Leçon 930 – Sémantique des langages de programmation. Exemples.

15 juillet 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 Cœur de la leçon

— Sémantique opérationnelle (petit/grand pas) et dénotationnelle des commandes de IMP.

— Équivalences entre les sémantiques.

— Exemples de programmes, avec une interprétation de leur correction dans une des sémantiques présentées ci-dessus.

3 À savoir

— Utiliser la sémantique pour énoncer des propriétés : correction d'un programme, terminaison, validité d'une optimisation.

— Compositionnalité de la sémantique dénotationnelle, qui fait que l'équivalence est une congruence.

— Identifier les relations fonctionnelles ou totales.

4 Ouvertures possibles

— Sémantique axiomatique (logique de Hoare).

— Paresse dans les expressions booléennes et non terminaison.

— Effets de bords dans les expressions.

— Instructions break et continue (en petit pas ou axiomatique).

5 Conseils au candidat

— Dans IMP, seule la non terminaison empêche d'avoir un résultat dans la sémantique à grand pas (ou dénotationnelle), ce qui n'est pas toujours le cas. En toute généralité, la non terminaison se définit plutôt à petit pas.

— Définir les expressions comme des termes.

— On peut définir la sémantique dénotationelle du while simplement comme un sup, et ainsi éviter le formalisme des CPO. Si on parle de point fixe, il faut connaître le théorème d'existence, et illustrer son intérêt. Par exemple, cela montre que le while et son déroulé ont la même sémantique.

— 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 soustraction *unaire*.

— Réfléchir aux liens avec d'autres leçons : sémantique petit pas pour le lambda-calcul et les machines de Turing, sémantique dénotationnelle pour les expressions régulière (langage).

6 Questions classiques

— Quel est l'intérêt des différentes sémantiques ?

— Quelles sémantiques permettent de définir une notion de complexité ?

— Quelles sémantiques sont utilisées en pratique pour vérifier des programmes ?

— Quelle sémantique permet de vérifier un programme qui ne termine pas ?

— 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...)

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

— Vérifier la continuité de la fonction utilisée pour définir le WHILE ? (si il a été défini ainsi)

— Comment ajouter un choix non déterministe dans vos sémantiques ?

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

— Est-ce que IMP est Turing-complet ? L'équivalence est-elle décidable ?

7 Références

— [*Win*] The formal semantics of programming langages - Winskel - à la BU/LSV

8 Dev

++ Équivalence des sémantiques opérationnelle et dénotationnelle - ([Win], *p. 61*) - 930
On peut aussi faire l'équivalence entre petit-pas et grand-pas, mais moins intéréssante.

++ Calcul de la sémantique d'un ou plusieurs programmes - (?,) - 930
Dans l'idéal, on montre que deux programmes sont équivalents, via des calculs de sémantiques, potentiellement dans des sémantiques différentes.