eBook Gratuit

APPRENEZ coq

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#coq

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: coq

It is an unofficial and free coq ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official coq.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec coq

Remarques

Cette section fournit une vue d'ensemble de ce qu'est le coq et des raisons pour lesquelles un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans coq, et établir un lien avec les sujets connexes. La documentation de coq étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Tester et installer Coq

Tester sans installer

Pour les nouveaux utilisateurs qui souhaitent tester Coq sans l'installer sur leur machine, il existe un IDE en ligne appelé JsCoq (présentation ici). La sous-fenêtre du package permet de tester divers packages supplémentaires bien connus.

Installation

La page de téléchargement contient des programmes d'installation pour Windows et MacOS.

Les utilisateurs de Linux sont généralement invités à compiler à partir de sources utilisant opam, afin d'obtenir la dernière version. Des instructions de base sur la façon de le faire sont données ici .

Une preuve simple

```
Theorem my_first_theorem : 1 + 1 = 2.
Proof.
  reflexivity.
Qed.
```

Essayez-le dans votre navigateur .

Installer Coq sur MacOS

Vous pouvez installer l'ensemble en téléchargeant le paquet dmg d' ici .

Le bundle contient un CoqIDE qui peut être utilisé pour écrire vos preuves ou vous pouvez utiliser la commande `coqtop` pour exécuter l'interpréteur sur votre terminal.

L'installation de Coq sur MacOS est également facile en utilisant l'homebrew

```
brew install coq
```

ou si vous utilisez MacPorts

```
sudo port install coq
```

Il n'y a pas de bon support vi pour Coq. Vous pouvez utiliser Proof General dans emacs qui a une bonne facilité d'utilisation.

Pour installer Proof General, supprimez les anciennes versions de Proof General clone la nouvelle version de GitHub

```
git clone https://github.com/ProofGeneral/PG ~/.emacs.d/lisp/PG
cd ~/.emacs.d/lisp/PG
make
```

Ajoutez ensuite ce qui suit à vos .emacs:

```
;; Open .v files with Proof General's Coq mode
(load "~/.emacs.d/lisp/PG/generic/proof-site")
```

Assurez-vous que les emacs sur lesquels vous exécutez les véritables Emacs. Si vous rencontrez des problèmes d'incompatibilité de version, vous devrez peut-être exécuter makefile en spécifiant explicitement le chemin Emacs

```
make clean; make EMACS=/Applications/Emacs.app/Contents/MacOS/Emacs
```

Installation avec Nix

Attention: ce n'est pas la manière standard d'installer Coq.

Pour les utilisateurs de Linux (et MacOS) qui souhaitent avoir accès à des versions à jour de Coq ou pour pouvoir utiliser plusieurs versions de Coq sur le même ordinateur, sans avoir à utiliser opam, et sans avoir à compiler depuis source, c'est une solution alternative.

Nix est un gestionnaire de paquets pour les systèmes d'exploitation de type Unix tels que Linux et MacOS. Il est livré avec sa propre collection de paquets qui est généralement beaucoup plus à jour que celle de Debian ou Ubuntu. Il n'entre pas en conflit avec le gestionnaire de paquets de votre distribution car il n'installe rien dans `/usr/bin` et autres.

Tout d'abord, vous devez installer Nix :

```
$ curl https://nixos.org/nix/install | sh
```

Pour vous assurer que les variables d'environnement nécessaires sont définies, connectez-vous à nouveau ou tapez:

```
. $HOME/.nix-profile/etc/profile.d/nix.sh
```

Ensuite, la commande suivante installera la dernière version de Coq:

```
$ nix-env -iA nixpkgs.coq_8_6
```

Vous pouvez également exécuter CoqIDE sans rien ajouter à votre PATH:

```
$ nix-shell -p coq_8_6 --run coqide
```

De même (en supposant que Emacs et Proof-General soient déjà installés):

```
$ nix-shell -p coq_8_6 --run emacs
```

Ceci est très utile pour exécuter différentes versions lorsque vous en avez besoin. Par exemple, pour exécuter Coq 8.5, utilisez la commande suivante:

```
$ nix-shell -p coq_8_5 --run coqide
```

Exemple de preuve par induction

Voici une preuve simple par induction.

```
Require Import Coq.Setoids.Setoid.
Require Import Coq.Arith.Lt.

(* A number is less than or equal to itself *)
Theorem aLTEa : forall a,
    a <= a.
    auto with arith. (* This follows by simple arithmetic *)
    Qed.

Theorem simplALTE : forall a b,
    S a <= S b <-> a <= b. (* If a <= b, then a + 1 <= b + 1 *)
Proof.
Admitted.

Theorem ltAlwaysLt: forall a b,
    a <= a + b.
Proof.
  intros. (* Introduce relevant variables *)
  induction a, b. (* Induction on every variable *)
  simpl. apply aLTEa. (* 0 <= 0 + S b *)
  rewrite -> plus_O_n. auto with arith. (* 0 <= S b *)
  rewrite <- plus_n_O. apply aLTEa. (* S a <= S a + 0 *)
  rewrite <- simplALTE in IHa. (* IHa: a <= a + S b. Goal: S a <= S a + S b. *)
  apply IHa. (* We rewrote the induction hypothesis to be in the same form as the goal, so it
applies immediately now *)
  Qed.
```

