

Sémantique élémentaire des langages: sémantiques d'expressions arithmétiques

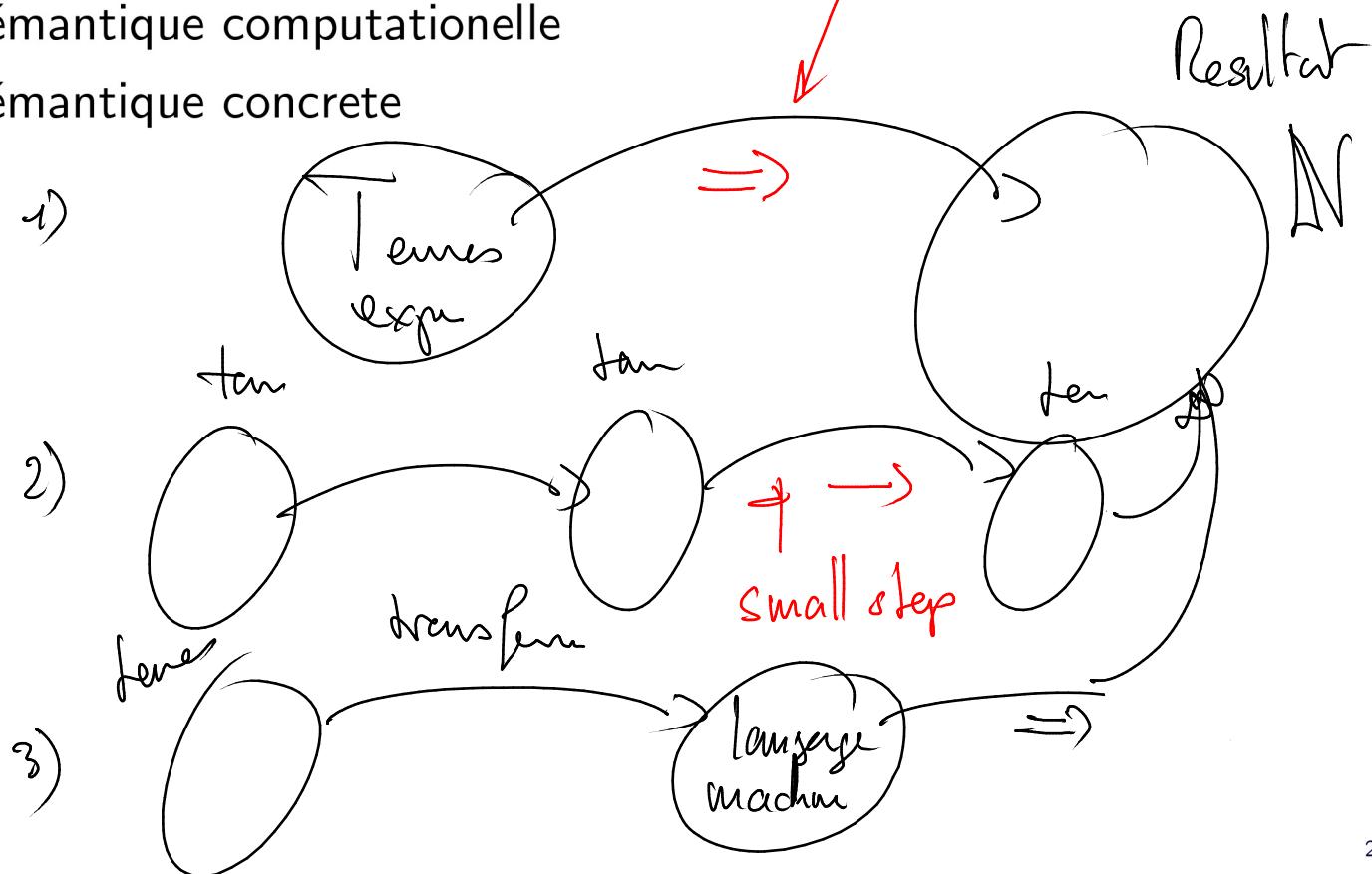
Didier Buchs

Université de Genève

5 mars 2018

Sémantique d'expression arithmétique

- 1) • Sémantique d'évaluation (dénotationnelle) *bigger*
 - 2) • Sémantique computationnelle
 - 3) • 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 :

