

Les assistants de preuve et applications à l'apprentissage du raisonnement mathématique

Julien Narboux

Unité de formation et de recherche

de **mathématique** et d'**informatique**

Université de Strasbourg

Toulouse

Table des matières

1 Exemples d'erreurs

- En informatique
- En mathématiques

2 Les assistants de preuve

- Qu'est-ce qu'une preuve ?
- Langages de preuve
- Quelques résultats obtenus

3 L'enseignement et les assistants de preuve

4 Compléments

- Qui vérifie le vérificateur ?
- Automatisation
- Curry-Howard

Table des matières

1 Exemples d'erreurs

- En informatique
- En mathématiques

2 Les assistants de preuve

- Qu'est-ce qu'une preuve ?
- Langages de preuve
- Quelques résultats obtenus

3 L'enseignement et les assistants de preuve

4 Compléments

- Qui vérifie le vérificateur ?
- Automatisation
- Curry-Howard

Exemples de bugs fameux

Ariane Vol 501 (1996)

"The software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. [...] The value was much higher than expected because the early part of the trajectory of Ariane 5 differs from that of Ariane 4 and results in considerably higher horizontal velocity values."



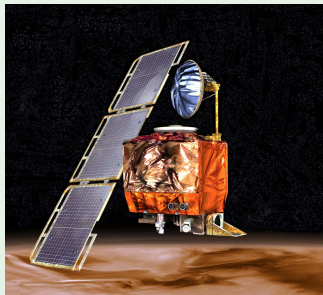
Ariane 501 Inquiry Board report:

<http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf>

Mars Climate Orbiter (1999)

"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 exemple élémentaire

Exercice: écrire une programme qui retourne la valeur absolue d'un entier x^a .

```
int abs(int x)
{
  int z;
  if (x<0)
    z=-x;
  else
    z=x;
  return z;
}
```

^aSource: David Mentré

Un exemple élémentaire

Exercice: écrire un programme qui retourne la valeur absolue d'un entier x^a .

```
int abs(int x)
{
  int z;
  if (x<0)
    z=-x;
  else
    z=x;
  return z;
}
```

^aSource: David Mentré

Ce programme est faux !

Si $x = -2^{31}$, 2^{31} n'existe pas.

Table des matières

1 Exemples d'erreurs

- En informatique
 - Dans des systèmes complexes. . .
 - . . . mais aussi dans votre premier programme
- En mathématiques
 - Des exemples historiques
 - Des exemples au collège
 - Un exemple récent
 - et dans les vérificateurs de preuves !

2 Les assistants de preuve

- Qu'est-ce qu'une preuve ?
- Langages de preuve
- Quelques résultats obtenus

3 L'enseignement et les assistants de preuve

4 Compléments

- Qui vérifie le vérificateur ?
- Automatisation
- Curry-Howard

Le géométrie est centrale dans l'histoire des preuves

Euclide (–325–265) *Les éléments.*

La méthode axiomatique

Hilbert (1862-1943) *Die Grundlagen der Geometrie.*

Les mathématiques formelles

Tarski (1902-1983) *Metamathematische Methoden in der Geometrie.*

Automatisation,
axiomatisation



Le géométrie est centrale dans l'histoire des preuves

Euclide (–325–265) *Les éléments*.

La méthode axiomatique

Hilbert (1862-1943) *Die Grundlagen der Geometrie*.

Les mathématiques formelles

Tarski (1902-1983) *Metamathematische Methoden in der Geometrie*.

Automatisation,
axiomatisation



Le géométrie est centrale dans l'histoire des preuves

Euclide (–325–265) *Les éléments.*

La méthode axiomatique

Hilbert (1862-1943) *Die Grundlagen der Geometrie.*

Les mathématiques formelles

Tarski (1902-1983) *Metamathematische Methoden in der Geometrie.*

Automatisation,
axiomatisation



Des preuves incomplètes:

Les éléments. La première construction suppose l'existence de l'intersection entre deux cercles donnés.

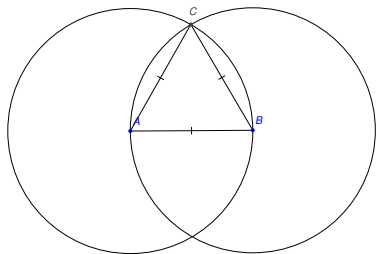
Die Grundlagen der Geometrie. Certaines preuves non triviales sont présentées comme évidentes dans les premières éditions.

Exemple de proposition

Proposition (Livre I, Prop. 1)

Soient A et B deux points, construire un triangle équilatéral ayant pour base AB .

