  construites sur les nombres et sur les opérateurs usuels.

$$Bloc_{F,V} = T_{\{-;-, \epsilon\}}(Instr_{F,V})$$

- Les fonctions  $Func_{F,V}$  construites sur les blocs et le nom de la fonction avec ses paramètres.

$Func_{F,V} = T_F(V) \times Bloc_{F,V}$  corps de la fonction  $f(x,y)$

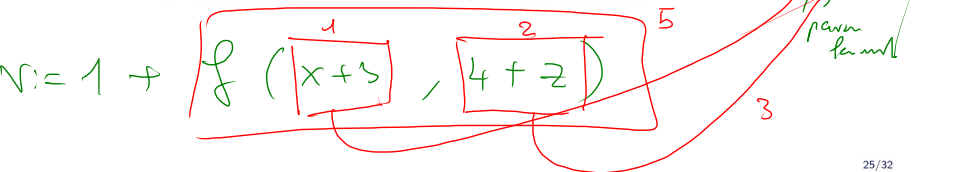
- Les programmes composés de fonctions et d'un corps

$$Prog_{F,V} = \wp(Func_{F,V}) \times Bloc_{F,V}$$

# Sémantique des programmes avec fonctions

Nous allons reprendre les relations précédentes, avec les changements suivants :

- Evaluer une fonction nécessite :
  - d'associer paramètres formels avec les paramètres effectifs
  - La visibilité des variables est limitée à la fonction
  - d'évaluer le corps de la fonction, le résultat étant fournis par l'instruction particulière 'return'. Ceci nécessite de stopper l'évaluation sitôt un 'return' exécuté (mécanisme de continuation) !
- le contexte inclus les définitions de fonctions



# Exemple de programme

```
| square x return x*x; e
```

```
rootsquare x  
  if x=0 then y:= 0 ;e  
  else  
    y:= 1;  
    while square(y) < x do  
      ( y:= y+1;e); e  
    endif;  
  return y; e
```

rootsquare(4);e ) corps du programme

e : est le programme vide

# Relations d'évaluations

Nous définissons des relations d'évaluations pour chaque domaine syntaxique :

- Les expressions :  $\wp(Func_{F,V}), Subs \vdash Expr_{F,V} \Rightarrow \mathbb{N}$
- Les instructions :  $\wp(Func_{F,V}), Subs \vdash Instr_{F,V} \Rightarrow_I Subs \times (\mathbb{N} \cup \{\perp\})$
- Les blocs :  $\wp(Func_{F,V}), Subs \vdash Bloc_{F,V} \Rightarrow_B Subs \times (\mathbb{N} \cup \{\perp\})$
- Les fonctions : L'évaluation est intégrée dans l'évaluation des expressions
- Les programmes :  $\wp(Func_{F,V}), Subs \vdash Bloc_V \Rightarrow_P Subs$

La valeur de retour  $(\mathbb{N} \cup \{\perp\})$  gère explicitement la non-définition du retour.

Nous allons examiner les différences principales avec l'évaluation simple des blocs sans fonctions.

resultat  
non calculé  
(pas de  
return)

$\perp$   
indique  
que la  
valeur  
de retour  
n'est  
pas  
comme

27/32



# Evaluation d'une instruction : return

Definition (Sémantique d'évaluation : Règle du retour )

$e \in ExprVar_V$  et  $v \in V$  ,  $S, S', S'' \in Subs$

$$R_{affectation} : \frac{S \vdash e \Longrightarrow n}{S \vdash \text{return } e \Longrightarrow_I \langle S, n \rangle}$$

# Evaluation d'un bloc

Le principal problème est d'assurer le 1er 'return', le reste des évaluations étant abandonnées

