Logique et informatique

Jean Goubault-Larrecq

Le 02 août 1999

Le but de ce cours est de comprendre la sémantique des langages fonctionnels, et la relation intime qu'elle entretient avec le lambda-calcul et la théorie de la démonstration. Parmi les langages fonctionnels, on compte notamment Lisp, Miranda, Haskell et ML. Un outil que l'on utilisera beaucoup est la notion de continuation, qui a été inventée pour donner une sémantique aux langages impératifs, comme Pascal ou C, et nous verrons que cette notion est très liée aux diverses traductions de la logique classique en logique intuitionniste pour l'aspect logique, et à la notion de point de programme pour l'aspect informatique.

Ce cours est structuré en trois parties :

1. **Lambda-calcul et langages fonctionnels** : il s'agit ici de donner une idée de ce que sont les langages fonctionnels et leur sémantique. Un outil de base sera le lambda-calcul, que l'on peut voir soit comme le noyau essentiel de tous ces langages, soit lui-même comme un langage fonctionnel universel. On en verra la sémantique opérationnelle (règles de réduction) et quelques modèles dénotationnels.

2. **Aspects logiques** : où l'on examinera diverses disciplines de typage du lambda-calcul, et où l'on verra que le lambda-calcul typé peut aussi être vu comme une notation pour des preuves dans divers systèmes logiques. La connexion entre logiques et langages de programmation, souvent appelée correspondance de Curry-Howard, a des conséquences parfois surprenantes. Nous l'appliquerons en particulier à démontrer que la logique classique est conservative sur la logique intuitionniste pour les formules Π^0_2.

3. **Aspects informatiques** : le but ici sera de comprendre comment on peut faire exécuter un programme fonctionnel par un ordinateur, soit en l'interprétant, soit en le compilant, tout en profitant des notions introduites dans les parties précédentes.