Sémantique élémentaire des langages: sémantiques d'expressions arithmétiques

Didier Buchs

Université de Genève

5 mars 2018

Sémantique d’expression arithmétique

- Sémantique d’évaluation (dénotationelle)
- Sémantique computationelle
- Sémantique concrete

Syntaxe des expressions arithmétiques

Definition (Expressions arithmétiques)

- Les expressions doivent être construites sur les nombres et sur les opérateurs usuels.
- $Exp = T_{\{+,-,*,/\}}(\mathbb{N})$

Exemple :
$(3 * 4) + 2 \in T_{\{+,-,*,/\}}(\mathbb{N})$
$3 * (4 + 2) \in T_{\{+,-,*,/\}}(\mathbb{N})$ [1]

1. Les parenthèses construisent l'arborescence du terme

Exercice

Construire les termes décrivant les listes.

Sémantique d'évaluation des expressions arithmétiques

- Les expressions doivent être évaluées sur les nombres.
- La relation d'évaluation détermine une relation : $eval \subseteq Exp \times \mathbb{N}$
- Notation : $e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ et $n \in \mathbb{N}$ alors on note $e \Longrightarrow n \Leftrightarrow (e, n) \in eval$

Definition (Sémantique d'évaluation)

$e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ et $n \in \mathbb{N}$
$+_{\mathbb{N}}, *_{\mathbb{N}}, -_{\mathbb{N}}, /_{\mathbb{N}}$ sont les fonctions sur $\mathbb{N}$

$$\text{R Constante} : \frac{}{n \Longrightarrow n}$$

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

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

Evaluation d'une expression arithmétique

Example

Soit l'expression 3* 4 à évaluer

$$\frac{\overline{3 \Longrightarrow 3}\ ,\ \overline{4 \Longrightarrow 4}}{3 * 4 \Longrightarrow 12}$$

Soit l'expression (3* 4) + 1 à évaluer

$$\frac{\dfrac{\overline{3 \Longrightarrow 3}\ ,\ \overline{4 \Longrightarrow 4}}{3 * 4 \Longrightarrow 12}\ ,\ \overline{1 \Longrightarrow 1}}{(3 * 4) + 1 \Longrightarrow 13}$$

Exercice : Evaluation d'une expression arithmétique

Propriété de la sémantique d'évaluation des expressions arithmétiques

Theorem (Unicité de la Sémantique d'évaluation)

$\forall e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ *et* $n, n' \in \mathbb{N}$
$e \Longrightarrow n, e \Longrightarrow n' \Rightarrow n = n'$

Démonstration.

- $P(x) = \{x \Longrightarrow n \wedge x \Longrightarrow n' \Rightarrow n = n'\}$
- $P(n)$
- Hyp. P(e) et P(e') vérifiée alors
  - $P(e + e')$
  - $P(e - e')$
  - $P(e * e')$
  - $P(e/e')$

□

Propriété de la sémantique d'évaluation des expressions arithmétiques - 2

Theorem (Existence de la Sémantique d'évaluation)

$\forall e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ *il existe* $n \in \mathbb{N}$ *t.q.* $e \Longrightarrow n$

Démonstration.

- $P(x) = \{\exists k \in \mathbb{N}, x \Longrightarrow k\}$

$\square$

Exercice : sémantique d'un véhicule programmable

Les mouvements possibles : $M = \{L, R, F\}$
Un programme est une liste construites avec (concaténation $\_ :: \_$, liste vide ϵ) de telles instructions.
Les programmes sont donc des termes :
$T_{\{\epsilon, \_::\_\}}(\{L, R, F\}) = T_{\{\epsilon, \_::\_\}}(M)$

Par exemple : suivre un carré correspond au programme :
$square = F :: R :: F :: R :: F :: R :: F :: R :: \epsilon$

Exercice : sémantique d'un véhicule programmable(suite)

Comment donner une sémantique à ce langage ?

