Chapitre 4

Introduction à la syntaxe et à la sémantique des langages de programmation

Plan

1. Introduction

2. Langage des expressions arithmétiques

3. Langage de programmation simple

Lectures conseillées :

- Notes de cours de Guy McCusker et Matthew Hennessy, University of Sussex. `http://www.informatics.sussex.ac.uk/courses/semantics/current/GuysNotes.pdf`
- David A. Schmidt, Programming Language Semantics. `http://people.cis.ksu.edu/~schmidt/705s12/Lectures/chapter.pdf`

Syntaxe et sémantique

La **syntaxe** d'un langage de programmation détermine ce qui constitue un programme valide du point de vue de la **forme**/du texte.

Quelles séquences de caractères constituent des programmes ?

La **sémantique** définit la signification d'un programme, c'est-à-dire ce qu'il calcule, comment il le calcule.

Que signifie ce programme ? Quand est-ce que deux programmes sont équivalents ? Est-ce que ce programme satisfait aux spécifications ?

Dans ce chapitre, on se focalisera sur la sémantique de langages de programmation très simples.

Sémantique formelle

Pourquoi définir la sémantique de manière formelle (plutôt que de manière informelle) ?

- Implémentation : correction des compilateurs, optimisation, analyse statique, etc.
- Vérification : permet de raisonner sur les programmes et leurs propriétés, que ce soit de manière automatique ou à la main.
- Design de langage : permet de détecter des ambiguités et d'organiser les choses de manière plus claire.

Plan

Avant de s'attaquer à un langage de programmation, on va se servir des expression arithmétiques (sans variable) pour définir les principaux concepts.

Expressions arithmétiques sur $\mathbb{N}$: syntaxe

L'ensemble des expressions *Exp* est défini récursivement comme suit :

- Cas de base : $\forall n \in \mathbb{N}$, $\mathtt{n} \in \mathit{Exp}$
- Cas récursifs : Si E_1 et $E_2 \in \mathit{Exp}$:
  - $(E_1 + E_2) \in \mathit{Exp}$
  - $(E_1 \times E_2) \in \mathit{Exp}$

On simplifie cette définition par une expression du type :

$$E \in \mathit{Exp} ::= \mathtt{n} | (E + E) | (E \times E)$$

où :

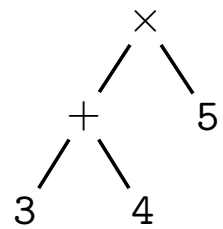
- "|" signifie "ou".
- $E + E$ signifie "une expression suivie de '+' suivi d'une expression"
- $\mathtt{n}$, '+', '$\times$' sont des *symboles terminaux* (cas de base). E est un *symbole non terminal* (cas récursif).

Cette notation s'appelle la *forme de Backus-Naur* (BNF)

Syntaxe concrète versus syntaxe abstraite

On distingue deux types de syntaxe :

- Syntaxe concrète : programme représenté par la suite de caractères tapés par l'utilisateur
  - Exemple : $3 + 4 \times 5$
- Syntaxe abstraite : programme représenté par une structure basée sur la définition récursive, typiquement un arbre
  - Exemple :



Passer de la syntaxe concrète à la syntaxe abstraite s'appelle *l'analyse syntaxique* du programme.

Dans ce chapitre, on supposera toujours que la syntaxe abstraite est connue (pas d'ambiguïté sur la manière dont l'expression est obtenue).

Sémantique

Essentiellement deux types de sémantiques :

- **Sémantique dénotationnelle :** la signification d'un programme est déterminée par une fonction, la *dénotation*, associant une valeur à un programme.

- **Sémantique opérationnelle :** la signification d'un programme est déterminée par le suite des *états* de la machine qui l'exécute.

Sémantique dénotationelle

On définit une fonction, appelée *dénotation*, prenant en argument un programme (et d'autres valeurs potentielles) et renvoyant la valeur résultant de l'exécution du programme.

