



UNIVERSITÉ
PARIS-SUD 11

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE



INRIA



centre de recherche
SACLAY - ÎLE-DE-FRANCE

L'équipe ProVal Responsable : Christine Paulin

*un focus sur
la preuve de programmes en virgule flottante*

Claude Marché

Accord INRIA Université Paris-Sud 11



Petit historique, et thèmes de recherche

LRI, équipe Demons

INRIA-projet Coq/Logical

Lyon, LIP

Univ. Paris 6

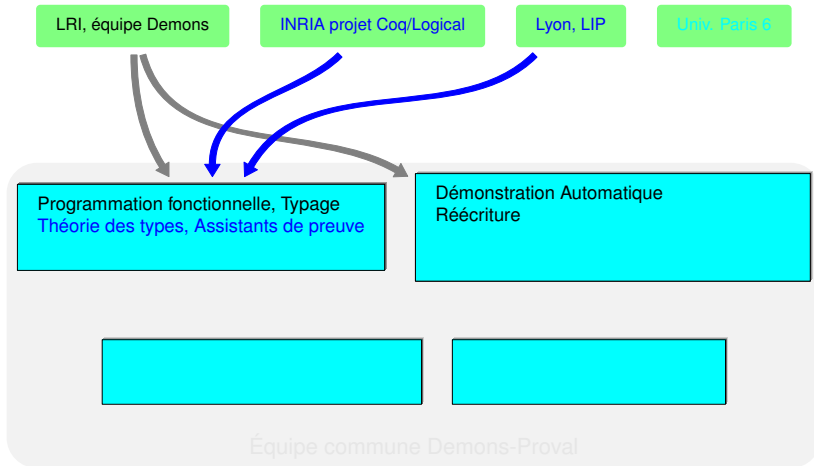
Programmation fonctionnelle, Typage

Démonstration Automatique
Réécriture

Équipe commune Demons-Proval

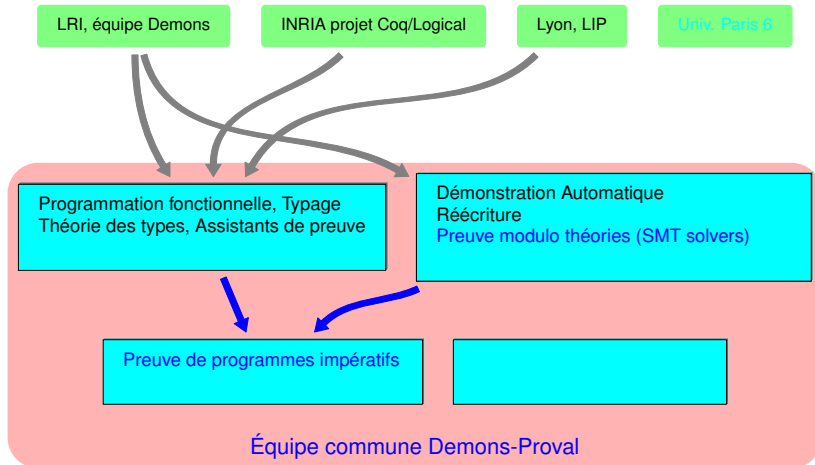


Petit historique, et thèmes de recherche

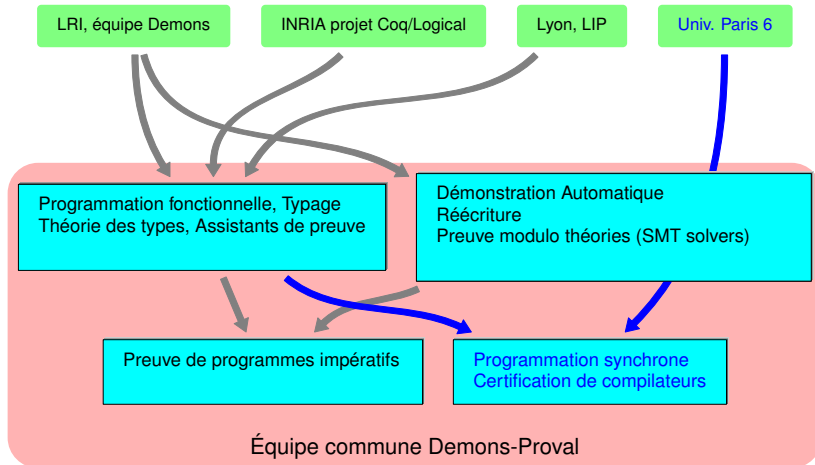




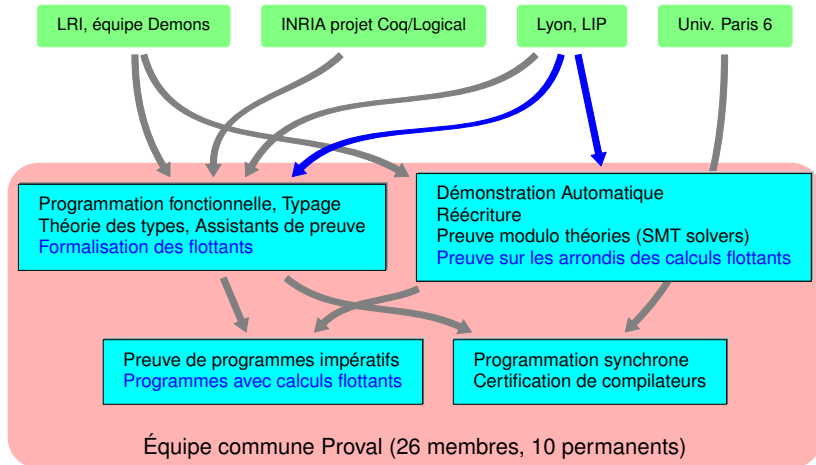
Petit historique, et thèmes de recherche



Petit historique, et thèmes de recherche



Petit historique, et thèmes de recherche



Motivations, contexte

Thème Preuves de programmes impératifs

- ▶ langages C et Java
- ▶ codes équipés de spécifications formelles
- ▶ domaine d'application : codes critiques
 - ▶ programmes de taille modeste
 - ▶ besoin de confiance très élevée

System@tic PFC

ANR CAT

Airbus France
Dassault Aviation
...

ANR U3CAT

Thème transversal : virgule flottante

- ▶ codes d'analyse numérique
- ▶ codes embarqués
- ▶ analyse du binaire compilé

ANR FOST

CEA List

Digitéo Hisseo

Exemple illustratif

Approximation polynomiale d'une fonction transcendante

- ▶ cosinus sur un intervalle autour de zéro

```
float my_cosine(float x) {  
    return 1.0f - x * x * 0.5f;  
}
```

