

Faire des preuves en Focal

Damien Doligez

INRIA

11 juin 2009

Plan

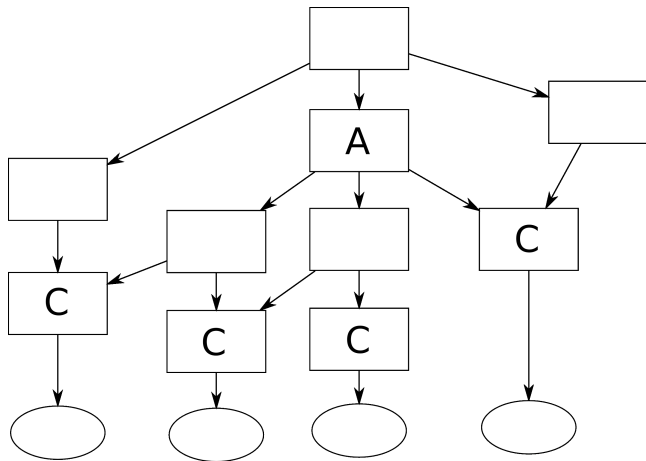
- 1 Contexte
- 2 Preuves simples
- 3 Preuves composées
- 4 Démo

Pourquoi faire des preuves

Quand faire les preuves

- Pour obtenir du code, il faut créer une collection.
- Pour créer une collection, il faut une espèce complète.
- Une espèce complète doit implémenter (avec `let`) toutes les fonctions déclarées (avec `signature`)
- **et** prouver (avec `proof of` ou avec `theorem`) toutes les propriétés déclarées (avec `property`).
- En pratique, il est préférable de prouver les propriétés le plus tôt possible (le plus haut possible dans l'arbre d'héritage), pour partager la preuve entre les espèces suivantes.

Quand faire les preuves



Plan

1 Contexte

2 Preuves simples

3 Preuves composées

4 Démo

Preuves simples

Il y a trois types de preuves simples en Focal :

- `assumed` (renoncer a faire la preuve)
- `by` (demander a Zenon de faire la preuve automatiquement)
- `coq proof` (faire la preuve en Coq a la main)

preuve par assumed

Format :

```
assumed {* commentaire *}
```

Cette technique est à utiliser dans deux cas :

- Quand on renonce temporairement à faire la preuve, par exemple pendant qu'on travaille sur un fichier.
- Quand on introduit un fait axiomatique, externe au programme Focal, en particulier dans la bibliothèque standard.

preuve par by

Format :

by *justification**

justification =

property *ident*, ...

definition of *ident*, ...

hypothesis *ident*, ...

step *label*, ...

type *ident*, ...

Cette methode de preuve invoque le prouveur automatique (Zenon) en lui disant de prouver le théorème à partir des propriétés et définitions qui sont listées.

Difficultés :

- Il faut trouver tout ce qui est nécessaire pour prouver le théorème.
- Zenon peut échouer si le théorème est trop compliqué.

preuve par coq proof

Format :

```
coq proof justification* { * script * }
```

- Il faut donner les *justifications* pour les avoir disponibles dans le contexte de la preuve Coq.
- Il faut écrire le script de preuve Coq à la main.

Pas pour les débutants.

Plan

- 1 Contexte
- 2 Preuves simples
- 3 Preuves composées**
- 4 Démo

Syntaxe

Format hiérarchique :

```
<1>1 assume x in Self,  
      assume H : P(x),  
      prove Q(x)
```

preuve

...

```
<1>x qed
```

preuve

Règles de liaison

<1>1 assume x in Self,
assume $H : P(x)$,
prove $Q(x)$

preuve (* x et H sont disponibles, prouver $Q(x)$ *)

<1>2 assume z in Self,
prove $R(z)$

preuve (* z et <1>1 sont disponibles *)

(* <1>1 = $\forall x \in \text{Self}, P(x) \Rightarrow Q(x)$ *)

<1>x qed

preuve (* <1>1 et <1>2 sont disponibles *)

Règles de liaison

```
<1>1 assume x in Self,  
      assume H : P(x),  
      prove Q(x)  
  preuve (* x et H sont disponibles, prouver Q(x) *)  
<1>2 assume z in Self,  
      prove R(z)  
  preuve (* z et <1>1 sont disponibles *)  
        (* <1>1 =  $\forall x \in Self, P(x) \Rightarrow Q(x)$  *)  
<1>x conclude  
  (* équivalent à qed by step <1>1, <1>2 *)
```

Plan

- 1 Contexte
- 2 Preuves simples
- 3 Preuves composées
- 4 Démo**

Démo