

INE22 : Preuves de Programmes

Introduction

Assia Mahboubi

INRIA - TypiCal, LIX École Polytechnique

24/04/2009

Pourquoi faire des preuves de programmes ?

Parce qu'un programme a priori c'est "faux".

Un premier exemple (28/09/07)

Ouvrez Microsoft Excel 2007[®]

- ▶ Dans la case A1, inscrivez 850
- ▶ Dans la case A2, inscrivez 77,1
- ▶ Dans la case A3, inscrivez la formule : = PRODUIT(A1 : A2)

Excel répond 100 000.

La réponse juste est 65 535.

Mais ce n'est qu'un bug d'affichage...

Merci S. Boldo

S'il fallait un autre exemple?

Le Zune[®], comportait cette tranche de code :

```
year = ORIGINYEAR; /* = 1980 */
```

```
while (days > 365)
{
    if (IsLeapYear(year))
    {
        if (days > 366)
        {
            days -= 366;
            year += 1;
        }
    }
    else
    {
        days -= 365;
        year += 1;
    }
}
```

Il en faut vraiment encore un ?

BinarySearch dans la JDK a été faux jusqu'en 2006. Ça veut dire pendant environ 9 ans.

En effet dans ce programme, l'instruction :

```
int mid = (low + high) / 2
```

doit calculer l'arrondi entier au plus proche de la moyenne.

Mais ça ne marche pas si low et high sont trop grands ($\geq 2^{30}$): le résultat est négatif.

Source : blog de Joshua Bloch, l'auteur de ce programme

Propriétés sur les programmes

Les propriétés qu'on veut assurer sur les programmes peuvent être très variées :

- ▶ Arrondis
- ▶ Arithmétique de pointeurs
- ▶ Index de tableaux
- ▶ Invariants de boucle
- ▶ Algorithmique
- ▶ Optimisation
- ▶ ...

Propriétés sur les programmes

Les outils mis en oeuvre le sont aussi :

- ▶ Revue croisée de code
- ▶ Tests
- ▶ Analyse statique, Model Checking
- ▶ Méthodes formelles en général

Mais il n'y a pas de recette magique adaptée à tout problème.

Méthodes formelles

Elles supposent:

- ▶ Une notion abstraite de propriété du code
- ▶ Une utilisation de l'ordinateur pour trouver voire **vérifier** les preuves de programmes.

Intérêts:

- ▶ On peut faire des vérifications massives.
- ▶ Pour les propriétés en question, la garantie est plus sûre que lorsque c'est un humain qui vérifie.
- ▶ On peut fournir du code accompagné de sa preuve de correction:
 - ▶ Code mobile
 - ▶ Pérennité

Preuves, programmes et machines

Ceci pose au moins trois problèmes :

- ▶ Comment représenter des énoncés et des preuves dans un ordinateur ?
- ▶ Comment faire confiance à la vérification faite par un ordinateur?
- ▶ Comment relier un programme à des spécifications (et des preuves) ?

Théorèmes et démonstrations comme objets

À la fin du XIX^{ème} siècle, la **logique** passe du statut d'outil philosophique à celui de branche mathématique à la faveur de la **crise des fondements**.

Begriffsschrift, Frege, 1872 :

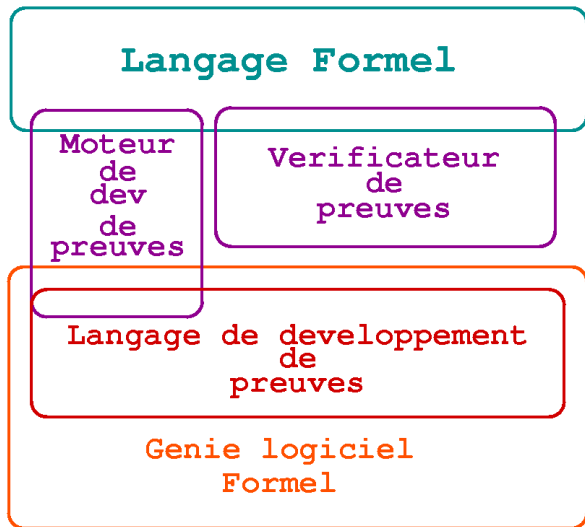
- ▶ La vérité mathématique devient vérifiable par des critères syntaxiques.
- ▶ Les preuves sont des arbres.