Preuve: Soient C_1 et C_2 les cercles de centres A et B et de rayon AB . Soit C l'intersection de C_1 et C_2 . La distance AB est égale à la distance AC et la distance AB est égale à la distance BC . Donc ABC est équilatéral.



Imprécisions dans les Éléments

Problème

Les postulats d'Euclide ne permettent pas de déduire que C existe.

- Hilbert a proposé un système axiomatique pour combler les lacunes dans les Éléments, les preuves ne sont pas les mêmes.
- Avigad, Dean et Mumma ont proposé une axiomatique alternative qui permet de justifier les preuves originales ¹.

¹avigad`formal`2009

Mais aussi une longue succession de preuves fausses du 5ième postulat d'Euclide:

En 1763, dans sa thèse Klügel une liste de 30 preuves fausses du postulat des parallèles.

- Ptolémée admet implicitement l'unicité des parallèles.
- Proclus admet implicitement que deux droites sont parallèles, toute droite qui intersecte l'une intersecte l'autre.
- Legendre a publié plusieurs preuves fausses du 5ième postulat dans le 'best-seller' "Éléments de géométrie".
- ...

Postulat du triangle

$$\hat{A} + \hat{B} + \hat{C} = 180^\circ$$



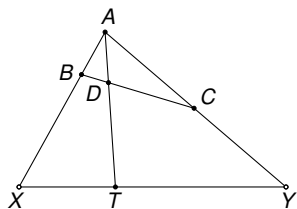
Adrien-Marie Legendre
(caricature de
Julien Léopold Boilly)

Postulat du triangle

“Il n'en est pas moins certain que le théorème sur la somme des trois angles du triangle doit être regardé comme l'une de ces vérités fondamentales qu'il est impossible de contester [...].”



Adrien-Marie Legendre
(caricature de
Julien Léopold Boilly)



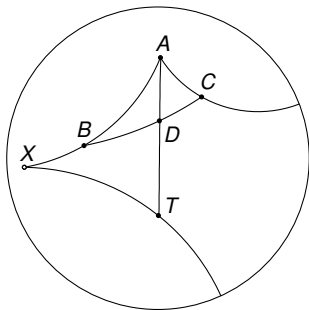


Table des matières

1 Exemples d'erreurs

- En informatique
 - Dans des systèmes complexes...
 - ... mais aussi dans votre premier programme
- En mathématiques
 - Des exemples historiques
 - Des exemples au collège
 - Un exemple récent
 - et dans les vérificateurs de preuves !

2 Les assistants de preuve

- Qu'est-ce qu'une preuve ?
- Langages de preuve
- Quelques résultats obtenus

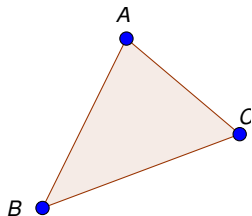
3 L'enseignement et les assistants de preuve

4 Compléments

- Qui vérifie le vérificateur ?
- Automatisation
- Curry-Howard

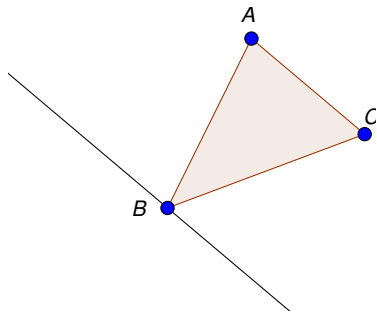
La somme des angles d'un triangle

On construit une parallèle à AC passant par B .



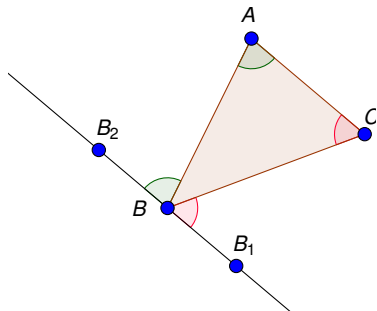
La somme des angles d'un triangle

On construit une parallèle à AC passant par B .



La somme des angles d'un triangle

On construit une parallèle à AC passant par B .

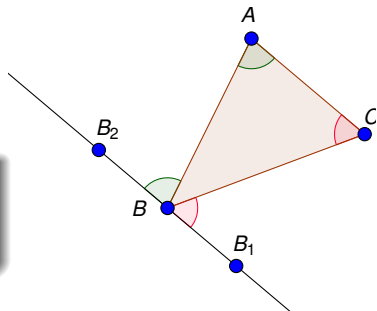


La somme des angles d'un triangle

On construit une parallèle à AC passant par B .

Problème !

Il faut montrer (ou admettre) que les angles sont alternes-internes.

