HAL
open science

Les assistants de preuve, ou comment avoir confiance en ses démonstrations.

Julien Narboux

To cite this version:

Julien Narboux. Les assistants de preuve, ou comment avoir confiance en ses démonstrations.. Séminaire L, Mar 2013, Strasbourg, France. hal-00809448

HAL Id: hal-00809448

https://inria.hal.science/hal-00809448

Submitted on 9 Apr 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Les assistants de preuve

ou comment avoir confiance en ses démonstrations

Julien Narboux

UNIVERSITÉ DE STRASBOURG

Séminaire L, Avril 2013

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

Exemples de bugs fameux

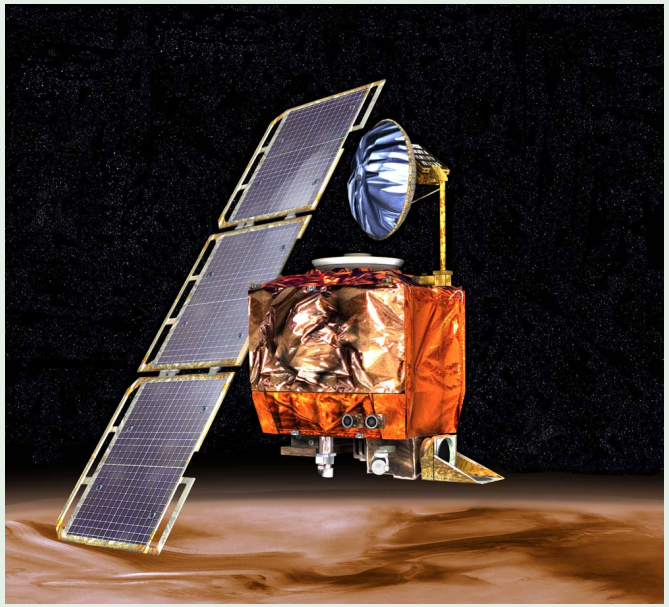
Ariane Vol 501

"Because of the different flight path, a data conversion from a 64-bit floating point to 16-bit signed integer value caused a hardware exception (more specifically, an arithmetic overflow, as the floating point number had a value too large to be represented by a 16-bit signed integer). Efficiency considerations had led to the disabling of the software handler (in Ada code) for this error trap, although other conversions of comparable variables in the code remained protected. This caused a cascade of problems, culminating in destruction of the entire flight."

Mars Climate Orbiter

"The 'root cause' of the loss of the spacecraft was the failed translation of English units into metric units in a segment of ground-based, navigation-related mission software."

Coût : $327 600 000.



Un logiciel pour corriger les bugs ?

Problème de l'arrêt

Il n'existe pas de programme permettant de décider si un programme termine ou pas.

Première idée

Tester !

On écrit une fonction:

Programme + Propriété $\rightarrow$ Vrai/Faux

Première idée

Tester !
On écrit une fonction:
Programme + Propriété $\rightarrow$ Vrai/Faux

Mais çà ne suffit pas !

Problème

On a une infinité de cas.

Solution

Réaliser une preuve.

Un raisonnement fini pour traiter une infinité de cas.

Oui mais. . .

Théorème de Fermat-Wiles

Il n'existe pas de nombres entiers non nuls , et tels que :

$$x^n + y^n = z^n$$

dès que est un entier strictement supérieur à 2.

- Paul Wolfskehl offre 100,000 marks
- deux conditions: relues par les pairs, attendre 2 ans
- 1907-1908 : 621 tentatives

Autre exemple récent

In the paper: Roman Mikhailov, Jie Wu On homotopy groups of the suspended classifying spaces Algebraic/ Geometric Topology, 2010, vol.10, pp.565–625. the authors state in Theorem 5.4: Let A4 be the 4-th alternating group. Then $\pi_4(\Sigma K(A4,1)) = Z/4$. *The elementary method used by the Kenzo program, known as the Whitehead tower, produces a different result, namely* $\pi_4(\Sigma K(A4,1)) = Z/12$. *The authors of the quoted paper inadvertently forgot the 3-primary component. This Kenzo computation was done by Ana Romero, using extra-modules devoted to group resolutions written by herself.*

Qu'est-ce qu'une preuve ?

- un argument convainquant

Qu'est-ce qu'une preuve ?

- un argument convainquant
- une suite de déductions à partir des axiomes

Qu'est-ce qu'une preuve ?

- un argument convainquant
- une suite de déductions à partir des axiomes
- un algorithme (correspondance de Curry-Howard)

Oui mais...

Il peut être difficile de se convaincre qu'une preuve est correcte :

1. présence de calculs non vérifiables à la main

Oui mais...

Il peut être difficile de se convaincre qu'une preuve est correcte :

1. présence de calculs non vérifiables à la main
2. preuves très longues, très compliquées

Oui mais...

