Séance 1 - A la découverte de Coq !

Amélie Ledein

Avant de commencer ...

- Cette UE = 22h30 de présence
- Cette UE = 6 ECTS
- "La valeur d'un crédit représente environ 25 à 30 heures de travail"

Qu'est-ce que Coq ?

- Un **assistant de preuve** développé par Inria depuis 1984,
  - Écrire des preuves complètement formelles est fastidieux
  - Idée : utiliser un ordinateur pour vérifier les raisonnements
  - Les logiciels permettant cette interaction entre l'homme et l'ordinateur sont appelés des assistants de preuve (**proof assistant/interactive theorem prover** en anglais)
  - Plusieurs outils existent : Coq, Isabelle/HOL, Agda, Mizar, PVS, etc.
- basé sur le **Calcul des Constructions Inductives**.
- Réussites :
  - En mathématiques : théorème des quatre couleurs (2004), théorème de Feit et Thomson (2012)
  - En informatique : CompCert un compilateur certifié pour le langage C

Comment vérifier un raisonnement avec un ordinateur ?

On voudrait pouvoir :

1. Définir des objets
2. Énoncer des théorèmes
3. Les prouver

Des langages différents (en première approximation) :

1. Un langage fonctionnel pur appelé Gallina
2. Une logique d'ordre supérieure
   - On peut quantifier sur des fonctions et des prédicats.
3. Un ensemble de tactiques

Tutoriel 1 - Coq, un assistant à la preuve

- Logique propositionnelle :
  - Constantes : `True`, `False`
  - Connecteurs : $\sim$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique)
- Logique des prédicats :
  - Quantificateurs : `forall`, `exists`
- Quelques théories :
  - Théorie de l'égalité, par exemple

Note 1 Certaines tactiques de base ont une correspondance évidente dans le calcul des séquents ou en déduction naturelle.

Note 2 L'utilisateur peut définir des tactiques (par exemple une tactique qui enchaîne plusieurs tactiques de base).

Tactiques pour manipuler la logique

- `intro`/`intros` pour introduire les variables quantifiées universellement
- `exists x` pour donner la valeur témoin attendue
- `split` pour détruire un but en plusieurs sous-buts
- `left`, `right` pour choisir une branche d'une disjonction
- `reflexivity` pour utiliser la réflexivité de l'égalité
- `symmetry` pour utiliser la symétrie de l'égalité
- `contradiction` si les hypothèses contiennent False ou encore (P et $\sim$ P)

Tactiques pour manipuler les hypothèses et le but

- `exact H` lorsque l'hypothèse H et le but actuel sont identiques
- `assumption` pour chercher dans les hypothèses
- `apply H` pour appliquer H dont la conséquence est la conclusion actuelle
- `rewrite [←] H` pour récrire en utilisant l'égalité H
- `replace x with y` pour remplacer x par y en prouvant x=y
- `subst x` pour éliminer x en utilisant les égalités disponibles
- `assert H as Ha` pour introduire un lemme intermédiaire H pendant la preuve
- `specialize (H x)` pour appliquer partiellement une hypothèse
- `pose proof lem as H` pour ajouter un théorème aux hypothèses
- `simpl` pour simplifier des termes
- `unfold f` pour remplacer f par sa définition
- `admit` pour admettre un but afin de faciliter le développement d'une preuve