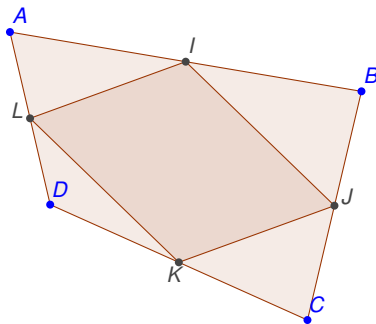


Le théorème de Varignon

Theorem

Soit $ABCD$ un quadrilatère. Soient I, J, K et L les milieux de AB, BC, CD , et AD , alors $IJKL$ est un parallélogramme.

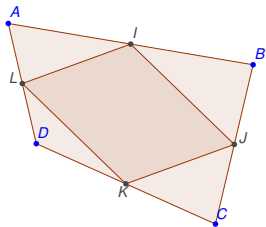
La preuve usuelle utilise le théorème de la droite des milieux: Dans le triangle ABC on a $AC \parallel IJ$. De même, on a $AC \parallel LK$. D'où $LK \parallel IJ$. De même $IL \parallel JK$.



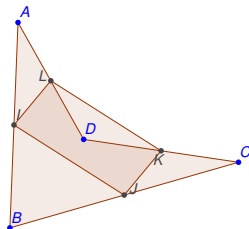
Si les côtés AB, BC, CD, DA d'une figure rectiligne de quatre côtés, sont divisés chacun en deux parties égales en F, G, H, E , & que les points des divisions soient joints par les lignes droites FE, EH, HG, GF , la figure quadrilatérale $FEHG$ est un parallélogramme; car en menant les lignes DI, AC , comme par l'hypothèse, $AF = FB$ & $AI = DI$, $AF \cdot FB = AI \cdot DI$, & ainsi (*Prop. 2.*) EF est parallèle à DB . De même puisqu'on, par l'hypothèse, $BC = GC$, & $DH = HC$, $BC = GC$, & $DH = HC$, & par conséquent (*Prop. 2.*) GH fera encore parallèle à la ligne BD . Donc EF & GH sont parallèles à la même troisième ligne, elles sont donc aussi parallèles entre elles. On peut par la même raison prouver que les lignes FG & EH sont parallèles à la ligne droite AC , & par conséquent parallèles entre elles. Donc le quadrilatère $FEHG$ est un parallélogramme.

Preuve originale

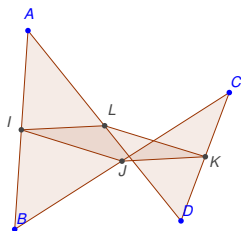
Le théorème de Varignon



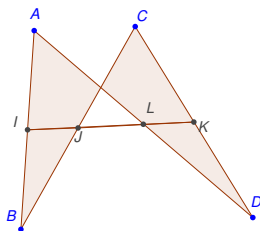
(a) Convex case



(b) Concave case



(c) Self-intersecting case



(d) Special case

Table des matières

1 Exemples d'erreurs

- En informatique
 - Dans des systèmes complexes. . .
 - . . . mais aussi dans votre premier programme
- En mathématiques
 - Des exemples historiques
 - Des exemples au collège
 - **Un exemple récent**
 - et dans les vérificateurs de preuves !

2 Les assistants de preuve

- Qu'est-ce qu'une preuve ?
- Langages de preuve
- Quelques résultats obtenus

3 L'enseignement et les assistants de preuve

4 Compléments

- Qui vérifie le vérificateur ?
- Automatisation
- Curry-Howard

Un exemple récent

"In October, 1998, Carlos Simpson submitted to the arXiv preprint server a paper called "Homotopy types of strict 3-groupoids". It claimed to provide an argument that implied that the main result of the " ∞ -groupoids" paper, which M. Kapranov and I had published in 1989, can not be true. However, Kapranov and I had considered a similar critique ourselves and had convinced each other that it did not apply. I was sure that we were right until the Fall of 2013 (!!)."

Vladimir Voevodsky

Source:

http://www.math.ias.edu/vladimir/files/2014_IAS.pdf,
page 9.

"It soon became clear that the only real long-term solution to the problems that I encountered is to start using computers in the verification of mathematical reasoning. "

Vladimir Voevodsky (médaille field 2002)

Source:

http://www.math.ias.edu/vladimir/files/2014_IAS.pdf,
page 13.

Bug dans les vérificateurs de preuves

Extrait de la liste des changements entre deux versions de Coq:

```
Changes from V8.5pl2 to V8.5pl3
```

```
=====
```

```
Critical bugfix
```

```
- #4876: Guard checker incompleteness when using primitive projections
```

En clair, avec Coq V8.5pl2 (par exemple) on pouvait prouver faux !

