EBook Gratis

APRENDIZAJE coq

Free unaffiliated eBook created from
Stack Overflow contributors.

#coq

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con coq

Observaciones

Esta sección proporciona una descripción general de qué es coq y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de coq, y vincular a los temas relacionados. Dado que la Documentación para coq es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Prueba e instalación de Coq.

Pruebas sin instalar

Para los usuarios nuevos que deseen comenzar a probar Coq sin instalarlo en su máquina, hay un IDE en línea llamado JsCoq (presentación aquí). La subventana del paquete permite probar varios paquetes adicionales conocidos.

Instalación

La página de descarga contiene instaladores para Windows y MacOS.

En general, se recomienda a los usuarios de Linux que compilen desde la fuente usando opam, para obtener la última versión. Aquí se dan instrucciones básicas sobre cómo hacerlo.

Una prueba sencilla

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

Pruébalo en tu navegador .

Instala Coq en MacOS

Puede instalar todo el paquete descargando el paquete dmg desde aquí .

El paquete contiene un CoqIDE que puede usarse para escribir sus pruebas o puede usar el comando `coqtop` para ejecutar el intérprete en su terminal

La instalación de Coq en MacOS es fácil de usar también en Homebrew

```
brew install coq
```

o si usas MacPorts

```
sudo port install coq
```

No hay buen soporte vi para Coq. Puedes usar Proof General dentro de emacs que tiene una buena usabilidad.

Para instalar Proof General, elimine las versiones anteriores de Proof General y clone la nueva versión de GitHub.

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

Luego agrega lo siguiente a tu .emacs:

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

Asegúrese de que los emacs están ejecutando el Emacs real. Si enfrenta problemas de discrepancia de versión, es posible que tenga que ejecutar makefile nuevamente especificando la ruta de Emacs explícitamente

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

Instalación con Nix

Advertencia: esta no es la forma estándar de instalar Coq.

Para los usuarios de Linux (y MacOS) que deseen obtener acceso a versiones actualizadas de Coq o poder usar varias versiones de Coq en la misma máquina, sin la molestia de usar opam, y sin tener que compilar desde Fuente, esta es una solución alternativa.

Nix es un administrador de paquetes para sistemas operativos tipo Unix, como Linux y MacOS. Viene con su propia colección de paquetes que generalmente se mantiene mucho más actualizada que la de Debian o la de Ubuntu. No entra en conflicto con el administrador de paquetes de su distribución porque no instala nada en `/usr/bin` y demás.

Primero, necesitas instalar Nix :

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

Para asegurarse de que las variables de entorno necesarias estén establecidas, vuelva a iniciar sesión o escriba:

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

Luego el siguiente comando instalará la última versión de Coq:

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

También puede ejecutar CoqIDE sin agregar nada a su RUTA:

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

Del mismo modo (suponiendo que ya tiene instalado Emacs y Proof-General):

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

Esto es muy útil para ejecutar diferentes versiones cuando las necesite. Por ejemplo, para ejecutar Coq 8.5 use el siguiente comando:

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

Ejemplo de prueba por inducción.

Aquí hay una prueba simple por inducción.

```
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.
```

Lea Empezando con coq en línea: https://riptutorial.com/es/coq/topic/3762/empezando-con-coq

Capítulo 2: Buscando un hecho existente con Búsqueda y variantes.

Sintaxis

- `Search qualid.` (* para Coq 8.4 y versiones más recientes *)
- `SearchAbout qualid.` (* sinónimo en desuso. *)

Parámetros

Parámetro	Descripción
`qualid`	El identificador o patrón a buscar. Puede implicar notaciones.

Observaciones

Antes de Coq 8.4, la `Search` tenía el significado del `SearchHead` actual: solo busque hechos donde el patrón coincida en la conclusión de la declaración.

Examples

Hechos sobre un identificador particular

Para ver todos los hechos que implican a `le` relación entre el preludio:

```
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
...
```

Para buscar todos los hechos relacionados con la notación `<` :

```
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
...
```

Buscando un patrón

Busque todos los hechos que involucren un patrón en una hipótesis o conclusión:

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

El carácter _ sirve como comodín, se puede usar varias veces:

```
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
```

También puedes buscar patrones no lineales:

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

Buscando un patrón en la conclusión de un lema.

Busque un lema cuando sepa cuál debería ser su conclusión:

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

También puede buscar una conclusión parcial (la conclusión y una o varias últimas hipótesis).

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

Advertencia: si confunde el orden de las hipótesis, no encontrará nada:

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

Lea Buscando un hecho existente con Búsqueda y variantes. en línea: https://riptutorial.com/es/coq/topic/4007/buscando-un-hecho-existente-con-busqueda-y-variantes-

Capítulo 3: Usando Tactics

Introducción

Esta sección incluye información sobre cómo usar varias tácticas y técnicas de Coq (análisis de casos, prueba por inducción, auto, etc.) para probar teoremas.

Examples

Ejemplo trivial de un análisis de caso.

En Coq, `destruct` más o menos corresponde a un análisis de caso. Es similar a la inducción, excepto que no hay hipótesis de inducción. Aquí hay un ejemplo de esta táctica (aunque bastante trivial):

```
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.
```

Lea Usando Tactics en línea: https://riptutorial.com/es/coq/topic/8335/usando-tactics

Creditos

S. No	Capítulos	Contributors
1	Empezando con coq	Anton Trunov, Apoorv Ingle, bukzor, Community, EJoshuaS, Zimm i48
2	Buscando un hecho existente con Búsqueda y variantes.	pintoch, Tej Chajed, Zimm i48
3	Usando Tactics	EJoshuaS, Zimm i48