Seule importe la valeur finale résultant de l'exécution et pas la manière de l'obtenir. Les deux programmes suivants sont identiques du point de vue de la sémantique dénotationnelle :

```
x = 0;
while x < 10
    x=x+1 ;
```

```
x = 10;
```

Le domaine des valeurs résultats de l'exécution du programme est appelé le *domaine sémantique* du programme.

Exemple : expressions arithmétiques

Le domaine sémantique est $\mathbb{N}$.

La dénotation est une fonction eval : $\mathit{Exp} \longrightarrow \mathbb{N}$ définie récursivement :

- Cas de base : $\mathrm{eval}(\mathtt{n}) = n$ où n est le naturel associé au numéral $\mathtt{n}$.
- Cas récursifs :
  - $\mathrm{eval}(E_1 + E_2) ::= \mathrm{eval}(E_1) + \mathrm{eval}(E_2)$
  - $\mathrm{eval}(E_1 \times E_2) ::= \mathrm{eval}(E_1) \cdot \mathrm{eval}(E_2)$

Exemple :

$$\begin{aligned} \mathrm{eval}((1 + (2 + 3))) &= \mathrm{eval}(1) + \mathrm{eval}(2 + 3) \\ &= 1 + \mathrm{eval}(2 + 3) \\ &= 1 + (2 + 3) \\ &= 6 \end{aligned}$$

Sémantique opérationnelle

La signification d'un programme est définie par une modélisation des étapes de son exécution.

Les programmes suivants sont très différents d'une point de vue sémantique opérationnelle :

```
x = 0;
while x < 10
    x=x+1 ;
```

```
x = 10;
```

L'exécution d'un programme peut être modélisée par une *machine d'état*.

Expressions arithmétiques

Idée générale : chaque étape de l'évaluation d'une expression consiste à simplifier le terme le plus à gauche (par convention). Exemple :

$$((1+2)+(4+5)) \rightarrow (3+(4+5)) \rightarrow (3+9) \rightarrow 12$$

Une machine d'état pour l'évaluation des expressions arithmétiques (addition uniquement) :

- $\mathcal{Q} = \mathit{Exp}$, l'ensemble des expressions
- $\mathcal{Q}_0 = \{E\}$, où $E \in \mathit{Exp}$ est l'expression à évaluer.
- δ est défini récursivement :
  - Cas de base : $\forall n_1, n_2 \in \mathbb{N} : (\mathtt{n}_1 + \mathtt{n}_2) \rightarrow \mathtt{n}_3 \in \delta$, où $n_3 = n_1 + n_2$
  - Cas récursif :
    - Si $E_2 \in \mathit{Exp}$ et $E_1 \rightarrow E_1' \in \delta$, alors $(E_1 + E_2) \rightarrow (E_1' + E_2) \in \delta$.
    - Si $E \rightarrow E' \in \delta$, alors $(\mathtt{n} + E) \rightarrow (\mathtt{n} + E') \in \delta$.

(Exercice : ajoutez la multiplication et simplifiez le terme à droite d'abord)

Règles déductives

Les transitions sont généralement définies sous la forme de régles déductives.

$$\frac{E_1 \to E_1'}{(E_1 + E_2) \to (E_1' + E_2)} \text{ (S-LEFT)}$$

$$\frac{E \to E'}{(\mathtt{n} + E) \to (\mathtt{n} + E')} \text{ (S-RIGHT)}$$

$$\frac{}{(\mathtt{n}_1 + \mathtt{n}_2) \to \mathtt{n}_3} \text{ (S-ADD)}$$

$$(\text{où } n_3 = n_1 + n_2)$$

Par exemple, la règle (S-LEFT) signifie que pour évaluer une addition, il faut d'abord évaluer le premier terme.