Pour résumer

Quelques types d'erreurs:

- Trou dans la preuve/cas oublié/hypothèse implicite
- Circularité
- Définitions changeantes
- Erreur de manipulation de quantificateurs

Les erreurs:

- Tout le monde en fait: du collégien au médaille field
- C'est normal.
- On n'en parle pas.

Les erreurs:

- Tout le monde en fait: du collégien au médaille field
- C'est normal.
- On n'en parle pas.

TOUT LE MONDE FAIT CACA!



Rascal et Pascal Lemaitre

Mais !

Heureusement !

L'humain arrive très bien à faire des preuves quasi-correctes même en ayant fait une erreur précédemment !

Attention !

Ce n'est pas vrai des prouveurs automatiques:
erreur de typo \rightarrow hypothèses contradictoires \rightarrow preuve de faux \rightarrow preuve triviale.

Table des matières

1 Exemples d'erreurs

- En informatique
- En mathématiques

2 Les assistants de preuve

- Qu'est-ce qu'une preuve ?
- Langages de preuve
- Quelques résultats obtenus

3 L'enseignement et les assistants de preuve

4 Compléments

- Qui vérifie le vérificateur ?
- Automatisation
- Curry-Howard

Qu'est-ce qu'une preuve ?

- 1 un argument convainquant

Qu'est-ce qu'une preuve ?

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

Qu'est-ce qu'une preuve ?

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

Point de vue

On peut voir une *preuve informelle* comme un argument pour convaincre le lecteur de l'existence d'une *preuve formelle*.

*“Du point de vue épistémologique, les considérations précédentes montrent qu’il suffit de savoir transcrire une démonstration mathématique en déduction naturelle dans le système de Quine pour être assuré de **l’existence d’une transcription formalisée** dans le cadre théorique du calcul des prédicats. Autrement dit, la distance entre démonstration pratique et démonstration formalisée n’est peut-être pas aussi infranchissable qu’on l’affirme parfois.”* dans MÉTHODES DE RAISONNEMENT ET LEURS MODÉLISATIONS LOGIQUES. SPÉCIFICITÉ DE L’ANALYSE. QUELLES IMPLICATIONS DIDACTIQUES ? Viviane Durand-Guerrier, Gilbert Arsas, volume 23-3, 2003 Recherches en didactique des mathématiques.

Processus social

Remarque

Même quand on a une preuve formalisée, cela reste un processus social, il faut se convaincre que les hypothèses, définitions et l'énoncé sont bien conformes à ce que l'on souhaite démontrer.

Oui mais...

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

- 1 hypothèses implicites

Oui mais...

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

- 1 hypothèses implicites
- 2 erreur typographique

Oui mais...

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

- 1 hypothèses implicites
- 2 erreur typographique
- 3 définitions incohérentes

Oui mais...

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

- 1 hypothèses implicites
- 2 erreur typographique
- 3 définitions incohérentes
- 4 présence de calculs non vérifiables à la main

Oui mais...

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

- 1 hypothèses implicites
- 2 erreur typographique
- 3 définitions incohérentes
- 4 présence de calculs non vérifiables à la main
- 5 preuves très longues

Oui mais...

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

- 1 hypothèses implicites
- 2 erreur typographique
- 3 définitions incohérentes
- 4 présence de calculs non vérifiables à la main
- 5 preuves très longues
- 6 preuves très compliquées

Oui mais...

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

- 1 hypothèses implicites
- 2 erreur typographique
- 3 définitions incohérentes
- 4 présence de calculs non vérifiables à la main
- 5 preuves très longues
- 6 preuves très compliquées
- 7 trop de détails techniques, trop de cas pour les traiter à la main sans faire d'erreurs

Heureusement !

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

Heureusement !

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

On peut donc construire des assistants de preuve!

Exemples

- Automath (1967)
- Coq (1984)
- HOL-Light (90s)
- HOL (1988)
- Isabelle (1986)
- Lean (2013)
- Matita (1999)
- Mizar (1973)
- ...

- Un nom relativement mal choisi:
 - ▶ Ils n'aident pas à trouver une preuve, seulement à *vérifier* qu'une preuve est correcte.
 - ▶ Comme c'est difficile d'obtenir une preuve formelle, on la construit de manière *interactive* (ITP Interactive Theorem Proving)
- Le domaine frère est celui de la preuve automatique (ATP: Automated Theorem Proving) :la machine qui transforme les cochons en saucisses.
- Des frères siamois:
 - ▶ L'automatisation est intégrée à la preuve interactive:
 - ★ A petite échelle (hachoir à viande + poussoir à saucisses).
 - ★ A échelle moyenne (machine à saucisses).
 - ▶ La preuve automatique est souvent un peu interactive (choix de l'algo et de ses paramètres, encodage, *hints*,...).