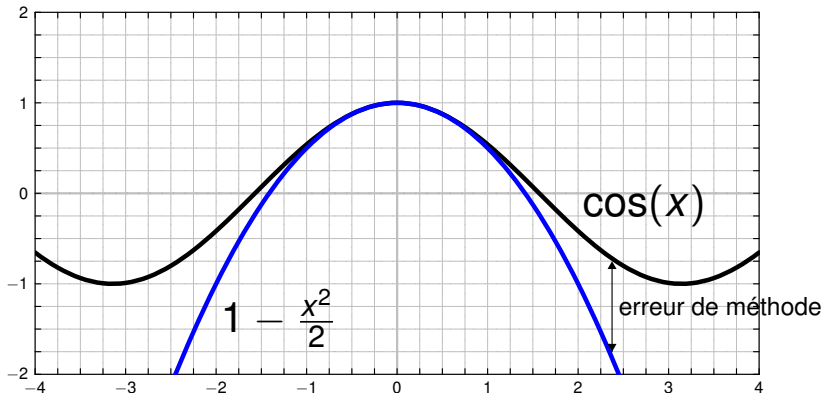

Exemple illustratif

Approximation polynomiale d'une fonction transcendante

- ▶ cosinus sur un intervalle autour de zéro

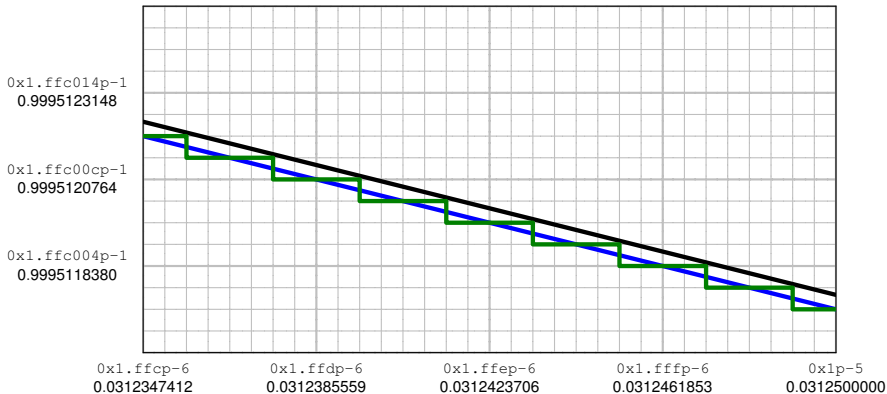
```
/*@ requires \abs(x) <= ??? ;  
   @ ensures \abs(\result - \cos(x)) <= ???;  
   @*/  
float my_cosine(float x) {  
    return 1.0f - x * x * 0.5f;  
}
```

Visualisation graphique





Visualisation : focus sur $[1/32 - \varepsilon, 1/32]$



bleu - noir : erreur de méthode

vert - bleu : erreur d'arrondi

vert - noir : erreur totale

Retour sur le code annoté

```
/*@ requires \abs(x) <= 0x1p-5; //x ∈ [-1/32,1/32]
   @ ensures \abs(\result - \cos(x)) <= 0x1p-23;
   @           // erreur totale
   @*/
float my_cosine(float x) {

    return 1.0f - x * x * 0.5f;
}
```

Retour sur le code annoté

```
/*@ requires \abs(x) <= 0x1p-5; //x ∈ [-1/32,1/32]
   @ ensures \abs(\result - \cos(x)) <= 0x1p-23;
   @           // erreur totale
   @*/
float my_cosine(float x) {
    //@ assert \abs(1.0 - x*x*0.5 - \cos(x)) <= 0x1p-24;
           // erreur de méthode
    return 1.0f - x * x * 0.5f;
}
```

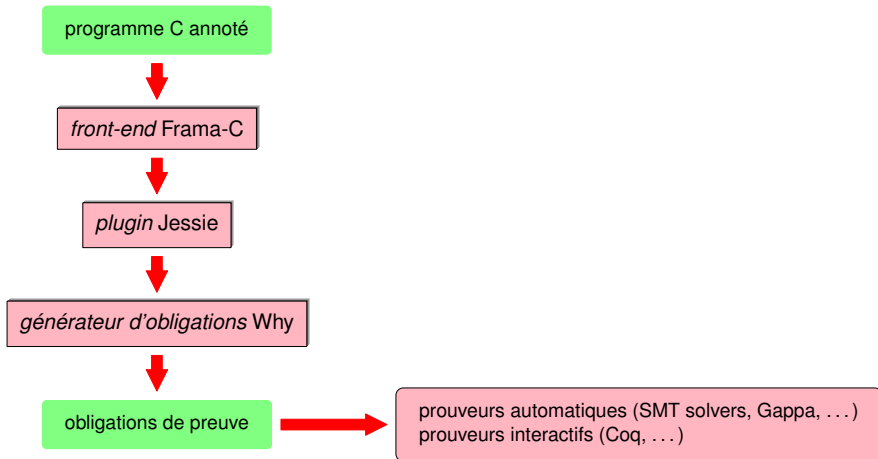
Retour sur le code annoté

```
/*@ requires \abs(x) <= 0x1p-5; //x ∈ [-1/32,1/32]
   @ ensures \abs(\result - \cos(x)) <= 0x1p-23;
   @           // erreur totale
   @*/
float my_cosine(float x) {
    //@ assert \abs(1.0 - x*x*0.5 - \cos(x)) <= 0x1p-24;
           // erreur de méthode
    return 1.0f - x * x * 0.5f;
}
```

