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

Didier Buchs

Université de Genève

23 février 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 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 différement.

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 chaques 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 : $assign : V \to \mathbb{N}$

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

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 :

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

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 $Dom(\epsilon) = \emptyset$

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 $\mathbb{N}$

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

$$R+ : \frac{S \vdash e \Longrightarrow n, S \vdash e' \Longrightarrow n'}{S \vdash e + e' \Longrightarrow n +_{\mathbb{N}} n'} \qquad R* : \frac{S \vdash e \Longrightarrow n, S \vdash e' \Longrightarrow n'}{S \vdash e * e' \Longrightarrow n *_{\mathbb{N}} n'}$$

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

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 :

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

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

Langage avec structures de contrôle et affectation

Relations sur les entiers.
Instructions avec :

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

Definition (Relations et Instructions)

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

Blocs

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

- $\epsilon : \rightarrow Bloc_V$
- $\_;\_ : Instr_V \times Bloc_V \rightarrow Bloc_V$

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 : $Subs \vdash Instr_V \Longrightarrow_I Subs$

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

Evaluation d'un bloc

Definition (Sémantique d'évaluation)

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

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

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

Evaluation d'une instruction : affectation

Definition (Sémantique d'évaluation : Règle affectation)

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

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

Satisfaction d'une relation

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

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

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

Idem pour les autres relations, la non satisfaction se note $S \nvdash e < e'$

Evaluation d'une instruction : if then else

Definition (Sémantique d'évaluation : Règles IFTHENELSE)

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

$$RIFTHEN : \frac{S \vdash r, S \vdash p \Longrightarrow_B S'}{S \vdash \textit{if } r \textit{ then } p \textit{ else } p' \Longrightarrow_I S'}$$

$$RIFELSE : \frac{S \nvdash r, S \vdash p' \Longrightarrow_B S'}{S \vdash \textit{if } r \textit{ then } p \textit{ else } p' \Longrightarrow_I S'}$$

Evaluation d'une instruction : while do

Definition (Sémantique d'évaluation : Règles WHILE)

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

$$RWHILEDO : \frac{S \vdash r, S \vdash p; while\ r\ do\ p \Longrightarrow_B S'}{S \vdash while\ r\ do\ p \Longrightarrow_I S'}$$

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

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' :

$$\dfrac{\dfrac{\overline{[c=2,s=3]\vdash\ c\Rightarrow 3}\ ,\ \overline{[c=2,s=3]\vdash\ 0\Rightarrow 0}\ ,c>_{\mathbb{N}}0}{[c=2,s=3]\vdash\ c>0}\ ,\ \dfrac{\dfrac{\overline{[c=2,s=3]\vdash\ s\Rightarrow 0}\ ,\ \overline{[c=2,s=3]\vdash\ c\Rightarrow 3}}{[c=2,s=3]\vdash s+c\Rightarrow 5}}{[c=2,s=3]\vdash s:=s+c\Rightarrow_I [c=2,s=5]}\quad \dfrac{\dfrac{\overline{[c=2,s=5]\vdash\ c\Rightarrow 3}\ ,\ \overline{[c=2,s=5]\vdash\ 1\Rightarrow 1}}{[c=2,s=5]\vdash c-1\Rightarrow 2}}{[c=2,s=5]\vdash c:=c-1\Rightarrow_I [c=1,s=5]}\ ,Loop3}{[c=2,s=5]\vdash (s:=s+c;c:=c-1);\mathit{while}\ c>0\ \mathit{do}(s:=s+c;c:=c-1)\Rightarrow_P [c=0,s=6]}$$

....

Loop1 :

$$\dfrac{\dfrac{\overline{\epsilon/[c=3]/[s=0]\vdash\ c\Longrightarrow 3}\ ,\ \overline{\epsilon/[c=3]/[s=0]\vdash\ 0\Longrightarrow 0}\ ,c>_{\mathbb{N}}0}{\epsilon/[c=3]/[s=0]\vdash\ c>0}\ ,\ \dfrac{\dfrac{\overline{\epsilon/[c=3]/[s=0]\vdash\ s\Longrightarrow 0}\ ,\ \overline{\epsilon/[c=3]/[s=0]\vdash\ c\Longrightarrow 3}}{\epsilon/[c=3]/[s=0]\vdash s+c;\Longrightarrow 3}}{\epsilon/[c=3]/[s=0]\vdash s:=s+c;\Longrightarrow_I \epsilon/[c=3]/[s=3]}\quad \dfrac{\overline{\epsilon/[c=3]/[s=3]}}{\epsilon/[c=3]/[s=3]\vdash c:=c-1;)}}{\epsilon/[c=3]/[s=0]\vdash (s:=s+c;c:=c-1;);\mathit{while}\ c>0\ \mathit{do}(s:=s+c}$$