Qu'est-ce qu'un assistant de preuve ?

Un logiciel qui permet de :

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

Qu'est-ce qu'un assistant de preuve ?

Un logiciel qui permet de :

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

Qu'est-ce que ce n'est pas ?

- Un démonstrateur automatique
- Un outil qui facilite l'obtention des preuves

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*;
- et la **vérification automatique** de la correction de démonstration *par le système*.

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

Un compromis

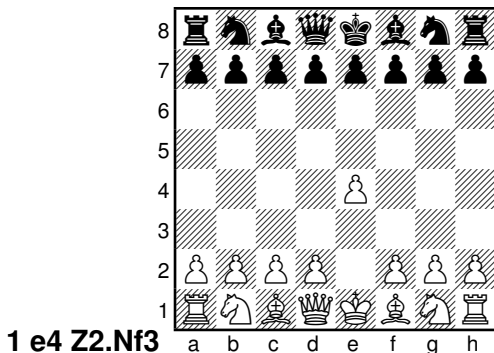
Un langage informatique est un compromis entre pouvoir d'expressivité et capacité d'exécution.

Pour décrire les preuves

- 1 Langages procéduraux.
- 2 Langages déclaratifs.

Pour décrire les preuves

- 1 Langages procéduraux. On donne des ordres (appelés tactiques) pour compléter l'arbre de preuve qui est en construction (LCF Style)
- 2 Langages déclaratifs. On décrit les assertions mathématiques que l'on veut déduire et une justification.



Le concept de noyau

- Les règles élémentaires sont dans le noyau.
- On doit faire **confiance** au noyau.
- Mais on a tout autour des mécanismes (notations, coercions, automatisations) pour **aider** à générer la suite d'application de règles élémentaires.



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 ?

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

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

Le problème de la vérification d'une preuve

Présence de calculs

Théorème des 4 couleurs

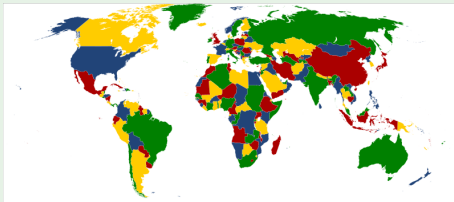
Quatre couleurs suffisent pour colorier une carte géographique *plane* sans que deux pays ayant une *frontière* en commun ne soient de la même couleur.

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



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



Photo par Robert Cudmore

1998 Preuve par Thomas Hales

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."

2004 - 2014 Projet Flyspeck: formalisation du théorème en HOL-light avec des contributions en Coq et Isabelle

Le problème de la vérification d'une preuve

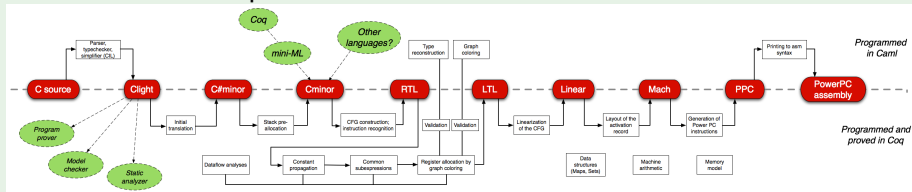
Trop de détails techniques

Un compilateur

CompCert un compilateur C prouvé formellement (Xavier Leroy).

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, 4 200 théorèmes

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

[//ssr2.msr-inria.inria.fr/~jenkins/current/progress.html](http://ssr2.msr-inria.inria.fr/~jenkins/current/progress.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 Exemples d'erreurs
 - En informatique
 - En mathématiques
- 2 Les assistants de preuve
 - Qu'est-ce qu'une preuve ?
 - Langages de preuve
 - Quelques résultats obtenus
- 3 L'enseignement et les assistants de preuve
- 4 Compléments
 - Qui vérifie le vérificateur ?
 - Automatisation
 - Curry-Howard

L'ordinateur pour l'enseignement des maths

On utilise des logiciels

- 1 pour faire des calculs numériques
- 2 pour faire des calculs symboliques (Maple. . .)
- 3 pour faire des conjecture et exercices de construction (Cabri, . . . , GeoGebra)
- 4 tester si une conjecture est “vraie” (GeoGebra récent)

Mais bizarrement

assez peu pour faire des **preuves** !