$$(3 * 4) + 2 \in T_{\{+,-,*,/\}}(\mathbb{N})$$
$$3 * (4 + 2) \in T_{\{+,-,*,/\}}(\mathbb{N})^1$$

3/40

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 :
 $\text{eval} \subseteq \text{Exp} \times \mathbb{N}$
- Notation : $e \in \text{Exp} = T_{\{+, -, *, /\}}(\mathbb{N})$ et $n \in \mathbb{N}$ alors on note
 $e \Rightarrow n \Leftrightarrow (e, n) \in \text{eval}$

Definition (Sémantique d'évaluation)

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

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

Sémantique
des opérateurs

R Constante : $\frac{}{n \Rightarrow n}$

$$R+ : \frac{e \Rightarrow n, e' \Rightarrow n'}{e + e' \Rightarrow n +_{\mathbb{N}} n'}$$

$$R- : \frac{e \Rightarrow n, e' \Rightarrow n'}{e - e' \Rightarrow n -_{\mathbb{N}} n'}$$

Syntaxe

opérations que calcule

$$R* : \frac{e \Rightarrow n, e' \Rightarrow n'}{e * e' \Rightarrow n *_{\mathbb{N}} n'}$$

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

Evaluation d'une expression arithmétique

Example

Soit l'expression $3 * 4$ à évaluer

$$\frac{3 \Rightarrow 3, 4 \Rightarrow 4}{3 * 4 \Rightarrow 12}$$

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

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

Exercice : Evaluation d'une expression arithmétique

$$\begin{array}{c} \overline{3 \Rightarrow 3} \quad \overline{4 \Rightarrow 4} \\ \hline 3+4 \Rightarrow ? \end{array} \qquad \begin{array}{c} \overline{6 \Rightarrow 6} \quad \overline{1 \Rightarrow 1} \\ \hline 6+1 \Rightarrow ? \end{array}$$
$$\hline (3+4) * (6+1) \Rightarrow 49$$

7/40

Propriété de la sémantique d'évaluation des expressions arithmétiques

Theorem (Unicité de la Sémantique d'évaluation)

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

Démonstration.

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

$$\left. \begin{array}{l} n_1 + n_2 = n'_1 + n'_2 \\ n_1 = n'_1 \text{ et } n_2 = n'_2 \end{array} \right\} \begin{array}{l} \text{car} \\ + \\ \text{est} \\ \text{deterministe} \end{array}$$



8/40



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. $e \Rightarrow n$

Démonstration.

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

□



domaine de def

$$3/0 \quad ? \quad 3/0 = \perp$$

9/40

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 :

$$T_{\{\epsilon, _ :: _\}}(\{L, R, F\}) = T_{\{\epsilon, _ :: _\}}(M) = \text{PROG}$$

Par exemple : suivre un carré correspond au programme :

$square = F :: R :: F :: R :: F :: R :: F :: R :: \epsilon$

PROG \Rightarrow Demande sémantique
pos x direction

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.

$$\text{PROG} \times (\text{pes} \times \text{clived}) \supseteq \Rightarrow$$

Exercice : sémantique d'un véhicule programmable (suite)

Le domaine sémantique est composé de :

$position \in Direct \times Coord$ où

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

$$\langle d, p \rangle, M \Rightarrow \langle d'', p'' \rangle, \quad \langle d'', p'' \rangle \text{ PROG} \Rightarrow \langle d', p' \rangle$$

et

$$\overline{\langle d, p \rangle \Sigma \Rightarrow \langle d, p \rangle}$$

$$\overline{\langle d, p \rangle \text{ PROG::M} \Rightarrow \langle d', p' \rangle}$$

