

Assistants de preuve

Fondements et applications

Christine Paulin-Mohring

Master Parisien de recherche en Informatique
Master Informatique Université Paris-Sud

18 septembre 2009

Introduction

- Construire des environnements interactifs pour le raisonnement mathématique
 - Un langage de description mathématique
 - Un moteur de **vérification** de preuve
 - Des procédures pour automatiser les preuves
- Applications
 - Mathématiques : théorème des 4 couleurs, théorème de Kepler ...
 - Preuve de programmes : compilateur C optimisant certifié ...
 - Modélisation informatique : plateforme JavaCard, cryptographie ...
- Thématiques
 - logique, démonstration automatique, programmation, interfaces ...

Exemple de preuve Coq

The screenshot shows the Coq IDE interface. The left pane displays the source code for a quicksort algorithm, and the right pane shows the current proof goals.

Source Code (quick.v):

```

Module QuickSort(O : OrderedType),
Module Facts := OrderedTypeFacts O,
Import O. Import Facts,
Require Import Coq.Sorting.PermutSetoid. Import Permutation,
Notation permutation := (permutation eq eq_dec),

Program Fixpoint split (pivot : t) (l : list t) {struct l} :
{ (inf,sup) : list t * list t |
  (forall x, (In x inf <-> (In x l & Lt x pivot)) &
   (In x sup <-> (In x l & ~ Lt x pivot))) &
  permutation l (inf ++ sup) } :=
  match l with
  | h :: tl => let '(inf,sup) :=split pivot tl in
              if lt_dec h pivot then (h :: inf, sup) else (inf, h :: sup)
  | nil => ([],[])
  end,
Next Obligation,
Proof,
intros; destruct_call split; clear split,
destruct x as [inf' sup']; simpl in *; destruct_conjs,
  
```

Proof Goals:

```

1 subgoal
pivot : t
h : t
tl : list t
inf : list t
sup : list t
inf' : list t
sup' : list t
H0 : forall x : t,
  (In x inf' <-> In x tl & Lt x pivot) &
  (In x sup' <-> In x tl & ~ Lt x pivot)
H1 : permutation tl (inf' ++ sup')
Heq_anonymous : (inf, sup) = (inf', sup')
H : lt h pivot
----- (1/1)
(forall x : t,
(h = x & In x inf <-> (h = x & In x tl) & Lt x pivot) &
(In x sup <-> (h = x & In x tl) & ~ Lt x pivot)) &
permutation (h :: tl) (h :: inf ++ sup)
  
```

Ready in QuickSort, proving split_obligation_1

Line: 34 Char: 60 CoqIDE started

Preuve de programme

The screenshot shows a C/C++ IDE with a code editor and a Prover View window.

Code Editor (cosine.c):

```

/*@ requires |x| <= 1./32
   @ ensures |\result - cos(x)| <= 2^(-23)
   @ */
float moncos(float x) {
    return 1.f-x*x*.5f;
}

```

Prover View Window:

		Simplify	Alt-Ergo	Yices	CoqIDE
▼ moncos_impl	●	■	■	■	■
1	●	■	■	■	■

Organisation

Deux cours qui se suivent

- **Fondements des systèmes de preuve**
 - notion de preuve en tant qu'objet mathématique et informatique
 - théorie des types
- **Assistants de preuve**
 - assistants de preuves en théorie des types
 - applications à la preuve de programmes
- Initiation à l'assistant **Coq** (mini-projet)
- Equipes dans ce domaine en France
(Rennes, Nice, Marseille, Bordeaux, Microsoft Research . . .)
en Europe (Chalmers, Munich Nottingham, Nijmegen,
Varsovie, Yale, Harvard . . .)
et ailleurs

Détails pratiques

- <http://www.mpri.master.univ-paris7.fr/>
- cours le **mardi de 08h45 à 11h45 à Chevaleret**
 - Fondements des systèmes de preuve :
15 septembre-20 novembre
Gilles Dowek, Sylvie Boldo
 - Assistants de preuve : 24 novembre-12 février
Christine Paulin-Mohring, Bruno Barras, Guillaume
Melquiond
- se faire connaître auprès des responsables