Exemple

Evaluation de $(3 + (2 + 1))$:

- Par l'axiome (S-ADD), on a $(2 + 1) \longrightarrow 3$.
- Par (S-RIGHT), $(3 + (2 + 1)) \longrightarrow (3 + 3)$
- Par (S-ADD), on a $(3 + 3) \longrightarrow 6$.
- Finalement :

$$(3 + (2 + 1)) \longrightarrow (3 + 3) \longrightarrow 6$$

La sémantique fixe l'ordre d'évaluation. On a :

$$((1 + 2) + (3 + 4)) \longrightarrow (3 + (3 + 4))$$

et pas

$$((1 + 2) + (3 + 4)) \longrightarrow ((1 + 2) + 7)$$

(Exercice : Ecrivez tous les $E \in Exp$ tels que $E \longrightarrow ((1 + 2) + 7)$)

Notations

Introduisons quelques notations.

Définition : On a $E \rightarrow^* E'$ si et seulement si soit $E = E'$, soit il existe une séquence finie :

$$E \rightarrow E_1 \rightarrow E_2 \rightarrow \ldots E_k \rightarrow E'.$$

($\Leftrightarrow E'$ est atteignable à partir de E).

Définition : La relation $\rightarrow^k$ pour tout $k \in \mathbb{N}$ est définie récursivement sur $\mathbb{N}$:

- Cas de base : $E \rightarrow^0 E$ pour tout $E \in Exp$
- Cas récursif : $E \rightarrow^{k+1} E'$ s'il existe E'' tel que :
  - $E \rightarrow^k E''$ et
  - $E'' \rightarrow E'$.

Définition : Soit une expression $E \in Exp$. n est la réponse finale de E si $E \rightarrow^* \mathtt{n}$.

Déterminisme et normalisation

Propriété de la sémantique dénotationnelle :

- Toute expression possède une réponse finale *(normalisation)*.
- Cette réponse est unique et ne peut être évaluée que d'une et une seule manière *(déterminisme)*.

Formellement : pour tout $E \in Exp$:

- Si $E \to E_1$ et $E \to E_2$, alors $E_1 = E_2$. (*déterminisme*)
- Si $E \to^* \mathtt{n}$ et $E \to^* n'$, alors $n = n'$. (*déterminisme*)
- Il existe n tel que $E \to^* \mathtt{n}$. (*normalisation*).

Déterminisme

Théorème : Si $E \to E_1$ et $E \to E_2$, alors $E_1 = E_2$.

Démonstation : Prouvons par induction structurelle le prédicat $P(E) =$"Pour tout F_1, F_2, si $E \to F_1$ et $E \to F_2$, alors $F_1 = F_2$".
Cas de base : $P(\mathtt{n})$ est vrai trivialement puisqu'il n'existe aucune transition à partie d'un numéral.
Cas inductif : $E = (E_1 + E_2)$. Supposons $P(E_1)$ et $P(E_2)$ vrais et montrons que $P(E_1 + E_2)$ est vrai.
Supposons que $E_1 + E_2 \to F_1$ et $E_1 + E_2 \to F_2$. On doit montrer que $F_1 = F_2$.
Deux cas possibles :

- E_1 est un numéral $\mathtt{n}_1$. Il y a alors deux façons de dériver $\mathtt{n}_1 + E_2 \to F_1$:
  - En appliquant la règle :
    $$\frac{E_2 \to G_1}{(\mathtt{n}_1 + E_2) \to (\mathtt{n}_1 + G_1)} \ \texttt{(S-RIGHT)}$$
    Dans ce cas $F_1 = \mathtt{n}_1 + G_1$, pour un G_1 tel que $E_2 \to G_1$. Puisqu'il y a une dérivation $E_2 \to G_1$, E_2 n'est pas un numéral. Toute dérivation à partir de E_2 doit donc utiliser `(S-RIGHT)`. On peut donc montrer similairement que $F_2 = \mathtt{n}_1 + G_2$, pour un G_2 tel que $E_2 \to G_2$.
    Or par hypothèse inductive, on a $G_1 = G_2$ et donc $F_1 = F_2$.