## Definition (Sémantique d'évaluation)

$i \in Instr_V$  et  $p \in Bloc_V$ ,  $S, S', S'' \in Subs$

⊥ bottom

$$\text{R Prog vide : } \frac{}{S \vdash \epsilon \Rightarrow_B \langle S, \perp \rangle}$$

$$\text{Rsequence : } \frac{S \vdash i \Rightarrow_I \langle S', \perp \rangle, S' \vdash p \Rightarrow_B \langle S'', m \rangle}{S \vdash i; p \Rightarrow_B \langle S'', m \rangle}$$

$$\text{Rsequenceceret : } \frac{n \neq \perp, S \vdash i \Rightarrow_I \langle S', n \rangle}{S \vdash i; p \Rightarrow_B \langle S', n \rangle}$$

Seulement d'un, l'autre

Le même principe doit être appliqué pour les règles sur le domaine  $Instr_F, V$

# Evaluation d'une fonction dans une expression

## Definition (Sémantique d'évaluation de l'appel de fonction)

$$\begin{array}{c}
 \langle f(x_1, \dots, x_n), b \rangle \in F, \\
 F, S \vdash e_1 \Rightarrow m_1, \dots, F, S \vdash e_n \Rightarrow m_n, \epsilon/[x_1 = m_1]/\dots/[x_n = m_n] \vdash b \Rightarrow_B \langle S', m \rangle \\
 \hline
 F, \underline{S} \vdash f(e_1, \dots, e_n) \Rightarrow m
 \end{array}$$

*Handwritten notes:*  
 - "contexte de la fonction" (circled in red) points to  $\langle f(x_1, \dots, x_n), b \rangle \in F$ .  
 - "val. globale" (green) points to  $S$ .  
 - "val. locale" (green) points to  $S'$ .  
 - "ou ou bl les valeurs locales" (red) points to  $S'$ .  
 - "paramètres" (circled in blue) points to  $e_1, \dots, e_n$ .

Explication :

- $f(e_1, \dots, e_n)$  est l'appel de la fonction  $f$  dans une expression
- $\langle f(x_1, \dots, x_n), b \rangle \in F$  est la définition de la fonction  $f$  à appeler.  
 $x_1, \dots, x_n$  sont les paramètres formels et  $b$  est le corps de la fonction.
- $F, S \vdash e_1 \Rightarrow m_1, \dots, F, S \vdash e_n \Rightarrow m_n$ , calcule les paramètres effectifs
- $\epsilon/[x_1 = m_1]/\dots/[x_n = m_n]$  construit l'assignation des paramètres formels aux paramètres effectifs
- $\epsilon/[x_1 = m_1]/\dots/[x_n = m_n] \vdash b \Rightarrow_B \langle S', m \rangle$  évalue le bloc  $b$  pour les valeurs prises par les paramètres formels, le résultat est  $m$

*Handwritten notes:*  
 - "arrivées avec des variables globales" (green) points to the environment update.  
 - "pas de Variables globales!" (black) points to the global environment.

# Propriétés et limites

Propriété de l'évaluation de fonction :

- L'absence de 'return' rend la fonction indéfinie
- Pas d'effet de bord i.e.

$\forall i$

$$\forall S, x \notin \text{Dom}(S), F, S \vdash x := f(e_1, \dots, e_n) \Rightarrow_I S' \\ \Rightarrow S' = S/[x = m]$$

- L'évaluation est déterministe i.e.

$$\forall S, (x \notin \text{Dom}(S), F, S \vdash x := f(e_1, \dots, e_n) \Rightarrow_I S', \\ y \notin \text{Dom}(S), F, S \vdash y := f(e_1, \dots, e_n) \Rightarrow_I S'') \\ \Rightarrow S''(y) = S'(x)$$

$$P_1(\{e_i\}, S)$$

$$P_2(\{e_i\}, S)$$

$$x \notin \text{Dom} \quad P(x) = \underline{\hspace{2cm}}$$

# Sujets supplémentaires

Ne sont pas couverts par cette présentation :

- Les aspects statiques de définition de types et de variables typées.
- Les aspects dynamiques de visibilité des variables globales et locales.
- Les autres structures de données, les pointeurs, les objets, les entrées-sorties.
- Les passages de paramètres par nom et par besoin.