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

$$\overline{\langle d, p \rangle, L \Rightarrow \langle d \times \text{rot}_{-90}, p \rangle}$$

$$\overline{\langle d, p \rangle, R \Rightarrow \langle d \times \text{rot}_{90}, p \rangle}$$

$$\overline{\langle d, p \rangle, F \Rightarrow \langle d, p + d * c \rangle}$$

Exercice : sémantique d'un véhicule programmable(suite)

La relation de transition pour une instruction est donc :

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

Les règles pour les actions élémentaires :

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

Exercice : sémantique d'un véhicule programmable (suite)

Pour les programmes comme succession d'instruction :

La relation est étendue

$$(\text{Direct} \times \text{Coord}) \times T_{\{\epsilon, \cdot :: \cdot\}}(M) \times \text{Direct} \times \text{Coord}$$

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

$$\overbrace{< d, p >, \epsilon}^{\text{---}} \rightarrow < d, p >$$

$$\text{Square} := F :: R :: F :: R :: F :: R$$

$$\overbrace{< d', p' >, R}^{\text{---}} \rightarrow < d', p' >, < d', p' >, F, R, F, R, F, R, < d, p >$$

$$\frac{< d, p >, F \rightsquigarrow < d', p' >, < d', p' >, R, F, R, F, R, F, R}{< d, p >, \text{Sqr} \rightsquigarrow < d, p >, < d, p >}$$

Exercice : sémantique d'un véhicule programmable (suite)

Question : Prouver que $\forall p, d : \langle d, p \rangle, \text{square} \rightarrow \langle d, p \rangle$

les quatres rotations successives donnent :

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

4x la rotation

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

$a + b$

$a = 9'9$

$b = 1'8$

$c = 2'0'1'1'8$

Correction : Evaluation d'une expression arithmétique avec des chiffres

$$\begin{array}{c} \overline{0 \in Ch} \quad \overline{1 \in Ch} \quad \overline{2 \in Ch} \\ \overline{3 \in Ch} \quad \overline{4 \in Ch} \quad \overline{5 \in Ch} \\ \overline{6 \in Ch} \quad \overline{7 \in Ch} \quad \overline{8 \in Ch} \\ \overline{9 \in Ch} \quad \overline{n \in Ch} \quad \overline{n \in Ch, m \in Num} \\ \overline{n \in Num} \quad \overline{m' n \in Num} \end{array}$$

Il s'agit de construire la relation : $T_{\{+\}}(Num) \Rightarrow Num$

Table de calcul sur les chiffres !! $T_{\{+\}}(Ch) \Rightarrow Num$

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



$$\overbrace{n + 0 \Rightarrow n}^{\text{Exp} \implies Num}$$

Exp \implies Num

$c, d, e \in \text{Nur}$

$n, m, o \in \text{Chiffre}$

Sans retenu

1) $m+n \Rightarrow o, c+d \Rightarrow e$

$$c'm + d'n \Rightarrow e'o$$

Avec retenue

2) $m+n \Rightarrow q'o, c+d \Rightarrow e, e+\varepsilon' \Rightarrow f$

$$c'm + d'n \Rightarrow f'o$$

3) $d'n + \varepsilon \Rightarrow d'n$

4) $\varepsilon + d'n \Rightarrow d'n$

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)