- En appliquant la règle (S-ADD). Dans ce cas, E_2 est un numéral $\mathtt{n}_2$ et donc F_1 est aussi un numéral $\mathtt{n}_3$, avec $n_1 + n_2 = n_3$. Considérons maintenant la dérivation $n_1 + E_2 \to F_2$. Puisque $E_2 = \mathtt{n}_2$, la seule règle possible est (S-ADD) et donc, $F_2 = \mathtt{n}_3$, ce qui montre $F_1 = F_2$.

- E_1 n'est pas un numéral. La seule règle possible est (S-LEFT) qui donne un règle de la forme :

$$\frac{E_1 \to G_1}{(E_1 + E_2) \to (G_1 + E_2)} \text{ (S-LEFT)}$$

F_1 est donc de la forme $G_1 + E_1$ pour un G_1 tel que $E_1 \to G_1$. De même, F_2 doit aussi être de la forme $G_2 + E_1$ pour un G_2 tel que $E_1 \to G_2$.

On utilisant $P(E_1)$, on obtient $G_1 = G_2$ et donc $F_1 = F_2$.

□

Déterminisme

Corrolaire : Pour tout naturel k et expression $E \in Exp$, si $E \rightarrow^k E_1$ et $E \rightarrow^k E_2$, alors $E_1 = E_2$.

Démonstation : La démonstration fonctionne par induction sur $\mathbb{N}$.
Soit le prédicat $P(k) =$" $E \rightarrow^k E_1$ et $E \rightarrow^k E_2$ implique $E_1 = E_2$".

Cas de base : Supposons que $E \rightarrow^0 E_1$ et $E \rightarrow^0 E_2$. On a trivialement $E_1 = E_2 = E$.

Cas inductif : Supposons $P(k)$ vrai et montrons $P(k+1)$. Par définition de $\rightarrow^{k+1}$, il existe E_1' et E_2' tels que

$$E \rightarrow^k E_1' \rightarrow E_1$$

$$E \rightarrow^k E_2' \rightarrow E_2$$

Par hypothèse inductive, on a $E_1' = E_2'$. Par le théorème précédent (déterminisme de $\rightarrow$), on en déduit que $E_1 = E_2$.

Déterminisme

Théorème : Si $E \rightarrow^* \mathtt{n}$ et $E \rightarrow^* \mathtt{n}'$, alors $\mathtt{n} = \mathtt{n}'$.

Démonstration : $E \rightarrow^* \mathtt{n}$ signifie qu'il existe $k \in \mathbb{N}$ tel que $E \rightarrow^k \mathtt{n}$. De même, il existe $k' \in \mathbb{N}$ tel que $E \rightarrow^{k'} \mathtt{n}'$.
Soit $k \leq k'$, soit $k \geq k'$. Supposons $k \leq k'$ (le cas $k \geq k'$ est symétrique). La dérivation est de la forme :

$$E \rightarrow^k \mathtt{n}$$
$$E \rightarrow^k E' \rightarrow^{(k'-k)} \mathtt{n}'$$

pour un E'. Par le corrolaire précédent, E' doit être égal à $\mathtt{n}$. Selon les règles du transparent 160, il n'y a pas de transition à partir d'un numéral. La réduction $E' \rightarrow^{(k'-k)} \mathtt{n}'$ doit donc être $\mathtt{n} \rightarrow^0 \mathtt{n}'$. Et donc $\mathtt{n} = \mathtt{n}'$.

□

Normalisation

Théorème : Pour tout E, il existe un n tel que $E \rightarrow^* n$.

Démonstation : La démonstration fonctionne par induction structurelle sur E.
Cas de base : $E = \mathtt{n}$. On a trivialement $\mathtt{n} \rightarrow^* \mathtt{n}$.

