

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

Didier Buchs

Université de Genève

23 février 2018

1/32



# Sémantique d'évaluation d'un langage

Langage étendu par rapport aux expressions arithmétiques en différentes étapes

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

## Langage avec Variables

Nous désirons connaitre 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
- $Exp_V = T_{\{+,-,*,/\}}(\mathbb{N} \cup V)$

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.

3/32



## 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 variable dans son domaine (ici Entier).

### Definition (Assignation)

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

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

## Exemple de substitution

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

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 :

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

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  $Dom(\epsilon) = \emptyset$

## Evaluation d'expressions avec variables :

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

$$S + e \Rightarrow n$$

### Definition (Sémantique d'évaluation )

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

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

$$\begin{aligned} R \text{ Constante} : & \frac{}{S \vdash n \Rightarrow n} \\ R_+ : & \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e + e' \Rightarrow n +_{\mathbb{N}} n'} \\ R_- : & \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e - e' \Rightarrow n -_{\mathbb{N}} n'} \end{aligned}$$

$$\begin{aligned} R \text{ var} : & \frac{}{S/[v = n] \vdash v \Rightarrow n} \\ R_* : & \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e * e' \Rightarrow n *_{\mathbb{N}} n'} \\ R/_* : & \frac{S \vdash e \Rightarrow n, S \vdash e' \Rightarrow n'}{S \vdash e/e' \Rightarrow n/_{\mathbb{N}} n'} \end{aligned}$$

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

7/32

## Evaluation d'une expression arithmétique

Exemple :

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

$$\epsilon / [y = 4] \vdash 3 + x * 2 \implies \text{indefini}$$

$$\begin{array}{c} R_{\text{comb}} \\ \dfrac{\dfrac{\dfrac{\overbrace{[x=4] \vdash x \Rightarrow 4}^{\overbrace{[x=4] \vdash x \Rightarrow 4}} \quad \overbrace{+ 2 \Rightarrow 2}^{\overbrace{+ 2 \Rightarrow 2}}}{[x=4] \vdash 3 \Rightarrow 3} \quad \dfrac{\overbrace{[x=4] \vdash x * 2 \Rightarrow 8}^{\overbrace{[x=4] \vdash x * 2 \Rightarrow 8}}}{[x=4] \vdash 3 + x * 2 \Rightarrow 11}} \end{array}$$

# Langage avec structures de contrôle et affectation

Relations sur les entiers.

Instructions avec :

- $\text{if\_then\_else} : \text{Rel}_V \times \text{Bloc}_V \times \text{Bloc}_V \rightarrow \text{Instr}_V$
- $\text{while\_do\_} : \text{Rel}_V \times \text{Bloc}_V \rightarrow \text{Instr}_V$
- $\text{:= (affectation)} : V \times \text{Expr}_V \rightarrow \text{Instr}_V$

## Definition (Relations et Instructions)

- Soit  $V$  l'ensemble des variables
- Les expressions  $\text{ExprVar}_V$  construites sur les nombres et sur les opérateurs usuels.
- $\boxed{\text{Rel}_V = T_{\{<, \leq, >, \geq, =\}}(\text{Expr}_V)}$
- $\boxed{\text{Instr}_V = T_{\{\text{if\_then\_else}, \text{while\_do}, :=\}}(\text{Expr}_V \cup \text{Rel}_V)}$

# Blocs

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

- $\epsilon : \rightarrow Bloc_V$
- $-; - : Instr_V \times Bloc_V \rightarrow \underline{Bloc_V}$

## Definition (Programmes)

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

## 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 :  $Subs \vdash Instr_V \Rightarrow_I Subs$

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

$$\frac{\begin{array}{c} S \vdash e \Rightarrow_B S \\ S \vdash i \Rightarrow S'', S'' + B \Rightarrow S' \end{array}}{S \vdash i; B \Rightarrow S'}$$

## Evaluation d'un bloc

Definition (Sémantique d'évaluation )