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

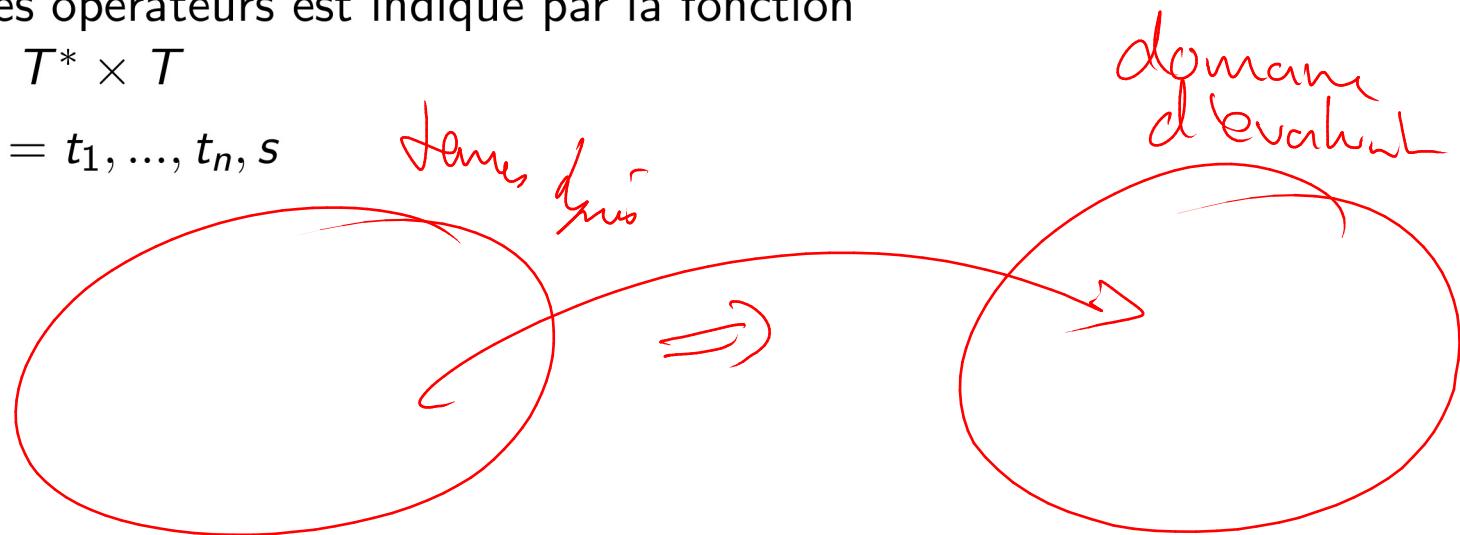
$$\frac{n + m \Rightarrow ret'k, k \in Ch, n \in Ch, d + ret \Rightarrow r}{n + d'm \Rightarrow r'k}$$

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. $\mu(op) = t_1, \dots, t_n, s$



20/40

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 $\mu(op) = t_1 t_2 \dots t_n$, t alors
 $op_t : Dom(t_1) \times Dom(t_2) \times \dots \times Dom(t_n) \rightarrow Dom(t)$

Nous allons donc déduire/construire la relation :

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

où :

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

Typage :règles

Definition (Sémantique d'évaluation typée)