- il faut fixer un domaine sémantique et
- définir des règles pour définir une sémantique d'évaluation.

Exercice : sémantique d'un véhicule programmable (suite)

Le domaine sémantique est composé de :
$position \in Direct \times Coord$ où

$$Direct = \{-1, 0, 1\} \times \{-1, 0, 1\} = \{-1, 0, 1\}^2$$

et

$$Coord = \mathbb{Z} \times \mathbb{Z}$$

Exercice : sémantique d'un véhicule programmable(suite)

La relation de transition pour une instruction est donc :

$(Direct \times Coord) \times M \times (Direct \times Coord)$ notée $< d, p >, m \Rightarrow < d, p >$

Les règles pour les actions élémentaires :

- $\dfrac{}{<d,p>,L \Rightarrow < \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} * d,p>}$
- $\dfrac{}{<d,p>,R \Rightarrow < \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} * d,p>}$
- $\dfrac{}{<d,p>,F \Rightarrow <d,p+d>}$

Exercice : sémantique d'un véhicule programmable (suite)

Pour les programmes comme succession d'instruction :
La relation est étendue
$(Direct \times Coord) \times T_{\{\epsilon, \_::\_\}}(M) \times Direct \times Coord$

$$\frac{<d,p>,mov \Rightarrow <d',p'>, <d',p'>,prog \Rightarrow <d'',p''>}{<d,p>,mov::prog \Rightarrow <d'',p''>}$$

Exercice : sémantique d'un véhicule programmable (suite)

Question : Prouver que $\forall p, d :< d, p >, square \rightarrow < d, p >$

les quatres rotations successives donnent :

$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^4 * d = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * d$

les mouvements induisent la nouvelle position :

$p = p + d + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^1 * d + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^2 * d + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^3 * d$

$p = p + (\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^1 + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^2 + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^3) * d$

$p = p + (\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} + \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}) * d$

$p = p + (\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}) * d$

Exercice : Evaluation d'une expression arithmétique avec des chiffres

Correction : Evaluation d'une expression arithmétique avec des chiffres

$$\frac{}{0 \in Ch} \qquad \frac{}{1 \in Ch} \qquad \frac{}{2 \in Ch}$$

$$\frac{}{3 \in Ch} \qquad \frac{}{4 \in Ch} \qquad \frac{}{5 \in Ch}$$

$$\frac{}{6 \in Ch} \qquad \frac{}{7 \in Ch} \qquad \frac{}{8 \in Ch}$$

$$\frac{}{9 \in Ch} \qquad \frac{n \in Ch}{n \in Num} \qquad \frac{n \in Ch, m \in Num}{m'n \in Num}$$

Il s'agit de construire la relation : $T_{\{+\}}(Num) \Rightarrow Num$
Table de calcul sur les chiffres ! ! $T_{\{+\}}(Ch) \Rightarrow Num$

$$
\begin{array}{ccc}
\overline{0+0 \Rightarrow 0} & \overline{0+1 \Rightarrow 1}\cdots & \overline{0+9 \Rightarrow 9} \\
\overline{1+0 \Rightarrow 0} & \overline{1+1 \Rightarrow 1}\cdots & \overline{1+9 \Rightarrow 1'0} \\
\cdots & \cdots & \cdots \\
\overline{9+0 \Rightarrow 9} & \overline{9+1 \Rightarrow 1'0}\cdots & \overline{9+9 \Rightarrow 1'8}
\end{array}
$$

Règles de l'addition décimale

$$\frac{n + m \Rightarrow k, k \in Ch, c + d \Rightarrow r}{c'n + d'm \Rightarrow r'k}$$

Avec la retenue ! !

$$\frac{n + m \Rightarrow ret'k, k \in Ch, c + d \Rightarrow rr, rr + ret \Rightarrow r}{c'n + d'm \Rightarrow r'k}$$

Longueur différente (opérande droite plus longue)

$$\frac{n + m \Rightarrow k, k \in Ch, n \in Ch}{n + d'm \Rightarrow d'k}$$

$$\frac{n + m \Rightarrow ret'k, k \in Ch, n \in Ch, d + ret \Rightarrow r}{n + d'm \Rightarrow r'k}$$

idem opérande gauche plus longue ...

