

Sémantique opérationnelle pour les expressions arithmétiques simples (EA).

Depuis le début du cours, on s'est intéressé à la *méthode inductive*. On essaie d'appliquer cette méthode à « l'exécution » des « programmes ».

On définira un programme comme un ensemble inductif : un programme est donc une structure de donnée. L'exécution d'un programme sera décrit comme des relations inductives (essentiellement binaires) sur les programmes. Définir ces relations, cela s'appelle la *sémantique opérationnelle*.

On considèrera deux sémantiques opérationnelles

- ▷ la sémantique à grands pas, où l'on associe un résultat à un programme ;
- ▷ la sémantique à petits pas, où l'on associe un programme « un peu plus tard » à un programme.

Notre objectif, dans un premier temps, est de définir OCaml, ou plutôt un plus petit langage fonctionnel inclus dans OCaml.

On se donne l'ensemble \mathbb{Z} (on le prend comme un postulat). On définit l'ensemble EA en Rocq par :

```
Inductive EA : Set :=  
  | Cst :  $\mathbb{Z} \rightarrow$  EA  
  | Add : EA  $\rightarrow$  EA  $\rightarrow$  EA.
```

Code 1 | Définition des expressions arithmétiques simples

Note 1. On se donne \mathbb{Z} et on note $k \in \mathbb{Z}$ (vu comme une métavariable). On définit (inductivement) l'ensemble EA des expressions arithmétiques, notées a, a', a_1, \dots par la grammaire

$$a ::= \underline{k} \mid a_1 \oplus a_2.$$

Exemple 1. L'expression $\underline{1} \oplus (\underline{3} \oplus \underline{7})$ représente l'expression Rocq

$$\text{Add}(\text{Cst } 1, \text{Add}(\text{Cst } 3)(\text{Cst } 7)),$$

que l'on peut représenter comme l'arbre de syntaxe...

Remarque 1. Dans le but de définir un langage minimal, il n'y a donc pas d'intérêt à ajouter \ominus et \otimes , représentant la soustraction et la multiplication.

1 Sémantique à grands pas sur EA.

On définit la sémantique opérationnelle à grands pas pour EA. L'intuition est d'associer l'exécution d'un programme avec le résultat. On définit la relation d'évaluation $\Downarrow \subseteq \text{EA} * \mathbb{Z}$, avec une notation infixe, définie par les règles d'inférences suivantes :

$$\frac{}{\underline{k} \Downarrow k} \quad \text{et} \quad \frac{a_1 \Downarrow k_1 \quad a_2 \Downarrow k_2}{a_1 \oplus a_2 \Downarrow k},$$

où, dans la seconde règle d'inférence, $k = k_1 + k_2$. Attention, le $+$ est la somme dans \mathbb{Z} , c'est une opération *externalisée*. Vu qu'on ne sait pas comment la somme a été définie dans \mathbb{Z} (on ne sait pas si elle est définie par induction/point fixe, ou pas du tout), on ne l'écrit pas dans la règle d'inférence.

La forme générale des règles d'inférences est la suivante :

$$\text{Cond. App.} \quad \frac{P_1 \quad \dots \quad P_m}{C} \mathcal{R}_i$$

– 2/9 –

où l'on donne les conditions d'application (ou *side condition* en anglais). Les P_1, \dots, P_m, C sont des relations inductives, mais les conditions d'applications **ne sont pas** forcément inductives.

Exemple 2.

$$\begin{array}{c} \begin{array}{c} \underline{3} \Downarrow 3 \\ 3 + 7 = 10 \end{array} \quad \begin{array}{c} \begin{array}{c} \underline{2} \Downarrow 2 \quad \underline{5} \Downarrow 5 \\ 2 + 5 = 7 \end{array} \\ \hline \underline{(2 \oplus 5)} \Downarrow 7 \end{array} \\ \hline \underline{3} \oplus (\underline{2} \oplus \underline{5}) \Downarrow 10 \end{array} .$$

2 Sémantique à petits pas sur EA.

On définit ensuite la sémantique opérationnelle à *petits pas* pour EA. L'intuition est de faire un pas exactement (la relation n'est donc pas réflexive) dans l'exécution d'un programme et, si possible, qu'elle soit déterministe.

