Découverte de l'assistant à la preuve Coq

Damien Pous, CNRS

Journées nationales de l'APMEP
Grenoble, 24.1o.2o11

Qu'est-ce que Coq?

- Un assistant à la preuve
- Un langage de programmation
- Un langage de spécification
- Un prouveur interactif
- Un projet démarré par Thierry Coquand en 1984

toujours en ébullition!

A quoi ça sert?

- Formaliser et vérifier des théorèmes
  - Théorème des quatre couleurs (2005)
  - Classification des groupes finis (en cours)
- Construire des logiciels certifiés
  - Compcert: un compilateur C entièrement certifié (2008)

Qu'est ce que Coq n'est pas?

- Un language de programmation ultra-rapide
- Un prouveur automatique
- Un oracle
- Quelque chose de facile à utiliser (malheureusement)

Principes: Poincaré

- Les preuves mathématiques peuvent être très compliquées,
- mais vérifier ces preuves reste relativement simple.

Principes: Poincaré

- Les preuves mathématiques peuvent être très compliquées,
- mais vérifier ces preuves reste relativement simple.

- Coq est formé d'un petit noyau pour exprimer les preuves (petit et vérifiable),
- sur lequel on s'appuit pour définir des tactiques (arbitrairement compliquées).

Principes: correspondance de Curry-Howard

"les preuves sont des programmes" (et inversement)

$$\begin{array}{lccccccc} p: & (A \to B) & \to & (B \to C) & \to & A & \to & C \\ & f & \mapsto & g & \mapsto & x & \mapsto & g(f(x)) \end{array}$$

Principes: correspondance de Curry-Howard

"les preuves sont des programmes" (et inversement)

$$\begin{array}{lccccccc} p: & (A \to B) & \to & (B \to C) & \to & A & \to & C \\ & f & \mapsto & g & \mapsto & x & \mapsto & g(f(x)) \end{array}$$

proposition P	type T	(interface)
preuve p	programme t	(implémentation)
vérification de preuve	vérification de type	

Au travail!

- On va découvrir Coq par la pratique en faisant:
  1. un peu de logique,
  2. des exercices sur les nombres premiers,
  3. une preuve du théorème de Bachet-Bézout et du théorème de Gauss.

- `http://sardes.inrialpes.fr/~pous/apmep/`

Conclusions

- Ouch, ça a pris plus de temps que prévu!
- On peut écrire facilement des théorèmes en Coq;
  mais les prouver, c'est plus dur...
- On a utilisé:
  - des tactiques de base (`intros`, `apply`, `split`, `rewrite`),
  - des tactiques plus évoluées (`ring`, `in_seq`, `auto`).
- Et on a fini par y arriver!