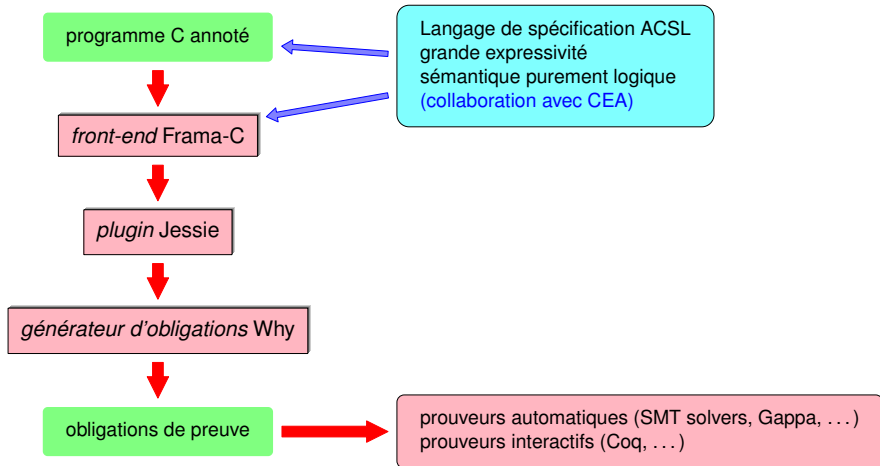
Dans les annotations :

- ▶ +, -, * opèrent sur les **nombre**s réels
- ▶ **Pas d'arrondi, pas d'overflow** dans les annotations