Une relation *déterministe* (ou *fonctionnelle*) est une relation \mathcal{R} telle que, si $a \mathcal{R} b$ et $a \mathcal{R} c$ alors $b = c$.

La relation de réduction $\rightarrow \subseteq \text{EA} * \text{EA}$, notée infix, par les règles d'inférences suivantes

$$\begin{array}{c} \begin{array}{c} k = k_1 + k_2 \\ \hline \underline{k_1} \oplus \underline{k_2} \rightarrow \underline{k} \end{array} \mathcal{A}, \\ \hline \frac{a_2 \rightarrow a'_2}{a_1 \oplus a_2 \rightarrow a_1 \oplus a'_2} \mathcal{C}_d \quad \text{et} \quad \frac{a_1 \rightarrow a'_1}{a_1 \oplus \underline{k} \rightarrow a'_1 \oplus \underline{k}} \mathcal{C}_g. \end{array}$$

Il faut le comprendre par « quand c'est fini à droite, on passe à gauche ».

Les règles \mathcal{C}_g et \mathcal{C}_d sont nommées respectivement *règle contextuelle droite* et *règle contextuelle gauche*. Quand $a \rightarrow a'$, on dit que a se *réduit* à a' .

Remarque 2. La notation $k \nrightarrow$ indique que, quelle que soit l'expression $a \in \text{EA}$, on n'a pas $k \rightarrow a$. Les constantes ne peuvent pas être exécutées.

Exercice 1. Et si on ajoute la règle

$$\frac{a_1 \rightarrow a'_1 \quad a_2 \rightarrow a'_2}{a_1 \oplus a_2 \rightarrow a'_1 \oplus a'_2},$$

appelée *réduction parallèle*, que se passe-t-il ?

Remarque 3. Il n'est pas possible de démontrer $\underline{2} \oplus (\underline{3} \oplus \underline{4}) \rightarrow \underline{9}$. En effet, on réalise *deux* pas.

3 Coïncidence entre grands pas et petits pas.

On définit la clôture réflexive et transitive d'une relation binaire \mathcal{R} sur un ensemble E , notée \mathcal{R}^* . On la définit par les règles d'inférences suivantes :

$$\frac{}{x \mathcal{R}^* x} \quad \text{et} \quad \frac{x \mathcal{R} y \quad y \mathcal{R}^* z}{x \mathcal{R}^* z}.$$

Lemme 1. La relation \mathcal{R}^* est transitive.

Preuve. On démontre

$$\forall x, y \in E, \quad \text{si } x \mathcal{R}^* y \text{ alors } \underbrace{\forall z, y \mathcal{R}^* z \implies x \mathcal{R}^* z}_{\mathcal{P}(x,y)}$$

par induction sur $x \mathcal{R}^* y$. Il y a *deux* cas.

- ▷ *Réflexivité.* On a donc $x = y$ et, par hypothèse, $y \mathcal{R}^* z$.
- ▷ *Transitivité.* On sait que $x \mathcal{R} a$ et $a \mathcal{R}^* y$. De plus, on a

l'hypothèse d'induction

$$\mathcal{P}(a, y) : \forall z, y \mathcal{R}^* z \implies a \mathcal{R}^* z.$$

Montrons $\mathcal{P}(x, y)$. Soit z tel que $y \mathcal{R}^* z$. Il faut donc montrer $x \mathcal{R}^* z$. On sait que $x \mathcal{R} a$ et, par hypothèse d'induction, $a \mathcal{R}^* z$. Ceci nous donne $x \mathcal{R}^* z$ en appliquant la seconde règle d'inférence.

□

Lemme 2. Quelles que soient a_2 et a'_2 , si $a_2 \rightarrow^* a'_2$, alors pour tout a_1 , on a $a_1 \oplus a_2 \rightarrow^* a_1 \oplus a'_2$.

Preuve. On procède par induction sur $a_2 \rightarrow^* a'_2$. Il y a *deux* cas.

