

Preuves assistées par ordinateur – TD n° 2

Premiers pas en Coq

La documentation du système Coq est consultable en ligne : <http://coq.inri.fr/doc/>

Démarrage du système

Il existe actuellement trois manières principales d'utiliser Coq :

- via `coqide`, un interface graphique basé sur `gtk`
- via `proofgener` 1, qui est un module pour `emacs`
- ou éventuellement en lançant `coqtop`, un bouclier d'interaction textuelle à la console

Chaque méthode a ses aficionados (même la dernière). Pour utiliser `proofgener` 1 sur les machines de l'UFR, il suffit d'ajouter la ligne (`load-file "~letouzey/.emacs-coq"`) à votre `~/.emacs` puis lancer `emacs` sur un fichier Coq (d'extension `.v`).

Une fois lancé, les deux interfaces `coqide` et `proofgener` 1 proposent une disposition assez similaire : le fichier en cours d'édition est à gauche, tandis que les preuves en cours sont affichées en haut à droite, et les messages du système en bas à droite (réponses de Coq ou messages d'erreurs).

Commandes Coq

En Coq, une commande est formée d'un nom de commande (commençant par une majuscule), éventuellement suivi d'un ou plusieurs arguments, terminée par un point. Exemples :

<code>Check 0.</code>	<code>Check 2 + 2 = 5.</code>
<code>Check S.</code>	<code>Check forall x, exists y, x = 2 * y /\ x = 2 * y + 1.</code>
<code>Check nat.</code>	<code>Definition id := fun (A : Set) (x : A) => x.</code>
<code>Print nat.</code>	<code>Check id.</code>
<code>SearchAbout nat.</code>	<code>Check id nat 7.</code>

Après avoir tapé quelques commandes, soumettez-les à Coq, en utilisant l'un des moyens de navigation (icônes, menus ou raccourcis clavier). Observez le déplacement de la zone colorée marquant la partie du fichier déjà exécutée.

Passage au mode preuve

L'utilisateur déclare son intention d'effectuer une preuve avec une commande de la forme

```
Lemm nd_commut :
  forall A B : Prop, A /\ B <-> B /\ A.
```

On notera que cette commande donne un nom (ici : `nd_commut`) au lemme, ce qui permettra de le référencer par la suite. (On peut remplacer `Lemm` par `Fct`, `Proposition` ou `Theorem`). Sans que cela soit obligatoire, on peut marquer le début de la preuve par la commande `Proof`.

Sous-buts et tactiques

Un fois le mod pr uv lancé, la zone supérieure gauche affiche en permanence un ou plusieurs sous-buts (*subgoals*) qu'il s'agit de démontrer. Ces sous-buts sont essentiels montrant des séquences de la déduction naturelle écrits verticalement : les hypothèses (nommées) sont en haut, et la conclusion figure en bas sous un trait. Dans la partie haute figurent également des déclarations de variables.

La pr uv se fait à l'aide de *tactiques* (distinguées des commandes par un minuscule initial), qui effectuent des transformations plus ou moins complètes sur le but courant. Par exemple, la tactique `intro` effectuée sur un but de la forme $A \rightarrow B$ introduit une hypothèse $H : A$ dans le contexte, et remplace la conclusion par B . À chaque règle d'inférence de la déduction naturelle correspond un ou plusieurs tactiques, mais certaines tactiques permettent également d'effectuer des modifications plus complètes, comme par exemple la résolution de contraintes linéaires en arithmétique de Prsburger (tactique `omega`).

Les tactiques sont susceptibles d'engendrer de nouveaux sous-buts (correspondant aux prémisses), ou au contraire de faire disparaître le but courant (lorsque celui-ci est résolu). La pr uv est terminée lorsqu'il n'y a plus de sous-but à démontrer. On passe alors au mode « commande » avec la commande `Qed`¹. Voici par exemple un pr uv complet pour l'énoncé précédent.

```
Lemma and_commut :  
  forall A B : Prop, A /\ B <-> B /\ A.  
Proof.  
  intros; split; intros.  
  destruct H; split; assumption.  
  destruct H; split; assumption.  
Qed.
```

Un tel «script de pr uv», une fois sauvegardé dans un fichier `mes_preuves.v`, peut ensuite être «compilé» via la commande unix `coqc mes_preuves.v`. Si les pr uv s qu'il contient ce fichier sont correctes, un fichier compilé `mes_preuves.vo` est alors produit, qui permettrait de charger ces pr uv s rapidement par la suite (cf. la commande `Require Import`).

Exercice 1 – Calcul propositionnel Poser l'existence de variables propositionnelles A, B, C via la commande `Prop, puis prouver en Coq les formules suivantes :`

1. $A \rightarrow A$
2. $(A \rightarrow B) \rightarrow (B \rightarrow C) \rightarrow A \rightarrow C$
3. $A \wedge B \leftrightarrow B \wedge A$
4. $A \vee B \leftrightarrow B \vee A$
5. $A \wedge (B \wedge C) \leftrightarrow A \wedge (B \wedge C)$
6. $(A \vee B) \vee C \leftrightarrow A \vee (B \vee C)$
7. $A \rightarrow \neg\neg A$
8. $(A \rightarrow B) \rightarrow \neg B \rightarrow \neg A$
9. $\neg\neg(A \vee \neg A)$

1. *Q od erat demonstrand m*, le « CQFD » latin.

Exercice 2 – Calcul des prédicats Après avoir effectué les déclarations suivantes

```
Parameter X Y : Set.  
Parameter A B : X -> Prop.  
Parameter R : X -> Y -> Prop.
```

établir en Coq les formules suivantes :

1. $(\forall x, A x \wedge B x) \leftrightarrow (\forall x, A x) \wedge (\forall x, B x)$
2. $(\exists x, A x \vee B x) \leftrightarrow (\exists x, A x) \vee (\exists x, B x)$
3. $(\exists y, \forall x, R x y) \rightarrow \forall x, \exists y, R x y$

Exercice 3 – Reprise Refaire en Coq les exercices 1 à 4 de la feuille de TD n°1. Pour simuler la règle de raisonnement par l'absurde, on pourra déclarer l'axiome suivant :

```
Axiom not_not_elim : forall A : Prop, ~~A -> A.
```