C'est l'essor de la métamathématique.

La mécanisation en pratique

- ▶ Le précurseur : *Automath*, N. G. de Bruijn, 1968 :
 - ▶ Premier programme informatique pour l'énoncé et la vérification de preuves formelles
 - ▶ Formalisation d'un ouvrage de référence d'analyse de Landau
- ▶ La voie est ouverte pour les **systèmes de preuve formelle** .

Systèmes de preuve formelle



La mécanisation en pratique, aujourd'hui

Une variété de systèmes, à vocations diverses :

- ▶ Théorie des ensembles :
 - ▶ Mizar (A. Trybulec, Pologne), <http://www.mizar.org>
 - ▶ Méthode B (J. R. Abrial, France),
<http://www.methode-b.com/>
- ▶ Higher Order Logic (HOL) :
 - ▶ HOL (M. Gordon, R.U.), <http://hol.sourceforge.net/>
 - ▶ HOL-Light (J.H. Harrison, R.U. /E. U.),
<http://www.cl.cam.ac.uk/~jrh13/hol-light/>
 - ▶ PVS (Owre, Rushby, Shankar, E.U),
<http://pvs.csl.sri.com/>
- ▶ Calcul des Constructions Inductives (CCI) :
 - ▶ Coq , <http://coq.inria.fr>

Et bien d'autres...

La mécanisation en pratique, aujourd'hui

- ▶ Modélisation, preuve et génération du code embarqué des lignes 14 (1998) puis 1 (2002) du métro parisien, avec la méthode B.
- ▶ Preuve complète du théorème des quatre couleurs en Coq (2004)
- ▶ Certification au niveau EAL7 des CC d'un produit commercial Gemalto basé sur JavaCard (2007)
- ▶ Certification d'un compilateur (d'un large sous-ensemble de) C pour PowerPC en Coq (2008)

Preuves et programmes

Une variété d'approches, parmi lesquelles :

- ▶ Annotation de code
- ▶ Utilisation d'un langage riche qui permette:
 - ▶ d'écrire des programmes
 - ▶ de les exécuter
 - ▶ d'écrire des propriétés, et leur preuves
 - ▶ de les vérifier
- ▶ Extraction de code

On peut se servir de Coq pour tout cela.

Un peu de Coq

Le Calcul des Constructions Inductives est une “théorie des types” :

- ▶ Un λ -calcul
- ▶ Un système de types sophistiqué
- ▶ L’isomorphisme de Curry-Howard

Un peu de Coq : le langage de programmation

- ▶ Un langage de programmation purement fonctionnel
- ▶ Un système de type plus riches
- ▶ Typage décidable
- ▶ Inférence de types
(mais c'est plus compliqué que Damas-Milner)

Un peu de Coq : les types sont des termes

Par exemple on peut écrire:

```
Fixpoint type_tuple (TS : list Type) : Type :=  
  match TS with  
  | nil => unit  
  | cons T TS' => T * (type_tuple TS')  
end.
```

Et alors :

```
tuple [nat; bool] = nat * (bool * unit)
```

Un peu de Coq : les types inductifs

Un peu comme en OCaml :

```
Inductive nat : Set := 0 : nat | S : nat -> nat.
```

on fait aussi des booléens, listes, des définitions par filtrage, ...

Et des récurrences “structurelles” :

```
Fixpoint plus (n m : nat) : nat :=  
  match n with  
  | 0 => m  
  | S p => S (plus p m)  
end.
```

Et des [preuves](#) par induction...

Un peu de Coq : les types dépendent de termes

Les listes de booléens de taille n :

```
Inductive vector : nat -> Type :=  
  | Vnil : vector 0  
  | Vcons : forall (a:A) (n:nat), vector n -> vector (S n).
```

Et la fonction de concaténation a le type :

$$\forall n m, \text{vector } n \rightarrow \text{vector } m \rightarrow \text{vector } (n + m)$$

Un peu de Coq : énoncés et preuves

L'isomorphisme de Curry-Howard :

- ▶ Un type est une proposition.
- ▶ Un théorème valide est un type habité.
- ▶ Un terme prouve donc le théorème exprimé par son type.
- ▶ Exemple : l'implication

Les commandes pour construire les preuves s'appellent des tactiques.

Un peu de Coq

À vous de jouer.