Lire Commencer avec coq en ligne: https://riptutorial.com/fr/coq/topic/3762/commencer-avec-coq

Chapitre 2: Recherche d'un fait existant avec Recherche et variantes

Syntaxe

- `Search qualid.` (* pour Coq 8.4 et versions plus récentes *)
- `SearchAbout qualid.` (* synonyme obsolète. *)

Paramètres

Paramètre	La description
`qualid`	Identifiant ou modèle à rechercher. Il peut s'agir de notations

Remarques

Avant Coq 8.4, `Search` avait le sens de `SearchHead` actuel: rechercher uniquement les faits où le motif correspond dans la conclusion de l'instruction.

Examples

Faits sur un identifiant particulier

Pour voir tous les faits concernant le `le` rapport du prélude:

```
Coq < Search le.
le_n: forall n : nat, n <= n
le_S: forall n m : nat, n <= m -> n <= S m
...
max_l: forall n m : nat, m <= n -> Nat.max n m = n
max_r: forall n m : nat, n <= m -> Nat.max n m = m
...
```

Pour rechercher tous les faits impliquant la notation `<` :

```
Coq < Search "<".
exists_lt: forall (Q : nat -> Prop) (k l : nat), exists_between Q k l -> k < l
in_int_intro: forall p q r : nat, p <= r -> r < q -> in_int p q r
in_int_lt: forall p q r : nat, in_int p q r -> p < q
...
```

Recherche d'un motif

Rechercher tous les faits impliquant un motif dans une hypothèse ou une conclusion:

```
Coq < Search (_ + O).
plus_n_O: forall n : nat, n = n + 0
```

Le caractère _ sert de caractère générique, il peut être utilisé plusieurs fois:

```
Coq < Search (S _ <= _).
le_S_n: forall n m : nat, S n <= S m -> n <= m
le_n_S: forall n m : nat, n <= m -> S n <= S m
```

Vous pouvez également rechercher des motifs non linéaires:

```
Coq < Search (?x <= ?x).
le_n: forall n : nat, n <= n
```

Recherche d'un motif dans la conclusion d'un lemme

Recherchez un lemme quand vous savez quelle devrait être sa conclusion:

```
Coq < SearchPattern (S _ <= _).
le_n_S: forall n m : nat, n <= m -> S n <= S m
```

Vous pouvez également rechercher une conclusion partielle (la conclusion et une ou plusieurs dernières hypothèses).

```
Coq < Require Import Arith.
Coq < SearchPattern (?x <= ?y -> ?y <= _ -> ?x <= _).
Nat.le_trans: forall n m p : nat, n <= m -> m <= p -> n <= p
```

Attention: si vous mélangez l'ordre des hypothèses, vous ne trouverez rien:

```
Coq < SearchPattern (?y <= _ -> ?x <= ?y -> ?x <= _).
```

Lire Recherche d'un fait existant avec Recherche et variantes en ligne:
https://riptutorial.com/fr/coq/topic/4007/recherche-d-un-fait-existant-avec-recherche-et-variantes

Chapitre 3: Utiliser la tactique

Introduction

Cette section comprend des informations sur l'utilisation de diverses tactiques et techniques Coq (analyse de cas, preuve par induction, auto, etc.) pour prouver des théorèmes.

Examples

Exemple trivial d'une analyse de cas

En Coq, la `destruct` plus ou moins correspond à une analyse de cas. Il est similaire à l'induction sauf qu'il n'y a pas d'hypothèse d'induction. Voici un exemple (certes plutôt trivial) de cette tactique:

```
Require Import Coq.Arith.Lt.

Theorem atLeastZero : forall a,
0 <= a.
Proof.
  intros.
  destruct a. (* Case analysis *)
  - reflexivity. (* 0 >= 0 *)
  - apply le_0_n. (* S a is always greater than zero *)
  Qed.
```

Lire Utiliser la tactique en ligne: https://riptutorial.com/fr/coq/topic/8335/utiliser-la-tactique

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec coq	Anton Trunov, Apoorv Ingle, bukzor, Community, EJoshuaS, Zimm i48
2	Recherche d'un fait existant avec Recherche et variantes	pintoch, Tej Chajed, Zimm i48
3	Utiliser la tactique	EJoshuaS, Zimm i48