Programmer = démontrer?
La correspondance de Curry-Howard aujourd'hui

Introduction générale

Xavier Leroy

Collège de France

2018-11-21

Informatique et logique mathématique

Depuis ses débuts, l'informatique tire de nombreuses idées et de nombreux principes de la logique mathématique.

(Church, Turing, von Neumann, et bien d'autres fondateurs de l'informatique étaient logiciens.)

Le cours 2018 est consacré à l'une de ces interactions, particulièrement intense et féconde, entre logique et informatique, et plus précisément entre théorie de la démonstration et langages de programmation.

Programmer avec de la logique

De nombreux langages informatiques présentent les programmes comme des formules d'une certaine logique :

- Programmation logique (Prolog, Mercury, …)
- Programmation par contraintes (Prolog III, CHIP, Oz, …)
- Requêtes dans les bases de données relationnelles.

L'exécution du programme revient à démontrer ou à réfuter la proposition logique correspondante.

Exemple en Prolog

Un programme en Prolog :

```
append([], L, L).
append([H|T], L, [H|M]) :- append(T, L, M).

?- append(X, Y, [1,2,3]).
```

Les clauses de Horn et la requête définissent une formule logique :

$$\begin{array}{rl} & (\forall L, \mathtt{append}([\,], L, L)) \\ \wedge & (\forall H, T, L, M, \mathtt{append}(T, L, M) \Rightarrow \mathtt{append}([H|T], L, [H|M])) \\ \Rightarrow & \exists X, Y, \mathtt{append}(X, Y, [1, 2, 3]) \end{array}$$

Exécuter le programmer revient à démontrer cette proposition en trouvant les X, Y qui satisfont le $\exists$.

Proposition = programme

Tout cela s'inscrit dans une correspondance assez naturelle mais qui n'est pas la correspondance de Curry-Howard :

langage de programmation	logique mathématique
programme	proposition
exécution	démonstration

La correspondance de Curry-Howard

Un isomorphisme entre une logique intuitionniste et le λ-calcul simplement typé (un noyau de langage de programmation fonctionnelle).

λ-calcul simplement typé	logique intuitionniste
type	proposition
terme (programme)	démonstration
réduction (exécution)	élimination des coupures (normalisation)

Changement radical de point de vue : les programmes ne sont plus des propositions, mais les démonstrations de ces propositions.

La correspondance de Curry-Howard

Inspire un regard nouveau sur les logiques et sur les langages de programmation, résumé par le slogan ≪PAT≫ :

Propositions As Types,
Proofs As Terms

Le slogan survit en franglais

Propositions = Types,
Preuves = Termes

mais pas en bon français

Propositions = Types,
Démonstrations = Termes

Programmer = démontrer ?

Loin d'être une coincidence, la correspondance de Curry-Howard s'est développée en un lien structurel profond entre langages et logiques, entre programmation et démonstration.

- Correspondances entre d'autres logiques et d'autres langages, plus expressifs, p.ex. l'arithmétique du second ordre et lambda-calcul polymorphe.
- Formalismes qui sont à la fois des logiques et des langages : la théorie des types de Martin-Löf, le calcul des constructions, ...
- Outils qui sont utilisables comme assistants à la démonstration et comme environnement de programmation : Coq, Agda, ...

Curry-Howard aujourd'hui

Un principe directeur pour concevoir, comprendre et formaliser les langages de programmation (fonctionnelle, mais pas que).

Un regard neuf sur les mathématiques classiques : quel est le contenu calculatoire d'une démonstration ?

De nouvelles manières de programmer, intégrant davantage la vérification formelle (types dépendants, etc).

De nouvelles manières de faire des mathématiques en s'aidant de la puissance de l'ordinateur.

Des outils puissants et polyvalents comme Coq et Agda pour parcourir cette frontière entre informatique et mathématiques.

Le séminaire

- 28 nov : Pierre-Évariste Dagand, *Les types dépendants : tout un programme !*
- 5 déc : Assia Mahboubi, *Mathématiques assistées par ordinateur*
- 12 déc : Matthieu Sozeau, *Programmer avec Coq : récursion et filtrage dépendant*
- 19 déc : Guillaume Munch-Maccagnoni, *Peut-on dupliquer un objet ? Linéarité et contrôle des ressources*
- 16 jan : Alexandre Miquel, *Forcing et réalisabilité : vers l'unification*
- 23 jan : Christine Tasson, *Sémantique des programmes fonctionnels probabilistes, à la lumière de la logique linéaire*
- 30 jan : Thierry Coquand, *Du calcul des constructions à la théorie des types univalente*