....

$$\dfrac{\dfrac{\overline{\epsilon\vdash 3\Longrightarrow 3}}{\epsilon\vdash c:=3\Longrightarrow_I \epsilon/[c=3]}\ ,\ \dfrac{\dfrac{\overline{\epsilon/[c=3]\vdash 0\Longrightarrow 0}}{\epsilon/[c=3]\vdash s:=0,\Longrightarrow_I \epsilon/[c=3]/[s=0]}\ ,\ \overline{\epsilon/[c=3]/[s=0]\vdash \mathit{while}\ c>0\ \mathit{do}(s:=s+c;c:=c-1;)\Longrightarrow_P \epsilon/[c=0]/[s=6]}^{\mathit{Loop1}}}{\epsilon/[c=3]\vdash s:=0;\mathit{while}\ c>0\ \mathit{do}(s:=s+c;c:=c-1;)\Longrightarrow_P \epsilon/[c=0]/[s=6]}}{\epsilon\vdash c:=3;s:=0;\mathit{while}\ c>0\ \mathit{do}(s:=s+c;c:=c-1;)\Longrightarrow_P \epsilon/[c=0]/[s=6]}$$

Exercice

Programmes avec fonctions

- Une fonction est un morceau de code qui peut etre appelé n'importe où dans les expressions.
- Une fonction à un nom (en λ - 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 :

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

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

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 sitot un 'return' éxécuté (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 :

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

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.

Evaluation d'une instruction : return

Definition (Sémantique d'évaluation : Règle du retour)

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

$$Raffectation : \frac{S \vdash e \Longrightarrow n}{S \vdash return\ e \Longrightarrow_I < S, n >}$$

Evaluation d'un bloc

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

Definition (Sémantique d'évaluation)

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

$$\text{R Prog vide} : \frac{}{S \vdash \epsilon \Longrightarrow_B < S, \perp >}$$

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

$$Rsequenceret : \frac{n \neq \perp, S \vdash i \Longrightarrow_I < S', n >}{S \vdash i; p \Longrightarrow_B < S', n >}$$

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)

$$\frac{< f(x_1, .., x_n), b > \in F, \quad F, S \vdash e_1 \Longrightarrow m_1, .., F, S \vdash e_n \Longrightarrow m_n, \ \epsilon/[x_1 = m_1]/.../[x_n = m_n] \vdash b \Longrightarrow_B < S', m >}{F, S \vdash f(e_1, ..., e_n) \Longrightarrow m}$$

Explication :

- $f(e_1, ..., e_n)$ est l'appel de la fonction f dans une expression
- $< f(x_1, ..., x_n), b > \in F$ est la définition de la fonction f à appeler. $x_1, ..., x_n$ sont les paramètres formels et b est le corps de la fonction.
- $F, S \vdash e_1 \Longrightarrow m_1, ..., F, S \vdash e_n \Longrightarrow m_n$, calcule les paramètres effectifs
- $\epsilon/[x_1 = m_1]/.../[x_n = m_n]$ construit l'assignation des paramètres formels aux paramètres effectifs
- $\epsilon/[x_1 = m_1]/.../[x_n = m_n] \vdash b \Longrightarrow_B < S', m >$ 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
- Pas d'effet de bord i.e.
  $\forall S, x \notin Dom(S),\ F, S \vdash x := f(e_1, ..., e_n) \Longrightarrow_I S'$
  $\Rightarrow S' = S/[x = m]$
- L'évaluation est déterministe i.e.
  $\forall S, (x \notin Dom(S),\ F, S \vdash x := f(e_1, ..., e_n) \Longrightarrow_I S',$
  $y \notin Dom(S),\ F, S \vdash y := f(e_1, ..., e_n) \Longrightarrow_I S'')$
  $\Rightarrow S''(y) = S'(x)$

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.