Typage :syntaxe

Il s'agit d'introduire une notion simple de type :

- Un type est attribué à chaque expression, T est l'ensemble des types.
- La compatibilité est liée aux noms.
- un ensemble OP des opérateurs est fournis.
- le profil des opérateurs est indiqué par la fonction $\mu : OP \rightarrow T^* \times T$
- i.e. $\mu(op) = t_1, ..., t_n, s$

Typage :sémantique

Domaine d'évaluation :

- chaque type $t \in T$ a un domaine de valeurs, $Dom(t)$
- chaque opérateur op a une correspondance sémantique op_t.
- si $\mu(op) = t_1 t_2 ... t_n, t$ alors
  $op_t : Dom(t_1) \times Dom(t_2) \times ... \times Dom(t_n) \rightarrow Dom(t)$

Nous allons donc déduire/construire la relation :

$$T \vdash e \Longrightarrow v : t$$

où :
$t \in T, v \in Dom(t), e \in Exp$

Typage :règles

Definition (Sémantique d'évaluation typée)

$e \in Exp = T_{OP}(\emptyset)$

op sont les fonctions sur les types

$$\text{R Constante} : \frac{n \in OP, \mu(n) = \epsilon t, t \in T}{T \vdash n \Longrightarrow n_t : t}$$

$$Rop2 : \frac{op \in OP, \mu(op) = t_1 t_2 t,\ T \vdash e \Longrightarrow n : t_1,\ T \vdash e' \Longrightarrow n' : t_2}{T \vdash e\ op\ e' \Longrightarrow n\ op_t\ n' : t}$$

Sémantique computationelle des expressions arithmétiques

- Calcul de l'effet de chaques étapes intermédiaires
- Forme de sémantique mettant en évidence les calculs effectifs
- La stratégie devient explicite

Nous allons :

- Donner les règles pour l'exemple des expressions arithmétiques
- Montrer leur validité et complétude

Sémantique computationelle des expressions arithmétiques

- La relation d' un pas dévaluation détermine un système de transition : $comp \subseteq Exp \times Exp$
- Notation : $e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ et $n \in \mathbb{N}$ alors on note $e \longrightarrow n \Leftrightarrow (e, n) \in comp$

Definition (Sémantique computationelle)

$e, e', e'' \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ et $n, n' \in \mathbb{N}$ et $+_{\mathbb{N}}, *_{\mathbb{N}}, -_{\mathbb{N}}, /_{\mathbb{N}}$ sont les fonctions sur $\mathbb{N}$

$$RC+ : \frac{}{n + n' \longrightarrow n +_{\mathbb{N}} n'} \qquad RC* : \frac{}{n * n' \longrightarrow n *_{\mathbb{N}} n'}$$

$$RC- : \frac{}{n - n' \longrightarrow n -_{\mathbb{N}} n'} \qquad RC/ : \frac{}{n/n' \longrightarrow n/_{\mathbb{N}} n'}$$

$$\forall op \in \{+, -, *, /\}$$

$$RCL : \frac{e \longrightarrow e''}{e\ op\ e' \longrightarrow e''\ op\ e'} \qquad RCR : \frac{e' \longrightarrow e''}{e\ op\ e' \longrightarrow e\ op\ e''}$$

Calcul d'une expression arithmétique

Example

Soit l'expression (3* 4) +1 à calculer

Solution :

$$\frac{\overline{3*4 \longrightarrow 12}}{(3*4)+1 \longrightarrow 12+1}, \frac{}{12+1 \longrightarrow 13}$$

Exercice : calcul d'une expression arithmétique

Non-déterminisme

Constat : il y a différentes dérivations possibles, c'est le non déterminisme de l'application de RCL et RCR, il s'agit de voir si cela implique un résultat de calcul différent ?