Il peut être difficile de se convaincre qu'une preuve est correcte :

1. présence de calculs non vérifiables à la main
2. preuves très longues, très compliquées
3. trop de détails techniques, trop de cas pour les traiter à la main sans faire d'erreurs

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

Une quête de la rigueur

1. Clarifier les *hypothèses*

Une quête de la rigueur

1. Clarifier les *hypothèses*
2. Clarifier ce qu'est une *preuve*

Une quête de la rigueur

1. Clarifier les *hypothèses*
2. Clarifier ce qu'est une *preuve*
3. Être si précis que l'on a plus besoin de *comprendre* la preuve pour la *vérifier*

Une quête de la rigueur

1. Clarifier les *hypothèses*
2. Clarifier ce qu'est une *preuve*
3. Être si précis que l'on a plus besoin de *comprendre* la preuve pour la *vérifier*
4. Automatiser des preuves

Par définition vérifier qu'une preuve est correcte est un problème *décidable*.

On peut donc construire des assistants de preuve.

Exemples

- Coq
- Isabelle
- PVS
- HOL-Light
- ...

Qu'est-ce que Coq ?

- un assistant de preuve
- développé et distribué librement par Inria

Il permet de :

- définir des notions mathématiques et/ou des programmes
- démontrer mécaniquement des théorèmes mathématiques mettant en jeu ces définitions

Processus de preuve

Les deux étapes du développement d'une démonstration dans Coq sont les suivantes :

- d'abord la construction *interactive* d'une démonstration *par l'utilisateur*;
- ensuite la vérification *automatique* de la correction de démonstration *par le système*.

L'utilisateur prouve, puis le système vérifie que la preuve est bien correcte.

Exemples d'énoncés mathématiques

- "Si les portes sont ouvertes c'est que la rame est en face d'un quai".
- $$\sum_{0}^{n} i = \frac{n(n+1)}{2}$$

Démo

On veut montrer que:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Démo

On veut montrer que:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

On va montrer que:

$$2 * \sum_{i=0}^{n} i = n(n+1)$$

Démo

On veut montrer que:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

On va montrer que:

$$2 * \sum_{i=0}^{n} i = n(n+1)$$

En Coq:

```
Lemma sun_n : forall n:nat,  2 * (sum_int n) =  n*(n+1).
```

Démo

On veut montrer que:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

On va montrer que:

$$2 * \sum_{i=0}^{n} i = n(n+1)$$

En Coq:

```
Lemma sun_n : forall n:nat,  2 * (sum_int n) =  n*(n+1).
```

Oui mais comment est définie l'opération $\sum$?

Démo

On veut montrer que:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

On va montrer que:

$$2 * \sum_{i=0}^{n} i = n(n+1)$$

En Coq:

```
Lemma sun_n : forall n:nat,  2 * (sum_int n) =  n*(n+1).
```

Oui mais comment est définie l'opération $\sum$?

$$\sum_{i=0}^{0} i = 0$$

$$\sum_{i=0}^{n} i = n + \sum_{i=0}^{n-1} i$$

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

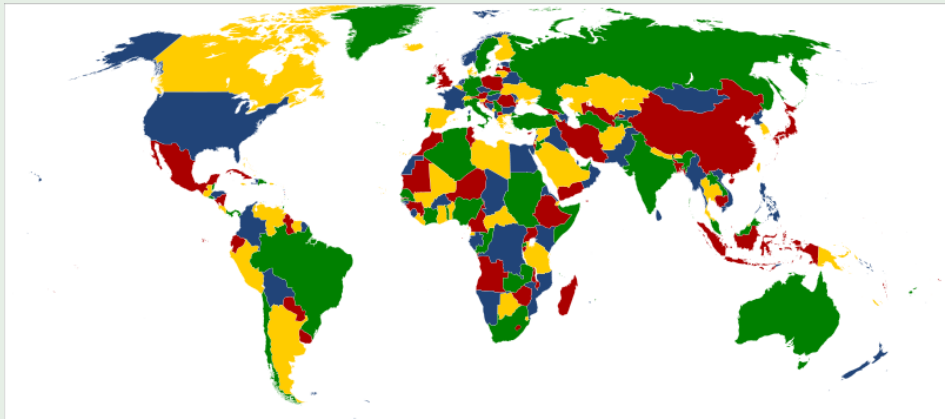
Le problème de la vérification d'une preuve

Présence de calculs

Théorème des 4 couleurs

Quatre couleurs suffisent pour colorier une carte géographique *plane* sans que deux pays ayant une *frontière* en commun ne soient de la même couleur.