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

$$R \text{ Prog vide} : \frac{}{S \vdash \epsilon \Rightarrow_B S}$$

$$R_{sequence} : \frac{S \vdash i \Rightarrow_I S', S' \vdash p \Rightarrow_B S''}{S \vdash i; p \Rightarrow_B S''}$$

## Evaluation d'une instruction : affectation

Definition (Sémantique d'évaluation : Règle affectation )

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

$$Raffection : \frac{S \vdash e \implies n}{S \vdash v := e \implies_I S/[v = n]}$$

## 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 )

$p, p' \in Bloc_V$  et  $r \in Rel_V$ ,  $S, S' \in Subs$

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

choix

## Evaluation d'une instruction : while do

Definition (Sémantique d'évaluation : Règles WHILE )

$p \in Bloc_V$  et  $r \in Rel_V$ ,  $S, S' \in Subs$

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

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

$\downarrow$

$$\frac{S \vdash r, S \vdash p \Rightarrow_B S', S' \vdash \text{while } r \text{ do } p \Rightarrow_B S''}{S \vdash \text{while } r \text{ do } p \Rightarrow_{\pm} S''}$$

repeat ?

repeat p until r

$$\frac{S \vdash p \Rightarrow_B S', \quad S' \not\vdash r}{S \vdash \text{repeat } p \text{ until } r \Rightarrow_I S'}$$

$$\frac{S \vdash p \Rightarrow_B S'', \quad S'' + r, \quad S'' \vdash \text{repeat } p \text{ until } r}{S \vdash \text{repeat } p \text{ until } r \Rightarrow_I S'}$$

## 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' :

$$\frac{\frac{[c=2,s=3] \vdash s \Rightarrow 0, [c=2,s=3] \vdash c \Rightarrow 3}{[c=2,s=3] \vdash s+c \Rightarrow 5}, [c=2,s=5] \vdash c \Rightarrow 3, [c=2,s=5] \vdash 1 \Rightarrow 1}{[c=2,s=5] \vdash c-1 \Rightarrow 2} , Loop3$$

$$\frac{[c=2,s=3] \vdash c \Rightarrow 3, [c=2,s=3] \vdash 0 \Rightarrow 0, c > \mathbb{N}^0}{[c=2,s=3] \vdash c > 0}, \frac{[c=2,s=3] \vdash s := s + c \Rightarrow [c=2,s=5]}{[c=2,s=5] \vdash c := c - 1; while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p}$$

$$\frac{[c=2,s=3] \vdash c > 0}{[c=2,s=5] \vdash (s := s + c; c := c - 1); while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p[c=0,s=6]}$$

Loop1 :

$$\dots$$

$$\frac{\frac{\frac{\frac{\epsilon / [c=3] / [s=0] \vdash s \Rightarrow 0, \epsilon / [c=3] / [s=0] \vdash c \Rightarrow 3}{\epsilon / [c=3] / [s=0] \vdash s+c \Rightarrow 3}, \epsilon / [c=3] / [s=3] \vdash c \Rightarrow 3}{\epsilon / [c=3] / [s=0] \vdash s := s + c \Rightarrow [c=3] / [s=3]}, \epsilon / [c=3] / [s=3] \vdash c := c - 1;}{\epsilon / [c=3] / [s=0] \vdash (s := s + c; c := c - 1); while\ c > 0\ do(s := s + c) \Rightarrow p[c=0,s=6]}}{\epsilon / [c=3] / [s=0] \vdash c > 0}$$

$$\frac{\frac{\frac{\frac{\epsilon / [c=3] \vdash 0 \Rightarrow 0}{\epsilon / [c=3] \vdash s := 0, \epsilon / [c=3] / [s=0] \vdash while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p}, \epsilon / [c=3] / [s=0] \vdash while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p \epsilon / [c=0] / [s=6]}{\epsilon / [c=3] / [s=0] \vdash s := 0; while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p \epsilon / [c=0] / [s=6]}}{\epsilon / [c=3] / [s=0] \vdash c := 3; s := 0; while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p \epsilon / [c=0] / [s=6]}$$

$$\frac{\frac{\frac{\frac{\epsilon / [c=3] \vdash 0 \Rightarrow 0}{\epsilon / [c=3] \vdash s := 0, \epsilon / [c=3] / [s=0] \vdash while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p}, \epsilon / [c=3] / [s=0] \vdash while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p \epsilon / [c=0] / [s=6]}{\epsilon / [c=3] / [s=0] \vdash s := 0; while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p \epsilon / [c=0] / [s=6]}}{\epsilon / [c=3] / [s=0] \vdash c := 3; s := 0; while\ c > 0\ do(s := s + c; c := c - 1) \Rightarrow p \epsilon / [c=0] / [s=6]}$$