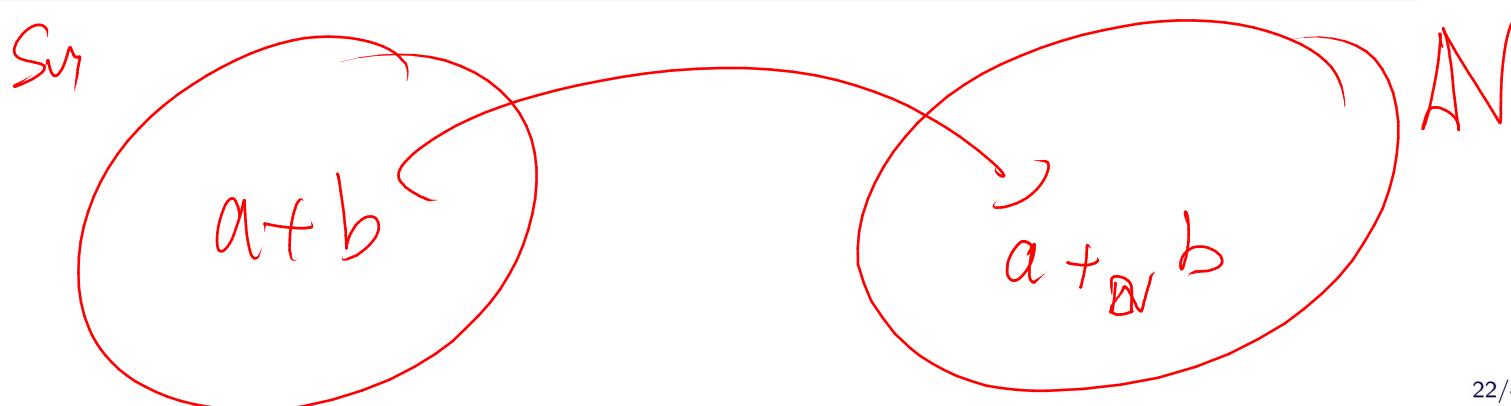
$$e \in Exp = T_{OP}(\emptyset)$$

op sont les fonctions sur les types

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

$$R_{op2} : \frac{op \in OP, \mu(op) = t_1 t_2 t, T \vdash e \implies n : t_1, T \vdash e' \implies n' : t_2}{T \vdash e \ op \ e' \implies n \ op_t n' : t}$$

Inference de type
Sémantique dynamique



22/40

Sémantique computationnelle des expressions arithmétiques

- Calcul de l'effet de chaque 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 computationnelle des expressions arithmétiques

- La relation d'un pas dévaluation détermine un système de transition : $comp \subseteq Exp \times Exp$
- Notation : $e \not\in Exp = T_{\{+, -, *, /\}}(\mathbb{N})$ et $n \in \mathbb{N}$ alors on note
 $e \rightarrow \underline{o} \Leftrightarrow (e, \underline{o}) \in comp$

Definition (Sémantique computationnelle)

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

$$RC+ : \frac{}{n + n' \rightarrow n +_{\mathbb{N}} n'}$$

$$RC- : \frac{}{n - n' \rightarrow n -_{\mathbb{N}} n'}$$

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

$$RCL : \frac{e \rightarrow e''}{e \ op \ e' \rightarrow e'' \ op \ e'}$$

$$RC* : \frac{}{n * n' \rightarrow n *_{\mathbb{N}} n'}$$

$$RC/ : \frac{}{n / n' \rightarrow n /_{\mathbb{N}} n'}$$

$$RCR : \frac{e' \rightarrow e''}{e \ op \ e' \rightarrow e \ op \ e''}$$

Calcul d'une expression arithmétique

Example

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

Solution :

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

Exercice : calcul d'une expression arithmétique

Stratégie droite

$$\begin{array}{c} (3+4) \rightarrow + \\ \hline (1+2) * (3+4) \rightarrow (1+2) * + \rightarrow 3 * + \rightarrow 21 \end{array}$$

$1+2 \rightarrow 3$

Stratégie ~~droite~~ gauche

$$\begin{array}{c} 1+3 \rightarrow 3 \\ \hline (1+2) * (3+4) \rightarrow 3 * (3+4) \rightarrow 3 * 7 \rightarrow 21 \end{array}$$

$3+4 \rightarrow +$

Non-déterminisme

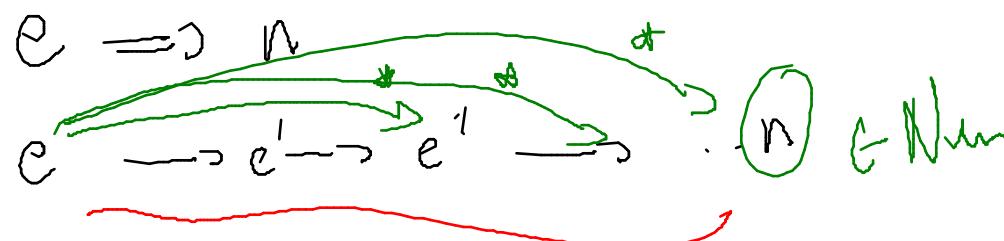
Constat : il y a différentes dérivation possibles, c'est le non déterminisme de l'application de RCL et RCR, il s'agit de voir si cela implique un résultat de calcul différent ? ?