Cas inductif : $E = (E_1 + E_2)$. Par hypothèse inductive, il existe n_1 et n_2 tels que $E_1 \rightarrow^* \mathtt{n}_1$ et $E_2 \rightarrow^* \mathtt{n}_2$. Pour chaque étape de la dérivation :

$$E_1 \rightarrow E_1' \rightarrow E_1'' \ldots \rightarrow \mathtt{n}_1,$$

appliquer la régle de réduction de l'argument de gauche donne :

$$(E_1 + E_2) \rightarrow (E_1' + E_2) \rightarrow (E_1'' + E_2) \ldots \rightarrow (\mathtt{n}_1 + E_2).$$

En appliquant ensuite la règle de réduction de l'argument de droite, on peut déduire :

$$(\mathtt{n}_1 + E_2) \rightarrow^* (\mathtt{n}_1 + \mathtt{n}_2) \rightarrow \mathtt{n}_3,$$

où $n_3 = n_1 + n_2$. D'où $(E_1 + E_2) \rightarrow^* \mathtt{n}_3$.

□

Déterminisme et normalisation

Corrolaire : Pour tout expression E, il y a exactement un n tel que $E \rightarrow^* \mathtt{n}$.

Démonstration : C'est une conséquence directe des théorèmes précédents. □

Correspondance

On peut montrer que les deux sémantiques sont équivalentes.

Théorème : Pour tout expression E, $\text{eval}(E) = n$ si et seulement si $E \rightarrow^* \mathtt{n}$.

Démonstration : Montrons par induction structurelle sur E que $P(E)$="$\text{eval}(E) = n \Leftrightarrow E \rightarrow^* \mathtt{n}$" est vrai pour tout E.

Cas de base : E est un numéral $\mathtt{n}$. Dans ce cas, on a clairement $\text{eval}(E) = n$ et $E \rightarrow^* \mathtt{n}$ et donc $P(E)$ est vérifié.

Cas inductif : $E = (E_1 + E_2)$. Supposons que $P(E_1)$ et $P(E_2)$ soient vérifiés et montrons que $P(E)$ est vérifié.

Montrons d'abord que $(E_1 + E_2) \rightarrow^* \mathtt{n}_3 \Rightarrow \text{eval}((E_1 + E_2)) = n_3$.

> Si $(E_1 + E_2) \rightarrow^* \mathtt{n}_3$, il existe $\mathtt{n}_1$ et $\mathtt{n}_2$ tels que : $(E_1 + E_2) \rightarrow^* (\mathtt{n}_1 + E_2) \rightarrow^* (\mathtt{n}_1 + \mathtt{n}_2) \rightarrow \mathtt{n}_3$ et $n_1 + n_2 = n_3$. Par hypothèse inductive, on a $\text{eval}(E_1) = n_1$ et $\text{eval}(E_2) = n_2$. Par définition de eval, on a donc $\text{eval}((E_1 + E_2)) = \text{eval}(E_1) + \text{eval}(E_2) = n_1 + n_2 = n_3$.

Montrons que $\text{eval}((E_1 + E_2)) = n_3 \Rightarrow (E_1 + E_2) \rightarrow^* \mathtt{n}_3$:

Soit n_1 et n_2 tels que $\text{eval}(E_1) = n_1$ et $\text{eval}(E_2) = n_2$. Par définition de eval, on a $\text{eval}((E_1 + E_2)) = n_1 + n_2 = n_3$. Par hypothèse inductive, on a $E_1 \rightarrow^* \mathtt{n}_1$ et $E_2 \rightarrow^* \mathtt{n}_2$ et donc $(E_1 + E_2) \rightarrow^* (\mathtt{n}_1 + E_2) \rightarrow^* (\mathtt{n}_1 + \mathtt{n}_2) \rightarrow \mathtt{n}_3$, qui implique $(E_1 + E_2) \rightarrow^* \mathtt{n}_3$.

Résumé : expression arithmétique

Jusqu'ici, nous avons :

- défini la syntaxe du langage des expressions arithmétiques par une forme BNF,
- défini sa sémantique dénotationnelle par une dénotation définie récursivement sur la syntaxe,
- défini sa sémantique opérationnelle par une machine d'état,
- prouvé le déterminisme et la normalisation de la sémantique opérationnelle,
- prouvé la correspondance entre les sémantiques dénotationnelle et opérationnelle.