# Exercice

20/32



## 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 ev. 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  $Exp_{F,V} = T_{\{+,-,*,/\}} \cup F(\mathbb{N} \cup V)$  et les relations sur les entiers.

Instructions avec :

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

## 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.  
$$Func_{F,V} = T_F(V) \times Bloc_{F,V}$$
- Les programmes composés de fonctions et d'un corps  
$$Prog_{F,V} = \wp(Func_{F,V}) \times Bloc_{F,V}$$

24/32



# Sémantique des programmes avec fonctions

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

$\Rightarrow \perp$     $\Rightarrow B$

- 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 sitot 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
```

e : est le programme vide

## Relations d'évaluations

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

- Les expressions :  $\wp(\text{Func}_{F,V}), \text{Subs} \vdash \text{Expr}_{F,V} \Rightarrow \mathbb{N}$
- Les instructions :  
 $\wp(\text{Func}_{F,V}), \text{Subs} \vdash \text{Instr}_{F,V} \implies_I \text{Subs} \times (\mathbb{N} \cup \{\perp\})$  *return*
- Les blocs :  
 $\wp(\text{Func}_{F,V}), \text{Subs} \vdash \text{Bloc}_{F,V} \implies_B \text{Subs} \times (\mathbb{N} \cup \{\perp\})$  *return*
- Les fonctions : L'évaluation est intégrée dans l'évaluation des expressions
- Les programmes :  $\wp(\text{Func}_{F,V}), \text{Subs} \vdash \text{Bloc}_V \implies_P \text{Subs}$   
*↓ indefini (pas de return)*

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.

## Evaluation d'une instruction : return

Definition (Sémantique d'évaluation : Règle du retour )

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

$$Raffection : \frac{S \vdash e \implies n}{S \vdash return\ e \implies_I < S, n >}$$

## 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  $p \in Bloc_V$ ,  $S, S', S'' \in Subs$

pas de return

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

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

calcul pas P

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

Le même principe doit être appliqué pour les règles sur le domaine  
 $Instr_{F,V}$

29/32

## Evaluation d'une fonction dans une expression

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

$$\frac{\boxed{< f(x_1, \dots, x_n), b > \in F,} \quad \text{fonction à évaluer} \\ F, S \vdash e_1 \implies m_1, \dots, F, S \vdash e_n \implies m_n, \boxed{\epsilon / [x_1 = m_1] / \dots / [x_n = m_n] \vdash b \implies_B < S', m >} }{F, S \vdash f(e_1, \dots, e_n) \implies m}$$

Explication :

- $f(e_1, \dots, e_n)$  est l'appel de la fonction  $f$  dans une expression
- $< f(x_1, \dots, x_n), b > \in F$  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.
- $F, S \vdash e_1 \implies m_1, \dots, F, S \vdash e_n \implies m_n$ , calcule les paramètres effectifs
- $\epsilon / [x_1 = m_1] / \dots / [x_n = m_n]$  construit l'assignation des paramètres formels aux paramètres effectifs
- $\epsilon / [x_1 = m_1] / \dots / [x_n = m_n] \vdash b \implies_B < S', m >$  évalue le bloc  $b$  pour les valeurs prises par les paramètres formels, le résultat est  $m$

## Propriétés et limites

Propriété de l'évaluation de fonction :

- L'absence de 'return' rend la fonction indéfinie

$\langle S, \perp \rangle ?$

- Pas d'effet de bord i.e.

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

- L'évaluation est déterministe i.e.

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

## Sujets supplémentaires

Ne sont pas couvert 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.