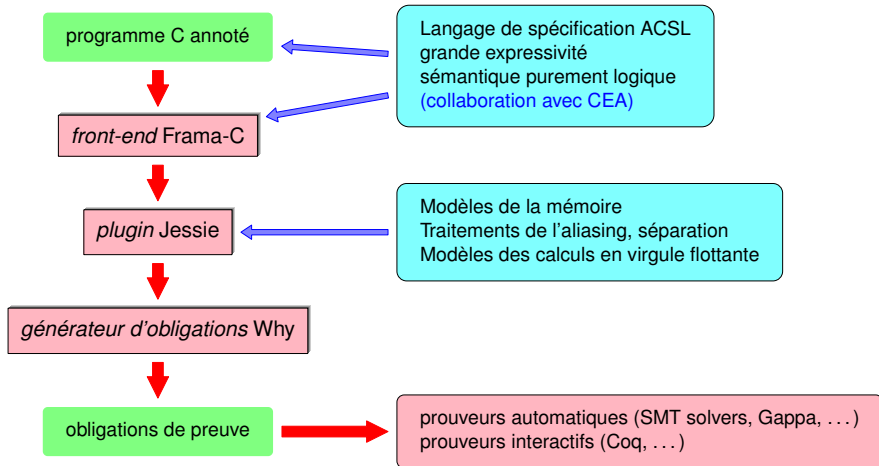
Démarche de preuve



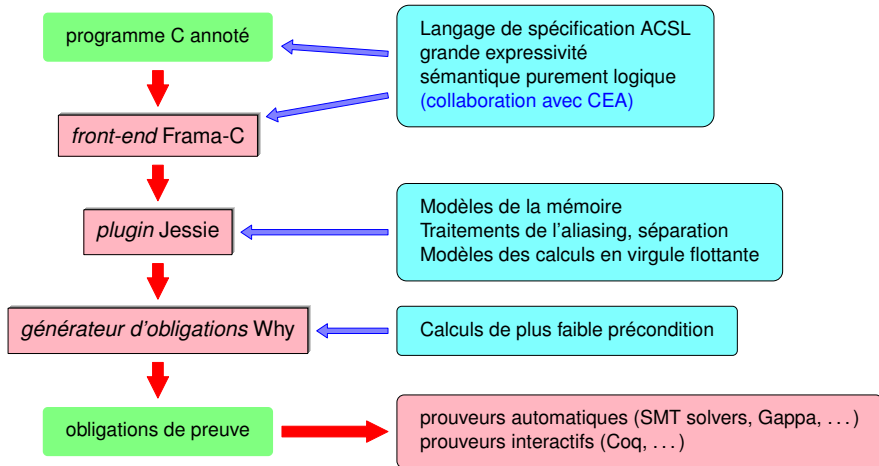
Démarche de preuve



Démarche de preuve



Démarche de preuve



Preuve des obligations

gWhy: a verification conditions viewer

File Configuration Proof

Proof obligations	Alt-Ergo 0.8	Simplify 1.5.4	Z3 1.3 (SS)	CVC3 devel (SS)	Gappa 0.11.2	Statis
Function my_cosine Default behavior	✗	✗	✗	✗	✗	1/2
1. assertion	▽	✂	✂	▽	▽	
2. postcondition	✂	✂	✂	▽	●	
Function my_cosine Safety	✗	✗	✗	✗	✓	5/5
1. check FP overflow	●	✂	●	●	●	
2. check FP overflow	▽	✂	✂	▽	●	
3. check FP overflow	●	✂	●	●	●	
4. check FP overflow	▽	✂	✂	▽	●	
5. check FP overflow	▽	✂	✂	▽	●	

```

my_cosine_safety_po_2
x: gen_float
H1: abs_real(float_value(x)) <= 0x1.p-5
H2: abs_real((1.0 - float_value(x) * float_value(x) * cos(float_value(x)))) <= 0x1.p-5
H3: no_overflow(Single, nearest_even, 1.0)
result: gen_float
H4: gen_float_of_real_post(Single, nearest_even, 1.0)

no_overflow(Single, nearest_even, float_value(x))

/*@ requires \abs(x) <= 0x1p-5 ;
   @ ensures \abs(\result - \cos(x)) <= 0x1.p-5 ;
   */
float my_cosine(float x) {
  //@ assert \abs(1.0 - x*x*0.5 - \cos(x)) <= 0x1.p-5 ;
  return 1.0f - x * x * 0.5f;
}
    
```

Timeout: 9 Pretty Printer file: mycos.c VC: check FP overflow

Preuve des obligations

Prouveurs de l'équipe

gWhy: a verification

File Configuration Proof

Proof obligations	Alt-Ergo 0.8	Simplify 1.5.4	Z3 1.3 (SS)	CVC3 dev (SS)	Gappa 0.11.2	Statis
Function my_cosine Default behavior	✗	✗	✗	✗	✗	1/2
1. assertion	▽	✂	✂	▽	▽	
2. postcondition	✂	✂	✂	▽	●	
Function my_cosine Safety	✗	✗	✗	✗	✓	5/5
1. check FP overflow	●	✂	●	●	●	
2. check FP overflow	▽	✂	✂	▽	●	
3. check FP overflow	●	✂	●	●	●	
4. check FP overflow	▽	✂	✂	▽	●	