Definition (Fermeture transitive)

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

$$RCB : \frac{e \rightarrow e'}{e \rightarrow^* e'}$$

$$RCI : \frac{e \rightarrow^* e'', e'' \rightarrow e'}{e \rightarrow^* e'}$$



27/40

Forme canonique

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 \rightarrow un système de transition la relation canonique \rightsquigarrow est une relation satisfaisant les propriétés suivantes :

$$R_{\text{Canon}} : \frac{e \xrightarrow{*} e'' \text{ et } e'' \not\rightsquigarrow e'}{e \rightsquigarrow e'}$$
$$R_{\text{Base}} : \frac{}{n \rightsquigarrow n}$$

Propriété de la sémantique computationnelle des expressions arithmétiques

Theorem (Confluence de la Sémantique computationnelle)

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

Démonstration.

- \implies contraposée
- \Leftarrow direct



Validité et complétude de la sémantique computationnelle des expressions arithmétiques

Theorem (Validité et complétude de la Sémantique computationnelle)

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

Démonstration.

- \Rightarrow par induction
- \Leftarrow par induction



Reprise exercice : sémantique d'un véhicule programmable

Pour les programmes comme succession d'instruction nous avions :

La relation est étendue

$$(\text{Direct} \times \text{Coord}) \times T_{\{\epsilon, \dots\}}(M) \times \text{Direct} \times \text{Coord}$$

sémantique d'évaluation

Il faut dans une sémantique par pas que :

$$(\text{Direct} \times \text{Coord}) \times T_{\{\epsilon, \dots\}}(M) \times \text{Direct} \times \text{Coord} \times T_{\{\epsilon, \dots\}}(M)$$

ce qui reste à évaluer

La règle était :

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

La règle devient :

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

$$\begin{array}{c} \exists (\forall x \ P(x)) \\ \} \times \exists P(x) \end{array}$$

31/40

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

Definition (Fermeture transitive)

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

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

Theorem (La forme canonique a la propriété)