“Among mathematicians computer proof verification was almost a forbidden subject. A conversation started about the need for computer proof assistants would invariably drift to the Goedel Incompleteness Theorem (which has nothing to do with the actual problem) or to one or two cases of verification of already existing proofs, which were used only to demonstrate how impractical the whole idea was.”

Voevodsky

Enseignement de la preuve

- Les enseignants ne connaissent souvent pas la logique, et en n'éprouvent pas le besoin.
- Seules quelques règles sont expliquées: absurde, contraposée, raisonnement par cas.
- L'apprentissage se fait principalement par imitation et vérification sémantique.

"If you can't explain mathematics to a machine, it is an illusion to think you can explain it to a student."
De Bruijn "Invited lecture at the Mathematics Knowledge Management Symposium", 25-29 November 2003, Heriot-Watt University, Edinburgh, Scotland



Point de vue

La preuve peut être vue comme une partie d'un jeu, il devient alors évident qu'il faut:

- expliciter les règles
- montrer le plateau

Thèse

Avoir une définition précise de preuve aide à expliquer ce qui est attendu.

Preuve formelle vs informelle

”La preuve formelle est impossible à obtenir.”

Objectifs du domaine:

- 1 Montrer que c’est faux.
- 2 Réduire le fossé entre la preuve formelle et la preuve informelle.

Granularité

La granularité usuelle n'est pas celle des règles de la logique.

Exemple: la preuve de $A \Rightarrow A$ dans le système de Hilbert.

$$\frac{\frac{\frac{A \rightarrow ((B \rightarrow A) \rightarrow A)}{((A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A))} \text{S}}{(A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A)} \text{MP}}{A \rightarrow A} \text{MP}$$

Un vernaculaire mathématique

La logique cohérente.

$$\forall x, H_1(x) \wedge \dots \wedge H_n(x) \rightarrow \begin{array}{c} \exists y, P_1(x, y) \wedge \dots \wedge P_k(x, y) \\ \vee \\ \dots \end{array}$$

Plusieurs auteurs ont identifié ce fragment de la logique du premier ordre comme pertinent pour l'automatisation ou la lisibilité.
on obtient des preuves **relativement** lisibles ².

Pourquoi utiliser un assistant de preuve pour enseigner la preuve ?

- Clarifier les règles du jeu: les règles de raisonnement sont explicites.
- Clarifier le contexte: les théorèmes admis sont explicites.
- Un critère objectif pour la correction d'une preuve: pas de conflits
- Clarifie les différentes notions de base: axiome, hypothèse, lemme, théorème, conjecture, contre-exemple. . .
- Distingue les hypothèses, de celles du dessin.
- Boucle d'interaction: l'ordinateur se lasse moins vite de corriger des copies
- Motivation: effet jeu vidéo
- Automatisation: possibilité d'entraînement à la maison, correction automatique

Disclaimer

- En complément d'une approche classique car ne mobilise pas toutes les compétences attendues:
 - ▶ Connaître les définitions et les théorèmes (le système le fait pour vous)
 - ▶ Savoir si un théorème s'applique ou pas (le système vous aide)
 - ▶ Savoir modéliser
- Utilisable uniquement pour des exercices simples: premières inductions, preuves sur les fonctions injective/surjective
- L'objectif n'est pas d'automatiser la correction

Limites

Limites

- L'utilisation d'un assistant de preuve permet de travailler un seul aspect: la rédaction d'une preuve pour **vérifier** pas pour **expliquer**.
- Cela ne permet pas de discuter et comprendre les définitions, connaître ses théorèmes, faire des conjecture, ...
- A mon sens un outil utile à un moment de l'apprentissage (L1)

Difficultés

- Trouver le bon langage/la bonne interface
- Avoir les fondements formalisés
- Rendre implicite/automatique ce qui doit l'être (en fonction du contexte)

Travaux existants

Deux communautés:

- 1 Didactique
- 2 Informatique/Logique

Travaux issus du monde de la didactique

Geometry Tutor ³, MENTONIEZH ⁴, DEFI ⁵, CHYPRE ⁶, Geometrix ⁷, Cabri Euclide ⁸, Baghera ⁹, AgentGeom, geogebraTUTOR and Turing ¹⁰

³ anderson`geometry`1985

⁴ py`reconnaissance`1990

⁵ ag-almouloud`ordinateur`1992

⁶ bernat`chypre:`1993

⁷ gressier`geometrix`1988

⁸ luengo`cabri-euclide:`1997

⁹ balacheff`baghera`1999

¹⁰ richard`didactic`2011

Preuve interactive pour l'enseignement

En informatique

- Niveau Master - preuve de programmes, sémantique des langages de programmation, théorie des langages de programmation: U-Penn, Portland, Princeton, Harvard, Warsaw, CNAM, Lyon, Nice, Paris, Strasbourg, ...