Definition (Fermeture transitive)

soit $\longrightarrow$ un système de transition la fermeture transitive $\longrightarrow^*$ est une relation satisfaisant les propriétés suivantes :

$$RCB : \frac{e \longrightarrow e'}{e \longrightarrow^* e'} \qquad RCI : \frac{e \longrightarrow^* e'', e'' \longrightarrow e'}{e \longrightarrow^* e'}$$

L'application des règles construit une suite de réduction, lorsqu'il n'y a plus de réduction possibles nous parlons de forme irréductibles (ou canonique).

Definition (Forme canonique)

soit $\longrightarrow$ un système de transition la relation canonique $\leadsto$ est une relation satisfaisant les propriétés suivantes :

$$RCanon : \frac{e \longrightarrow^{*} e', e' \not\longrightarrow e''}{e \leadsto e'}$$

$$RBase : \frac{}{n \leadsto n}$$

Propriété de la sémantique computationelle des expressions arithmétiques

Theorem (Confluence de la Sémantique computationelle)

$\forall e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ *et* $\exists e' \in Exp$ *et* $\exists n \in \mathbb{N}$
$e \rightsquigarrow e' \Leftrightarrow e' = n$

Démonstration.

- $\Longrightarrow$ contraposée
- $\Longleftarrow$ direct

$\square$

Validité et complétude de la sémantique computationelle des expressions arithmétiques

Theorem (Validité et complétude de la Sémantique computationelle)

$\forall e \in Exp = T_{\{+,-,*,/\}}(\mathbb{N})$ *et* $\exists n \in \mathbb{N}$
$e \Rightarrow n \Leftrightarrow e \rightsquigarrow n$

Démonstration.

- $\Longrightarrow$ par induction
- $\Longleftarrow$ par induction

$\square$

Reprise exercice : sémantique d'un véhicule programmable

Pour les programmes comme succession d'instruction nous avions :
La relation est étendue
$(Direct \times Coord) \times T_{\{\epsilon,\_::\_\}}(M) \times Direct \times Coord$
Il faut dans une sémantique par pas que :
$(Direct \times Coord) \times T_{\{\epsilon,\_::\_\}}(M) \times Direct \times Coord \times T_{\{\epsilon,\_::\_\}}(M)$

La règle était :
$$\frac{<d,p>,mov \Rightarrow <d',p'>, <d',p'>,prog \Rightarrow <d'',p''>}{<d,p>,mov::prog \Rightarrow <d'',p''>}$$

La règle devient :
$$\frac{<d,p>,mov \Rightarrow <d',p'>}{<d,p>,mov::prog \rightarrow <d',p'>,prog}$$

Reprise exercice : sémantique d'un véhicule programmable(2)

Definition (Fermeture transitive)

soit $\longrightarrow$ un système de transition la fermeture transitive $\longrightarrow^*$ est une relation satisfaisant les propriétés suivantes :

$$RCB : \frac{< d, p >, prog \longrightarrow < d', p' >, prog'}{< d, p >, prog \longrightarrow^* < d', p' >, prog'}$$

$$RCI : \frac{< d, p >, prog \longrightarrow^* < d'', p'' >, prog'' \longrightarrow < d', p' >, prog'}{< d, p >, prog \longrightarrow^* < d', p' >, prog'}$$

Theorem (La forme canonique a la propriété)

$$RCanon : \frac{< d, p >, prog \longrightarrow^* < d', p' > \epsilon}{< d, p >, prog \rightsquigarrow < d', p' >}$$

Sémantique concrète des expressions arithmétiques

Il s'agit de la sémantique classique d'un langage fournie par son compilateur ! Dans notre exemple nous idéaliserons le compilateur ainsi que la machine abstraite d'exécution. Le processus de calcul est donc decomposé en :