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

programme
l'individualité

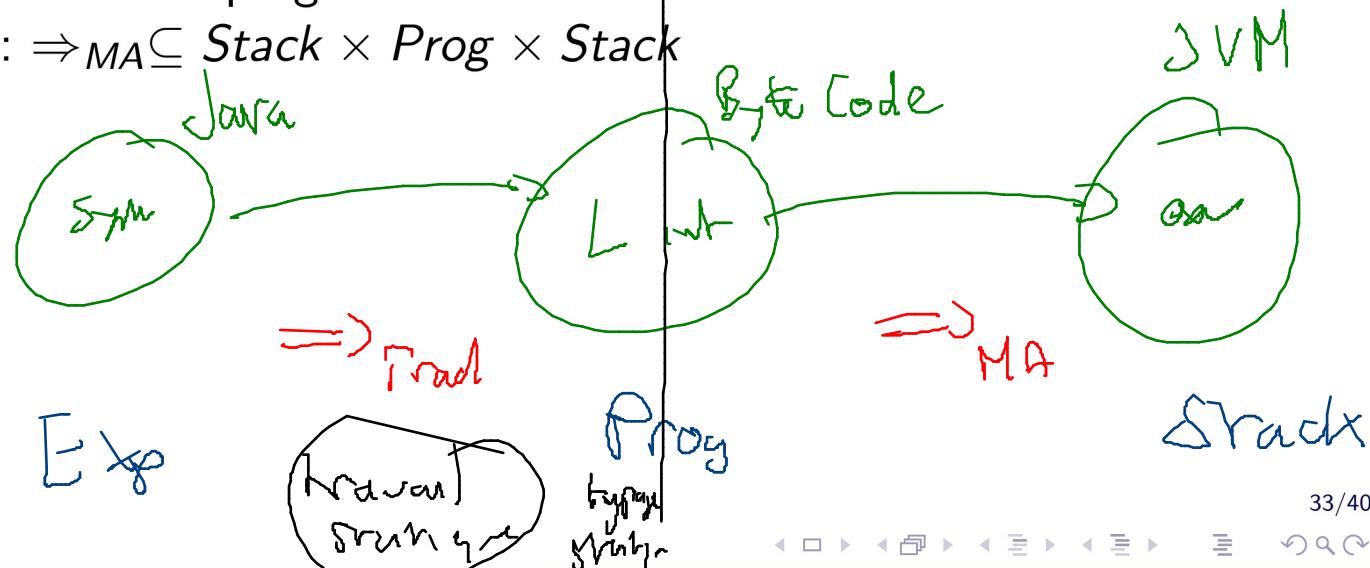
32/40



Sémantique concrète des expressions arithmétiques

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

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

Un programme est une suite d'instruction :

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

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

Il s'agit de donner un domaine sémantique à la fonction d'évaluation, ici on choisit 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 $n, n' \in \mathbb{N}$

$$RPush : \frac{}{p \xrightarrow{Push(n)} p :: n} \quad RPop : \frac{}{p :: n \xrightarrow{Pop(n)} p}$$

Sémantique opérationnelle 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}

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

$$RAp \xrightarrow{} p :: n :: n' \xrightarrow{\text{Apply}(op)} p :: n \underset{\mathbb{N}}{op} n'$$

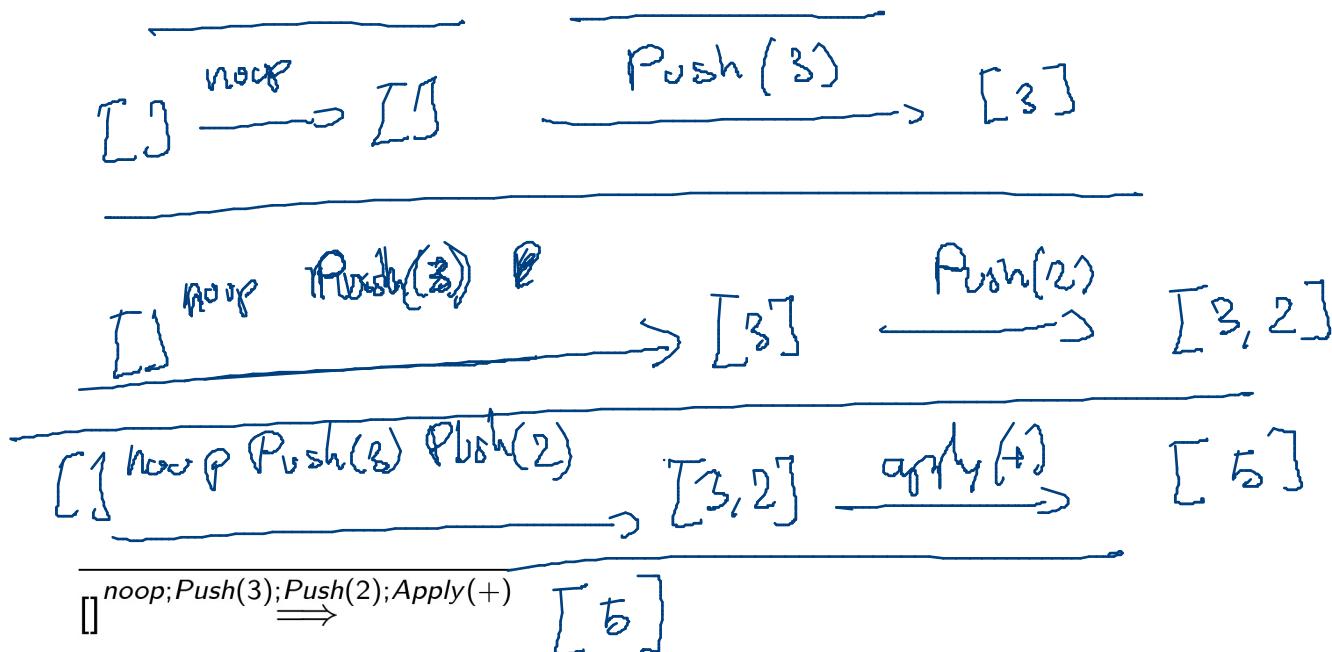
$$Rvide : \frac{}{p \xrightarrow{\text{noop}} p}$$