- 1879 Preuve fausse par Kempe
- 1890 Heaywood trouve l'erreur
- 1976 Appel and Hake (1478 configurations, 1200 heures de calcul)
- 2004 Formalisation en Coq par Gonthier et Werner



Le problème de la vérification d'une preuve

Présence de calculs

Conjecture de Kepler/Théorème de Hales

Pour un empilement de sphères égales, la densité maximale est atteinte pour un empilement cubique à faces centrées.

- 1998 Preuve par Thomas Hales
- 2004 - ? Projet Flyspeck: formalisation du théorème en cours en HOL-light avec des contributions en Coq et Isabelle (plus de 300000 lignes)

Photo par Robert Cudmore

Robert MacPherson, éditeur, écrit que:

> *"The news from the referees is bad, from my perspective. They have not been able to certify the correctness of the proof, and will not be able to certify it in the future, because they have run out of energy to devote to the problem. This is not what I had hoped for. The referees put a level of energy into this that is, in my experience, unprecedented. "*

Le problème de la vérification d'une preuve

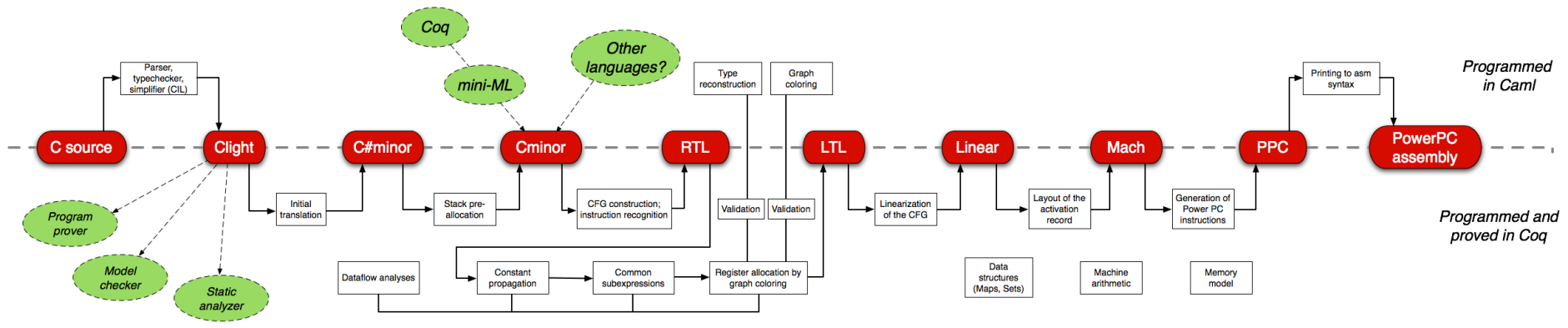
Trop de détails techniques

Un compilateur

CompCert un compilateur C prouvé formellement.

Génère de l'assembleur PowerPC et ARM à partir de code C.

Preuve formelle de la correction: le code assembleur se comporte de la même manière que le code source.



Le problème de la vérification d'une preuve

Trop de détails techniques

Système de pilotage d'une ligne de métro

- Méthode B
- Paris (ligne 14, 1998), Paris (ligne 1, 2005), Lyon (ligne D), ...

Le problème de la vérification d'une preuve

Trop de détails techniques

Un système d'exploitation

sel4 : Micro kernel prouvé en Isabelle/HOL.
Preuve: 165000 lignes, 11 années/homme.
Code: 15000 lignes, 2.5 années/homme.

Le problème de la vérification d'une preuve I

La taille de la preuve

Théorème de Feit-Thompson

```
Theorem Feit_Thompson (gT:finGroupType) (G:{group gT}):
odd ##|G| -> solvable G.
```

Preuve en Coq par Georges Gonthier et son équipe (septembre 2012)[a]:
170 000 lignes, 15 000 définitions, 4 200 théorèmes

[a] http://ssr2.msr-inria.inria.fr/~jenkins/current/progress.html

Autres exemples

http://www.cs.ru.nl/~freek/100/index.html

Le problème de la vérification d'une preuve

Trop de détails techniques

Un système de gestion de cartes à puces

Gemalto: preuve formelle avec Coq.
Certification d'un système JavaCard au niveau EAL7.