5. check FP overflow	▽	✂	✂	▽	●	

```

my_cosine_safety_po_2
x: gen_float
H1: abs_real(float_value(x)) <= 0x1.p-5
H2: abs_real((1.0 - float_value(x) * float_value(x) * cos(float_value(x)))) <= 0x1.p-5
H3: no_overflow(Single, nearest_even, 1.0)
result: gen_float
H4: gen_float_of_real_post(Single, nearest_even, 1.0)

no_overflow(Single, nearest_even, float_value(x))

/*@ requires \abs(x) <= 0x1p-5 ;
   @ ensures \abs(\result - \cos(x)) <= 0x1p-5 ;
   */
float my_cosine(float x) {
  //@ assert \abs(1.0 - x*x*0.5 - \cos(x)) <= 0x1p-5 ;
  return 1.0f - x * x * 0.5f;
}

```

Timeout: 9 Pretty Printer file: mycos.c VC: check FP overflow

Preuve des obligations

Prouveurs de l'équipe

Proof obligations	Alt-Ergo 0.8	Simplify 1.5.4	Z3 1.3 (SS)	CVC3 dev (SS)	Gappa 0.11.2	Statis
Function my_cosine Default behavior	✗	✗	✗	✗	✗	1/2
1. assertion	▽	✂	✂	▽	▽	
2. postcondition	✂	✂	✂	▽	●	
Function my_cosine Safety	✗	✗	✗	✗	✓	5/5
1. check FP over						
2. check FP over						
3. check FP overflow	●	✂	●	●	●	
4. check FP overflow	▽	✂	✂	▽	●	
5. check FP overflow	▽	✂	✂	▽	●	

Erreur de méthode non prouvée !

```

my_cosine_safety_po_2
x: gen_float
H1: abs_real(float_value(x)) <= 0x1.p-5
H2: abs_real((1.0 - float_value(x) * float_value(x) * cos(float_value(x)))) <= 0x1.p-5
H3: no_overflow(Single, nearest_even, 1.0)
result: gen_float
H4: gen_float_of_real_post(Single, nearest_even, float_value(x))

/*@ requires \abs(x) <= 0x1p-5 ;
   @ ensures \abs(\result - \cos(x)) <= 0x1.p-5 ;
   */
float my_cosine(float x) {
  //@ assert \abs(1.0 - x*x*0.5 - \cos(x)) <= 0x1.p-5 ;
  return 1.0f - x * x * 0.5f;
}
  
```

Timeout: 9 | Pretty Printer | file: mycos.c VC: check FP overflow



Et l'obligation non prouvée ?