$$RSeq : \frac{p \xrightarrow{\text{prog}} p', p' \xrightarrow{i} p''}{p \xrightarrow{\text{prog}; i} p''}$$

en plus pour bytecode :
- affectation ?
- condition
- branche et

36/40

Exemple : programme machine



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 connaît un mécanisme d'évaluation.

Definition (Sémantique traduction)

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

$$RTMAProg : \frac{e \Rightarrow_{Trad} prog, e' \Rightarrow_{Trad} prog'}{e \ op \ e' \Rightarrow_{Trad} prog; prog'; Apply(op)}$$
$$RTMAAn : \frac{}{n \Rightarrow_{Trad} noop; Push(n)}$$
$$\forall op \in \{+, -, *, /\}$$

outils de métaprogrammation
classe / LVM
préservent la sémantique.
gca

Correction de la sémantique par traduction

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

Theorem

$$\forall e, \exists n, \exists \text{prog } t.q.$$

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

sémantique par
traduction

(\Leftrightarrow)

sémantique
d'evalution

(à prouver en exercice !!)

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

Exercice

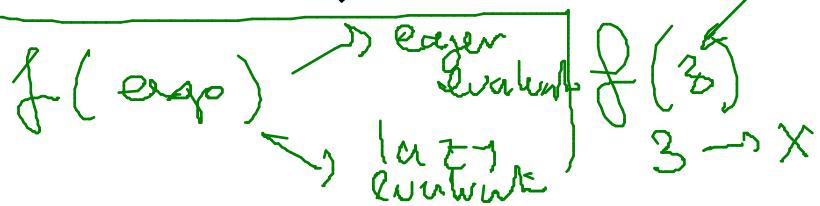
Suite :

Langage complet

- affectation
- conditions
- contrôle

"diviser pour régner"
modularisation

- fonctions procédures



(sémantique d'évaluation)

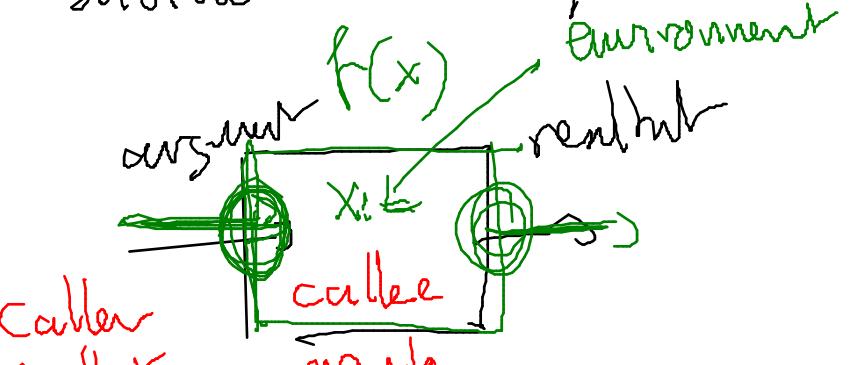
variables

(membre container)

if then else

while , repeat , for

"divide and conquer"



40/40