1. On a $a'_2 = a_2$. Il suffit donc de montrer que l'on a

$$a_1 \oplus a_2 \rightarrow^* a_1 \oplus a_2,$$

ce qui est vrai par réflexivité.

2. On sait que $a_2 \rightarrow a$ et $a \rightarrow^* a'_2$. On sait de plus que

$$\forall a_1, \quad a_1 \oplus a \rightarrow^* a_1 \oplus a'_2$$

par hypothèse d'induction. On veut montrer que

$$\forall a_1, \quad a_1 \oplus a_2 \rightarrow^* a_1 \oplus a'_2.$$

On se donne a_1 . On déduit de $a_2 \rightarrow a$ que $a_1 \oplus a_2 \rightarrow a_1 \oplus a$ par \mathcal{C}_d . Par hypothèse d'induction, on a $a_1 \oplus a \rightarrow^* a_1 \oplus a'_2$. Par la seconde règle d'inférence, on conclut.

□

Lemme 3. Quelles que soient les expressions a_1 et a'_1 , si $a_1 \rightarrow^* a'_1$ alors, pour tout k , $a_1 \oplus \underline{k} \rightarrow^* a'_1 \oplus \underline{k}$. □

Attention, le lemme précédent est faux si l'on remplace \underline{k} par une

expression a_2 . En effet, a_2 ne peut pas être « spectateur » du calcul de a_1 .

Proposition 1. Soient a une expression et k un entier. On a l'implication

$$a \Downarrow k \implies a \rightarrow^* \underline{k}.$$

Preuve. On le démontre par induction sur la relation $a \Downarrow k$. Il y a deux cas.

1. Dans le cas $a = \underline{k}$, alors on a bien $\underline{k} \rightarrow^* \underline{k}$.
2. On sait que $a_1 \Downarrow k_1$ et $a_2 \Downarrow k_2$, avec $k = k_1 + k_2$. On a également deux hypothèses d'induction :

$$\triangleright (H_1) : a_1 \rightarrow^* \underline{k_1};$$

$$\triangleright (H_2) : a_2 \rightarrow^* \underline{k_2}.$$

On veut montrer $a_1 \oplus a_2 \rightarrow^* \underline{k}$, ce que l'on peut faire par :

$$a_1 \oplus a_2 \xrightarrow{(H_2)+\text{lemme 2}}^* a_1 \oplus \underline{k_2} \xrightarrow{(H_1)+\text{lemme 3}}^* \underline{k_1} \oplus \underline{k_2} \xrightarrow{\text{sl}} \underline{k}.$$

□

Proposition 2. Soient a une expression et k un entier. On a l'implication

$$a \rightarrow^* \underline{k} \implies a \Downarrow k.$$

□

4 L'ensemble EA avec des erreurs à l'exécution.

On exécute des programmes de EA. On considère que $\underline{k_1} \oplus \underline{k_2}$ s'évalue comme

$$\frac{(k_1 + k_2) \times k_2}{k_2}.$$

Le cas $k_2 = 0$ est une situation d'erreur, une « **situation catastrophique** ». (C'est une convention : quand un ordinateur divise par

zéro, il explose !)

4.1 Relation à grands pas.

On note encore \Downarrow la relation d'évaluation sur $\mathbf{EA} * \mathbb{Z}_\perp$, où l'on définit l'ensemble $\mathbb{Z}_\perp = \mathbb{Z} \cup \{\perp\}$. Le symbole \perp est utilisé pour représenter un cas d'erreur.

