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 : ℤ → EA
|Add : EA → EA → 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 $a, a', a_1, \ldots$ 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

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

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 ajoute $\ominus$ 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 $\Downarrow \subseteq \mathsf{EA} * \mathbb{Z}$, avec une notation infixe, définie par les règles d'inférences suivantes :

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

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 :

$$\text{Cond. App.} \ \frac{P_1 \qquad \ldots \qquad P_m}{C} \ \mathcal{R}_i$$

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

Exemple 2.

$$\dfrac{\underline{3} \Downarrow 3 \qquad \dfrac{\underline{2} \Downarrow 2 \qquad \underline{5} \Downarrow 5}{(\underline{2} \oplus \underline{5}) \Downarrow 7}\ {\scriptstyle 2+5=7}}{\underline{3} \oplus (\underline{2} \oplus \underline{5}) \Downarrow 10}\ {\scriptstyle 3+7=10}.$$

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 $a \mathrel{\mathcal{R}} b$ et $a \mathrel{\mathcal{R}} c$ alors $b = c$.

La relation de réduction $\to \subseteq \mathsf{EA} * \mathsf{EA}$, notée infixe, par les règles d'inférences suivantes

$$\dfrac{}{\underline{k_1} \oplus \underline{k_2} \to \underline{k}}\ \mathscr{A}\ {\scriptstyle k = k_1 + k_2},$$

$$\dfrac{a_2 \to a_2'}{a_1 \oplus a_2 \to a_1 \oplus a_2'}\ \mathscr{C}_{\mathrm{d}} \quad \text{et} \quad \dfrac{a_1 \to a_1'}{a_1 \oplus \underline{k} \to a_1' \oplus \underline{k}}\ \mathscr{C}_{\mathrm{g}}.$$

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

Les règles $\mathscr{C}_{\mathrm{g}}$ et $\mathscr{C}_{\mathrm{d}}$ sont nommées respectivement *règle contextuelle droite* et *règle contextuelle gauche*. Quand $a \to a'$, on dit que a se *réduit* à a'.

Remarque 2. La notation $\underline{k} \not\to$ indique que, quelle que soit l'expression $a \in \mathsf{EA}$,on n'a pas $\underline{k} \to a$. Les constantes ne peuvent pas être exécutées.

Exercice 1. Et si on ajoute la règle

$$\frac{a_1 \to a_1' \qquad a_2 \to a_2'}{a_1 \oplus a_2 \to a_1' \oplus a_2'}\ ,$$

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

Remarque 3. Il n'est pas possible de démontrer $\underline{2} \oplus (\underline{3} \oplus \underline{4}) \to \underline{9}$. 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 $\mathscr{R}$ sur un ensemble E, notée $\mathscr{R}^\star$. On la définit par les règles d'inférences suivantes :

$$\frac{}{x \mathrel{\mathscr{R}^\star} x} \quad \text{et} \quad \frac{x \mathrel{\mathscr{R}} y \qquad y \mathrel{\mathscr{R}^\star} z}{x \mathrel{\mathscr{R}^\star} z}\ .$$

Lemme 1. La relation $\mathscr{R}^\star$ est transitive.

Preuve. On démontre

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

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

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

l'hypothèse d'induction

$$\mathscr{P}(a, y) : \forall z, y \mathrel{\mathscr{R}^\star} z \implies a \mathrel{\mathscr{R}^\star} z.$$

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

□

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

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

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

$$a_1 \oplus a_2 \to^\star a_1 \oplus a_2,$$

ce qui est vrai par réflexivité.

2. On sait que $a_2 \to a$ et $a \to^\star a_2'$. On sait de plus que

$$\forall a_1, \quad a_1 \oplus a \to^\star a_1 \oplus a_2'$$

par hypothèse d'induction. On veut montrer que

$$\forall a_1, \quad a_1 \oplus a_2 \to^\star a_1 \oplus a_2'.$$

On se donne a_1. On déduit de $a_2 \to a$ que $a_1 \oplus a_2 \to a_1 \oplus a$ par $\mathscr{C}_\mathrm{d}$. Par hypothèse d'induction, on a $a_1 \oplus a \to^\star a_1 \oplus a_2'$. Par la seconde règle d'inférence, on conclut.

□

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

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
$$a \Downarrow k \implies a \to^\star \underline{k}.$$

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 $\underline{k} \to^\star \underline{k}$.
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 :
   - ▷ (H_1) : $a_1 \to^\star \underline{k}_1$;
   - ▷ (H_2) : $a_2 \to^\star \underline{k}_2$.

   On veut montrer $a_1 \oplus a_2 \to^\star \underline{k}$, ce que l'on peut faire par :
$$a_1 \oplus a_2 \xrightarrow{(H_2)+\text{lemme } 2}{}^\star a_1 \oplus \underline{k}_2 \xrightarrow{(H_1)+\text{lemme } 3}{}^\star \underline{k}_1 \oplus \underline{k}_2 \xrightarrow{\mathcal{A}} \underline{k}.$$

□

Proposition 2. Soient a une expression et k un entier. On a l'implication
$$a \to^\star \underline{k} \implies a \Downarrow k.$$

□

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 $\mathsf{EA} * \mathbb{Z}_\perp$, 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 :

$$\frac{}{\underline{k} \Downarrow k} \qquad {\scriptstyle k = k_1 + k_2 \atop k \neq 0}\ \frac{a_1 \Downarrow k_1 \qquad a_2 \Downarrow k_2}{a_1 \oplus a_2 \Downarrow k} \qquad \frac{a_1 \Downarrow k_1 \qquad a_2 \Downarrow 0}{a_1 \oplus a_2 \Downarrow \perp},$$

et les règles de propagation du $\perp$:

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

4.2 Relation à petits pas.

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

$${\scriptstyle k = k_1 + k_2 \atop k_2 \neq 0}\ \frac{}{\underline{k}_1 \oplus \underline{k}_2 \to \underline{k}} \qquad {\scriptstyle a_2 \neq \perp}\ \frac{a_2 \to a_2'}{a_1 \oplus a_2 \to a_1 \oplus a_2'}$$

$${\scriptstyle a_1 \neq \perp}\ \frac{a_1 \to a_1'}{a_1 \oplus \underline{k} \to a_1' \oplus \underline{k}} \qquad \frac{}{\underline{k}_1 \oplus \underline{0} \to \perp},$$

et les règles de propagation du $\perp$:

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

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 $\mapsto$: $\mathsf{EA} \times \mathsf{EA}$ par la règle :

$$\frac{}{E[\underline{k_1} \oplus \underline{k_2}] \mapsto E[\underline{k}]}\; k = k_1 + k_2,$$

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 $a \in \mathsf{EA}$, 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 $a \in \mathsf{EA}$ qui n'est pas une constante, il existe un unique triplet (E, k_1, k_2) tel que

$$a = E[\underline{k}_1 \oplus \underline{k}_2].$$

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

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

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

Preuve. Pour démontrer cela, on procède par double implication :
- ▷ « $\implies$ » par induction sur $a \to a'$;
- ▷ « $\impliedby$ » par induction sur E.

□