Leçon 930 – Sémantique des langages de programmation. Exemples.

9 février 2019

1 Extraits du Rapport

Rapport de jury 2017

L'objectif est de formaliser ce qu'est un programme : introduction des sémantiques opérationnelle et dénotationnelle, dans le but de pouvoir faire des preuves de programmes, des preuves d'équivalence, des preuves de correction de traduction. Ces notions sont typiquement introduites sur un langage de programmation (impératif) jouet. On peut tout à fait se limiter à un langage qui ne nécessite pas l'introduction des CPOs et des théorèmes de point fixe généraux. En revanche, on s'attend ici à ce que les liens entre sémantique opérationnelle et dénotationnelle soient étudiés (toujours dans le cas d'un langage jouet). Il est aussi important que la leçon présente des exemples d'utilisation des notions introduites, comme des preuves d'équivalence de programmes ou des preuves de correction de programmes.

2 Coeur de la leçons

- Sémantique opérationnelle (petit/grand pas) et dénotationnelle des expressions arithmétiques élémentaires.
- Adéquation et de correction entre les sémantiques
- Utiliser la sémantique pour énoncer des propriétés (correction d'un programme, terminaison, validité d'une optimisation)

3 À savoir

- Sémantiques des commandes d'um langage jouet choisi.
- Exemples de programmes, avec une interprétation de leur correction dans une des sémantiques présentées
- Justification des choix des constructeurs du langage utilisé
- Étudier les effets et leur traitement dans les différents modèles (non-terminaison, non-déterminisme)
- Lien avec d'autres leçons : sémantique petit pas pour le lambda-calcul et les machines de Turing, sémantique à grand pas pour la logique (arbres de preuve), sémantique dénotationnelle pour les expressions rationnelles (langage).
- Utiliser la sémantique pour énoncer des propriétés (correction d'un programme, terminaison, validité d'une optimisation)

4 Ouvertures possibles

- Éviter le formalisme des CPO n'est pas forcément le plus simple

5 Conseils au candidat

— Il n'est pas nécessaire d'aller vers les langages fonctionnels pour faire une bonne leçon, IMP suffit largement !

— Bien faire attention aux références qui prennent des directions *incompatibles* très tôt. Par exemple, si le domaine de définition des variables est $\mathbb{N}$ alors l'opération de soustraction doit être *binaire*, alors que dans le cas de $\mathbb{Z}$ on peut se contenter d'une soustraciton *unaire*.

— Il faut présenter les trois principales sémantiques (petit pas, grand pas, dénotationnel) mais ne pas oublier de *justifier* leur intérêt propre ! Se demander pourquoi on en a trois est d'ailleurs pertinent.

6 Questions classiques

— **TMP** Le grand classique des CPO plats, et de la boucle while qui converge en un nombre fini d'étapes... Sauf que c'est pour une valeur fixée ! **TMP**

— Quel est l'intérêt des différentes sémantiques ?

— Pour quelle sémantique la notion de complexité est-elle la plus naturelle ?

— Pour le programme d'une centrale nucléaire, quelle sémantique paraît la plus adaptée ?

— Peut-on faire un lien entre la sémantique dénotationnelle et la preuve de programme ?

— Calculer la sémantique dénotationnelle d'un programme avec une boucle while simple (division euclidienne, fibonacci...)

— Conaissez-vous un langage pour lequel il n'y a pas de théorème d'adéquation ? De théorème de correction ?

— Connaissez-vous un autre moyen de définir une sémantique ? (Jeux, plus faible précondition)

— *(si les CPO ont été introduits)* Justifier que le $(\sup f_i)(x) = \sup f_i(x)$ si f_i est une famille dirigée d'applications scott-continues.

— Comment ajouter un choix non déterministe dans vos sémantiques ?

— Comment ajouter un choix probabiliste (pièce non biaisée par exemple) ?

7 Références

— **TMP** Glynn Winskel **TMP**

— **TMP** `http://www.cl.cam.ac.uk/~gw104/dens.pdf` **TMP** (aussi Glynn Winskel)

8 Dev

— **TMP** Complétude de Hoare **TMP**

— **TMP** Équivalence de sémantiques **TMP**

— **TMP** Calcul de la sémantique d'un programme (factorielle, division euclidienne) **TMP**