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

Correspondance de Curry-Howard I

$$\frac{f : string \to int \quad a : string}{int}$$

Correspondance de Curry-Howard I

$$\frac{f : string \to int \quad a : string}{f(a) : int}$$

Correspondance de Curry-Howard I

$$\frac{f : string \rightarrow int \quad a : string}{f(a) : int}$$

Règle de typage de l'application:

$$\frac{f : A \rightarrow B \quad a : A}{f(a) : B}$$

Correspondance de Curry-Howard I

$$\frac{f : string \to int \quad a : string}{f(a) : int}$$

Règle de typage de l'application:

$$\frac{f : A \to B \quad a : A}{f(a) : B}$$

Modus ponens:

$$\frac{A \Rightarrow B \quad A}{B}$$

Correspondance de Curry-Howard I

$$\frac{f : string \to int \quad a : string}{f(a) : int}$$

Règle de typage de l'application:

$$\frac{A \to B \qquad A}{B}$$

Modus ponens:

$$\frac{A \Rightarrow B \quad A}{B}$$

Correspondance de Curry-Howard I

logique	programmation
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$	$\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\text{fun } x : A \mapsto t) : A \to B}$
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}$

Correspondance de Curry-Howard I

logique	programmation
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$	$\dfrac{\Gamma, \quad A \vdash \quad B}{\Gamma \vdash \quad A \to B}$
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash \quad A \to B \quad \Gamma \vdash \quad A}{\Gamma \vdash \quad B}$

Correspondance de Curry-Howard I

logique	programmation
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	$\dfrac{\Gamma, \quad A \vdash \quad B}{\Gamma \vdash \quad A \to B}$
$\dfrac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash \quad A \to B \quad \Gamma \vdash \quad A}{\Gamma \vdash \quad B}$

Correspondance de Curry-Howard II

$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	$\dfrac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) :}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$	
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$	

Correspondance de Curry-Howard II

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

Correspondance de Curry-Howard II

$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	$\dfrac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$	$\dfrac{\Gamma \vdash t : A \times B}{\Gamma \vdash A}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash t : A \times B}{\Gamma \vdash B}$

Correspondance de Curry-Howard II

$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	$\dfrac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$	$\dfrac{\Gamma \vdash t : A \times B}{\Gamma \vdash \mathit{fst}\ t : A}$
$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash t : A \times B}{\Gamma \vdash \mathit{snd}\ t : B}$

Correspondance de Curry-Howard III

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash \mathit{inl}\ a : A + B}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathit{inr}\ b : A + B}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

$$\frac{\Gamma \vdash m : A \vee B \quad \Gamma, x : A \vdash t : C \quad \Gamma, x : B \vdash u : C}{\Gamma \vdash \mathit{case}\ m\ \mathit{of}\ \mathit{inl}(a) => t \mid \mathit{inr}(a) => u : C}$$

Règles de simplification

$$
\begin{array}{rcl}
fst(A, B) & = & A \\
snd(A, B) & = & B \\
case\ (inl\ m)\ of\ inl(a) => t | inr(a) => u & = & t[x := m] \\
case\ (inr\ m)\ of\ inl(a) => t | inr(a) => u & = & u[x := m]
\end{array}
$$

Correspondance de Curry-Howard

Logique	**λ-calcul/programmation**
formule	type
preuve	terme/programme
vérification d'une démonstration	vérification de type
normalisation des preuves	β-réduction/calcul

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

Défis

- Comment décrire les preuves ?
- Comment formaliser ?
- Comment automatiser les preuves à grande échelle ou à petite échelle ?
- Comment échanger des preuves ?
- ...

Conclusion

- Les assistants de preuve sont utiles en mathématiques et en informatique.
- Ils permettent de manipuler la notion de preuve.
- Attention c'est addictif !

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

Pub

Une bourse de thèse fléchée sur le preuve formelle appliquée à l'enseignement en 2013.

`http://dpt-info.u-strasbg.fr/~narboux/stages.html`

Table des matières

1. Le problème
2. Les assistants de preuve
3. Quelques succés
4. Curry-Howard
5. Conclusion
6. Pub
7. Si on a le temps

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et
- au compilateur et

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et
- au compilateur et
- au microprocesseur et

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et
- au compilateur et
- au microprocesseur et
- à vos définitions et énoncés et

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et
- au compilateur et
- au microprocesseur et
- à vos définitions et énoncés et
- à vos axiomes.

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et
- au compilateur et
- au microprocesseur et
- à vos définitions et énoncés et
- à vos axiomes.

Qui vérifie le vérificateur ?

Il faut faire confiance à:

- la théorie sous jacente à l'assistant de preuve et
- que l'implantation correspond bien à la théorie et
- au compilateur et
- au microprocesseur et
- à vos définitions et énoncés et
- à vos axiomes.

Critère de de Bruijn

- Les preuves sont certifiée par un *noyau*.
  - Isabelle (très petit)
  - HOL (très petit)
  - Coq (assez petit)

En revanche, ni Mizar, ni PVS n'ont une notion de noyau.

Et l'automatisation ?

Un approche sceptique: on ne fait pas confiances aux démonstrateurs automatiques.
Deux solutions:

1. Prouver le prouveur.
2. Vérifier le résultat du prouveur: un certificat.

Exemples

- Egalité entre deux expressions prises dans un anneau ou dans un corps.
- Tautologies.
- Inégalités linéaires.
- Théorèmes en géométrie.
- ...