Les règles d'inférences définissant \Downarrow sont :

$$\frac{}{\underline{k} \Downarrow k} \quad \begin{array}{c} k = k_1 + k_2 \\ k \neq 0 \end{array} \quad \frac{a_1 \Downarrow k_1 \quad a_2 \Downarrow k_2}{a_1 \oplus a_2 \Downarrow k} \quad \frac{a_1 \Downarrow k_1 \quad a_2 \Downarrow 0}{a_1 \oplus a_2 \Downarrow \perp},$$

et les règles de propagation du \perp :

$$\frac{a_1 \Downarrow \perp \quad (a_2 \Downarrow r)}{a_1 \oplus a_2 \Downarrow \perp} \quad \frac{(a_1 \Downarrow r) \quad a_2 \Downarrow \perp}{a_1 \oplus a_2 \Downarrow \perp}.$$

4.2 Relation à petits pas.

On (re)-définit la relation $\rightarrow \subseteq \mathbf{EA} * \mathbf{EA}_\perp$, où $\mathbf{EA}_\perp = \mathbf{EA} \cup \{\perp\}$, par les règles d'inférences

$$\begin{array}{c} k = k_1 + k_2 \\ k_2 \neq 0 \end{array} \quad \frac{}{\underline{k}_1 \oplus \underline{k}_2 \rightarrow \underline{k}} \quad \begin{array}{c} a_2 \rightarrow a'_2 \\ a_2 \neq \perp \end{array} \quad \frac{}{a_1 \oplus a_2 \rightarrow a_1 \oplus a'_2}$$

$$\begin{array}{c} a_1 \rightarrow a'_1 \\ a_1 \neq \perp \end{array} \quad \frac{}{a_1 \oplus \underline{k} \rightarrow a'_1 \oplus \underline{k}} \quad \frac{}{\underline{k}_1 \oplus \underline{0} \rightarrow \perp},$$

et les règles de propagation du \perp :

$$\frac{a_1 \rightarrow \perp}{a_1 \oplus \underline{k} \rightarrow \perp} \quad \text{et} \quad \frac{a_2 \rightarrow \perp}{a_1 \oplus a_2 \rightarrow \perp}.$$

Pour démontrer l'équivalence des relations grand pas et petits pas, ça semble un peu plus compliqué...

5 Sémantique contextuelle pour EA.

On définit la relation $\mapsto : \mathbf{EA} \times \mathbf{EA}$ par la règle :

$$k = k_1 + k_2 \quad \overline{E[k_1 \oplus k_2] \mapsto E[k]},$$

où E est un *contexte d'évaluation* que l'on peut définir par la grammaire

$$E ::= [] \mid \boxed{?}.$$

Le *trou* est une constante, notée $[]$ qui n'apparaît qu'une fois par contexte d'évaluation. Pour E un contexte d'évaluation et $a \in \mathbf{EA}$, alors $E[a]$ désigne l'expression arithmétique obtenue en remplaçant le trou par a dans E .

Exemple 3. On note $E_0 = \underline{3} \oplus ([] \oplus \underline{5})$ et $a_0 = \underline{1} \oplus \underline{2}$. Alors

$$\underline{3} \oplus ((\underline{1} \oplus \underline{2}) \oplus \underline{5}).$$

Que faut-il mettre à la place de $\boxed{?}$?

Exemple 4 (Première tentative). On pose

$$E ::= [] \mid \underline{k} \mid E_1 \oplus E_2.$$

Mais, ceci peut introduire *plusieurs* trous (voire aucun) dans un même contexte. C'est raté.

Exemple 5 (Seconde tentative). On pose

$$E ::= [] \mid a \oplus E \mid E \oplus a.$$

Mais, on pourra réduire une expression à droite avant de réduire à gauche. C'est encore raté.

Exemple 6 (Troisième (et dernière) tentative). On pose

$$E ::= [] \mid a \oplus E \mid E \oplus \underline{k}.$$

Là, c'est réussi !

Lemme 4. Pour toute expression arithmétique $a \in \mathbf{EA}$ qui n'est pas une constante, il existe un unique triplet (E, k_1, k_2) tel que

$$a = E[k_1 \oplus k_2].$$

Ceci permet de justifier la proposition suivante, notamment au niveau des notations.

Proposition 3. Pour tout a, a' , on a

$$a \rightarrow a' \quad \text{si, et seulement si,} \quad a \mapsto a'.$$

Preuve. Pour démontrer cela, on procède par double implication :

- ▷ « \implies » par induction sur $a \rightarrow a'$;
- ▷ « \impliedby » par induction sur E .

□