CoqIde

File Edit Navigation Try Tactics Templates Queries Display Compile Windows Help

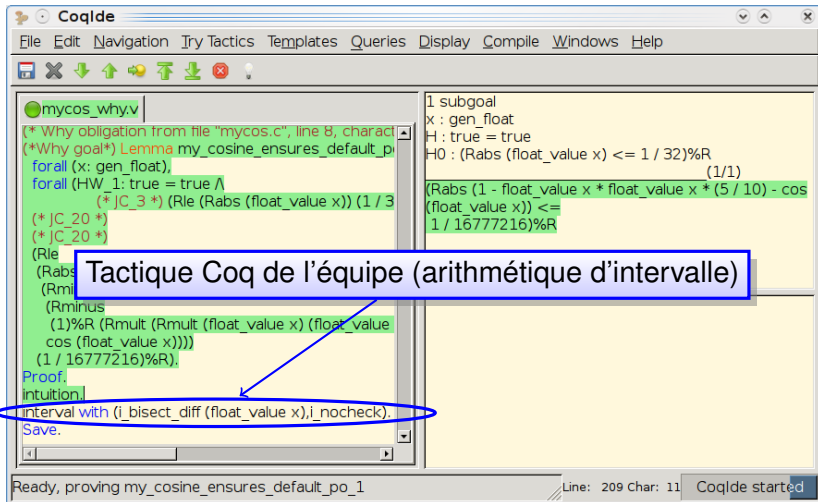
mycos_why.v

```
(* Why obligation from file "mycos.c", line 8, charac
(*Why goal*) Lemma my_cosine_ensures_default_p
forall (x: gen_float),
forall (HW_1: true = true  $\wedge$ 
(*JC_3 *) (Rle (Rabs (float_value x)) (1 / 3
(* JC_20 *)
(* JC_20 *)
(Rle
(Rabs
(Rminus
(Rminus
(1)%R (Rmult (Rmult (float_value x) (float_value
cos (float_value x))))
(1 / 16777216)%R).
Proof.
intuition.
interval with (i_bisect_diff (float_value x),i_nocheck).
Save.
```

1 subgoal
x : gen_float
H : true = true
H0 : (Rabs (float_value x) <= 1 / 32)%R
(1/1)
(Rabs (1 - float_value x * float_value x * (5 / 10) - cos
(float_value x)) <=
1 / 16777216)%R

Ready, proving my_cosine_ensures_default_po_1 Line: 209 Char: 11 CoqIde started

Et l'obligation non prouvée ?



CoqIDE

File Edit Navigation Try Tactics Templates Queries Display Compile Windows Help

```

mycos_why.v
(* Why obligation from file "mycos.c", line 8, character 1 *)
(* Why goal *) Lemma my_cosine_ensures_default_po_1 :
forall (x : gen_float),
forall (HW_1: true = true /\
      (* JC_3 *) (Rle (Rabs (float_value x)) (1 / 3
      (* JC_20 *)
      (* JC_20 *)
      (Rle
      (Rabs
      (Rmin
      (Rminus
      (1)%R (Rmult (Rmult (float_value x) (float_value
      cos (float_value x))))
      (1 / 16777216)%R).
Proof.
intuition.
interval with (i_bisect_diff (float_value x),i_nocheck).
Save.

```

1 subgoal
x : gen_float
H : true = true
H0 : (Rabs (float_value x) <= 1 / 32)%R
(Rabs (1 - float_value x * float_value x * (5 / 10) - cos
(float_value x)) <=
1 / 16777216)%R

Tactique Coq de l'équipe (arithmétique d'intervalle)

Ready, proving my_cosine_ensures_default_po_1 Line: 209 Char: 11 CoqIDE started

Un élément de perspective

Un axe émergent : **Certification** des méthodes et des outils

- ▶ Qualification DO-178B de Alt-Ergo (Airbus France)
- ▶ Production de traces de preuves vérifiables
- ▶ Certification formelle
 - ▶ Compilateur SCADE pour le synchrone
 - ▶ Chaîne Frama-C/Jessie/Why