En mathématiques

- Niveau Licence - Logique: Bordeaux, Warsaw, Pohang
- Niveau Licence - Maths générales: Nijmegen (ProofWeb), Nice (CoqWeb)
- Niveau Master - Géométrie: Strasbourg
- ...

Lurch (Carter et Monks)

”Un traitement de texte qui vérifie les preuves”

- Input sous forme de texte
- **Pseudo** langue naturelle
- Avec annotations

Theorem: *There does not exist a set of all sets which are not members of themselves.*

Proof:

(*) Assume $\exists S, S = \{x : x \notin x\}$.
Let R be such a set, i.e. $R = \{x : x \notin x\}$ by EI.
Suppose $R \in R$.
Then $R \in \{x : x \notin x\}$ by substitution,
and so $R \notin R$ by the definition of set builder notation.
But this is equivalent to $\neg(R \in R)$ by the definition of \notin .
This is a contradiction to our assumption that $R \in R$ by the definition of contradiction.
Hence we have a proof by contradiction that $\neg(R \in R)$
(since assuming $R \in R$ led to a contradiction).
But this shows that $R \notin R$ by the definition of \notin
which in turn implies that $R \in \{x : x \notin x\}$ by the definition of set builder notation.
So $R \in R$ by substitution since $R = \{x : x \notin x\}$.
Once again this is a contradiction to our proof that $\neg(R \in R)$ by the definition of contradiction.
Hence $\neg \exists S, S = \{x : x \notin x\}$ by another proof by contradiction since assuming (*) led to a contradiction.

■

- Application web pour faire des preuves de manière interactive
- Embarque Coq
- Expériences dans quelques universités françaises
- Input: souris, la syntaxe est toujours correcte par construction (un peu comme en Scratch pour la programmation)
- Que des déductions correctes (même si pas forcément les bonnes)
- Variables du raisonnement (meta-variables)

Deux modes

1 Logique

- ▶ Utilise les règles de la déduction naturelle.
- ▶ On peut afficher l'arbre de preuve.
- ▶ Raisonnement vers arrière

2 Maths

- ▶ Raisonnement vers l'avant ou vers l'arrière.
- ▶ Etapes de preuve à grain moins fin qu'en mode logique.

Home Calculus Exercise 12 edukera

Deduction of E 12

Prove: Reasoning

- parallelism of opposite sides
- non-contradiction
- equal opposite sides
- equality of opposite vectors

Initial content

Let A 1 2

Let B Preview

Let C 5 AKBJ is a parallelogram

Let I Conclusion

Let J 6 (AJ) and (KB) are parallel

1 Assum according to 5, by parallelism of opposite sides

2 Assum ↶ ↷ ↸ Apply

Let K be a point

3 K is the symmetric of J with respect to I
by construction of K

4 I is the midpoint of the segment [KJ]
according to 3, by definition of the symmetric

5 AKBJ is a parallelogram Deduce

according to 4 1, by intersection of the diagonals at their mutual midpoint I x

Conclusion

6 (BC) and (IJ) are parallel

Justify 0:28 / 1:42 ⚙️ 🖥️ 📐

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?
- Comment définir les structures algébriques ? (sous-typage, héritage, . . .)

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?
- Comment définir les structures algébriques ? (sous-typage, héritage, . . .)
- Comment automatiser les preuves à grande échelle ou à petite échelle ?

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?
- Comment définir les structures algébriques ? (sous-typage, héritage, . . .)
- Comment automatiser les preuves à grande échelle ou à petite échelle ?
- Comment échanger des preuves ?

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?
- Comment définir les structures algébriques ? (sous-typage, héritage, . . .)
- Comment automatiser les preuves à grande échelle ou à petite échelle ?
- Comment échanger des preuves ?
- Quelles sont les bonnes définitions ?

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?
- Comment définir les structures algébriques ? (sous-typage, héritage, . . .)
- Comment automatiser les preuves à grande échelle ou à petite échelle ?
- Comment échanger des preuves ?
- Quelles sont les bonnes définitions ?
- Comment prouver des programmes (impératifs ou fonctionnels) ?

Quelques défis de ce domaine de recherche

- Comment décrire les preuves ?
- Comment définir les structures algébriques ? (sous-typage, héritage, ...)
- Comment automatiser les preuves à grande échelle ou à petite échelle ?
- Comment échanger des preuves ?
- Quelles sont les bonnes définitions ?
- Comment prouver des programmes (impératifs ou fonctionnels) ?
- ...