- La traduction des expressions en programmes de la machine abstraite : $\Rightarrow_{Trad} \subseteq Exp \times Prog$
- l'exécution du programme de la machine abstraite sur une pile : $\Rightarrow_{MA} \subseteq Stack \times Prog \times Stack$

Syntaxe concrète d'une machine abstraite

La machine abstraite pour notre exemple implémente une pile pour le calcul des expressions : Les instructions de la machine abstraite :

- $\mathit{Push} : \mathbb{N} \rightarrow \mathit{Instruction}$
- $\mathit{Pop} :\rightarrow \mathit{Instruction}$
- $\mathit{Apply}(+) :\rightarrow \mathit{Instruction}$
- $\mathit{Apply}(*) :\rightarrow \mathit{Instruction}$
- $\mathit{Apply}(-) :\rightarrow \mathit{Instruction}$
- $\mathit{Apply}(/) :\rightarrow \mathit{Instruction}$

Un programme est une suite d'instruction :

- $\_;\_ : \mathit{Prog}, \mathit{Instruction} \rightarrow \mathit{Prog}$
- $\mathit{noop} :\rightarrow \mathit{Prog}$

Sémantique opérationelle d'évaluation de la machine abstraite (1)

Il s'agit de donner un domaine sémantique à la fonction d'évaluation, ici on choisi une d'une pile d'entier engendrée par les opérations [] et :: (pile vide et concaténation) : $Stack = T_{\{[],\_::\_\}}(\mathbb{N})$

Definition (Sémantique machine abstraite (1))

$p \in Stack$ et $n, n' \in \mathbb{N}$

$$RPush : \frac{}{p \overset{Push(n)}{\Longrightarrow} p :: n} \qquad RPop : \frac{}{p :: n \overset{Pop(n)}{\Longrightarrow} p}$$

Sémantique opérationelle d'évaluation de la machine abstraite(2)

Definition (Sémantique machine abstraite (2))

$p \in Stack$ et $n, n' \in \mathbb{N}$ et $+_{\mathbb{N}}, *_{\mathbb{N}}, -_{\mathbb{N}}, /_{\mathbb{N}}$ sont les fonctions sur $\mathbb{N}$

$$RAp \frac{\forall op \in \{+, -, *, /\}}{p :: n :: n' \overset{Apply(op)}{\Longrightarrow} p :: n \; op_{\mathbb{N}} \; n'}$$

$$Rvide : \frac{}{p \overset{noop}{\Longrightarrow} p} \qquad RSeq : \frac{p \overset{prog}{\Longrightarrow} p', p' \overset{i}{\Longrightarrow} p''}{p \overset{prog;i}{\Longrightarrow} p''}$$

Exemple : programme machine

$$\overline{[] \stackrel{noop;Push(3);Push(2);Apply(+)}{\Longrightarrow}}$$

Sémantique par traduction des expressions arithmétiques

Une manière simple et efficace de fournir une sémantique consiste en la traduction de ce langage en un autre langage dont on connait un mécanisme d'évaluation.

Definition (Sémantique traduction)

$prog\,, prog' \in Prog$ et $n, n' \in \mathbb{N}$

$$RTMAn : \frac{}{n \Rightarrow_{Trad}\ noop; Push(n)}$$

$$\forall op \in \{+, -, *, /\}$$

$$RTMAProg : \frac{e \Rightarrow_{Trad}\ prog\,, e' \Rightarrow_{Trad}\ prog'}{e\ op\ e' \Rightarrow_{Trad}\ prog; prog'; Apply(op)}$$

Correction de la sémantique par traduction

Afin d'assurer la correction du processus il faut que :

Theorem

$\forall e, \exists n, \exists prog \ \ t.q.$

$e \Rightarrow n \Leftrightarrow e \Rightarrow_{Trad} prog \wedge [] \xrightarrow{prog} [] :: n$

(à prouver en exercice ! !)

Remarque : L'opération $\_;\_ : Prog, Instruction \rightarrow Prog$ doit être étendue vers $\_;\_ : Prog, Prog \rightarrow Prog$

Exercice