Conclusion

- Les assistants de preuve sont utiles en mathématiques et en informatique.
- C'est maintenant faisable de formaliser de grandes preuves.
- Essayons de s'en servir pour l'enseignement.
- Attention c'est addictif !

Merci de votre attention.

Table des matières

- 1 Exemples d'erreurs
 - En informatique
 - En mathématiques
- 2 Les assistants de preuve
 - Qu'est-ce qu'une preuve ?
 - Langages de preuve
 - Quelques résultats obtenus
- 3 L'enseignement et les assistants de preuve
- 4 Compléments
 - Qui vérifie le vérificateur ?
 - Automatisation
 - Curry-Howard

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 certifiées par un *noyau*.
 - ▶ Isabelle (très petit)
 - ▶ HOL (très petit)
 - ▶ Coq (relativement petit)

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

Et l'automatisation ?

Une approche sceptique: on ne fait pas confiance aux démonstrateurs automatiques.

Deux solutions:

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

Exemples

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

Correspondance de Curry-Howard I

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

Correspondance de Curry-Howard I

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

Correspondance de Curry-Howard I

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

Règle de typage de l'application:

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

Correspondance de Curry-Howard I

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

Règle de typage de l'application:

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

Modus ponens:

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

Correspondance de Curry-Howard I

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

Règle de typage de l'application:

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

Modus ponens:

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

Correspondance de Curry-Howard I

logique

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

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

programmation

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\text{fun } x : A \mapsto t) : A \rightarrow B}$$

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

Correspondance de Curry-Howard I

logique	programmation
$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$	$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$
$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$

Correspondance de Curry-Howard I

logique	programmation
$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$	$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$
$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$

Correspondance de Curry-Howard II

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

Correspondance de Curry-Howard II

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

Correspondance de Curry-Howard II

$$\begin{array}{c} \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \\ \\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \\ \\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \end{array} \quad \left| \quad \begin{array}{c} \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B} \\ \\ \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash A} \\ \\ \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash B} \end{array}$$

Correspondance de Curry-Howard II

$$\begin{array}{c|c} \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} & \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B} \\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} & \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \text{fst } t : A} \\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} & \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \text{snd } t : B} \end{array}$$

Correspondance de Curry-Howard III

$$\frac{\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}}{\Gamma \vdash A \vee B} \quad \left| \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash \mathit{inl} a : A + B} \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathit{inr} b : A + B}}{\Gamma \vdash \mathit{case } m \text{ of } \mathit{inl}(a) \Rightarrow t \mid \mathit{inr}(a) \Rightarrow u : C}$$
$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

Règles de simplification

$$\mathit{fst}(A, B) = A$$

$$\mathit{snd}(A, B) = B$$

$$\mathit{case}(\mathit{inl} m) \mathit{of} \mathit{inl}(a) \Rightarrow t \mid \mathit{inr}(a) \Rightarrow u = t[x := m]$$

$$\mathit{case}(\mathit{inr} m) \mathit{of} \mathit{inl}(a) \Rightarrow t \mid \mathit{inr}(a) \Rightarrow u = u[x := m]$$

Correspondance de Curry-Howard

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

Sémantique de Heyting-Kolmogorov

La sémantique de Heyting-Kolmogorov consiste à donner une interprétation fonctionnelle des démonstrations.

- Une preuve de $A \rightarrow B$ est une *fonction* qui, à partir d'une preuve de A donne une preuve de B .
- Une preuve de $A \wedge B$ est une *paire* composée d'une preuve de A et d'une preuve de B .
- Une preuve de $A \vee B$ est une paire (i, p) avec ($i = 0$ et p une preuve de A) ou ($i = 1$ et p une preuve de B).
- Une preuve de $\forall x.A$ est une fonction qui, pour chaque objet t construit un objet de type $A[x := t]$.

Cette interprétation consiste à *calculer avec des preuves*. Cela paraît très proche de la programmation fonctionnelle et du λ -calcul.

Retour sur la correspondance de Curry-Howard à travers la curryfication

Definition `curry A B C (f : (A * B) -> C)`
`(x : A) (y : B) := f (x, y) .`

```
Lemma curry_prop: forall A B C : Type,  
  (A * B -> C) -> A -> B -> C.
```

Proof.

```
  intros A B C f x y.  
  apply f.  
  split.  
  apply x.  
  apply y.  
Defined.
```

Print `curry_prop`.

On obtient:

```
curry_prop =  
fun (A B C : Type) (f : A * B -> C)  
      (x : A) (y : B) => f (x, y)  
  : forall A B C : Type,  
      (A * B -> C) -> A -> B -> C
```