Aspects pratiques de la théorie des types

Yves Bertot, Inria
Pierre Castéran, Univ. Bordeaux et LaBRI

Bordeaux, 13 décembre 2018

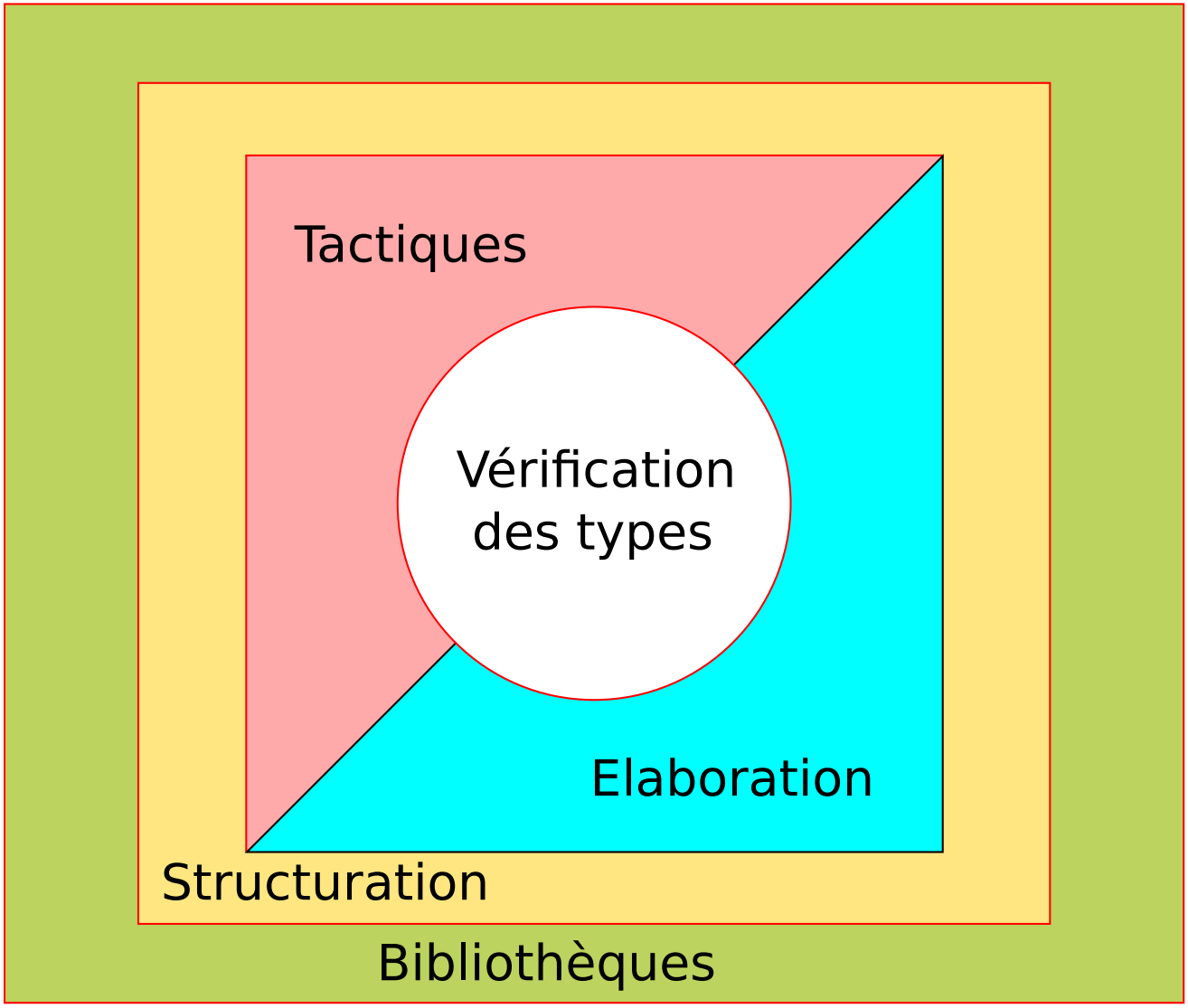
Plan de l'exposé

Mêler algorithmes, propriétés et preuves

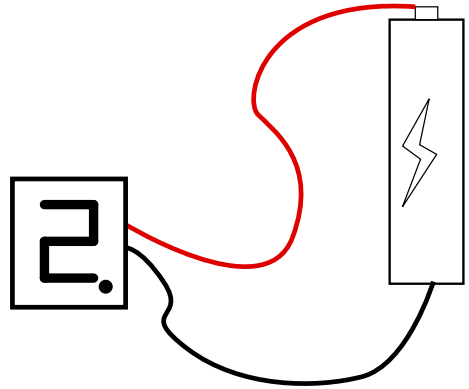
Un langage unique pour programmer, spécifier, prouver

- Les propriétés documentent les algorithmes
  - Documentation vérifiée par l'ordinateur
- Les propriétés logiques évitent les ambiguités
  - Les preuves de ces propriétés sont exhaustives
- Les possibilités d'erreurs sont réduites
- Exemple : Compcert par X. Leroy (2006)
  - Algorithme de compilation écrit en Coq
  - Langage C et assembleur également décrits formellement
  - La propriété est que le comportement des programmes est préservé
  - Le meilleur en termes de fiabilité (Yang et al. 2011)

Architecture du système



Typage et fiabilité



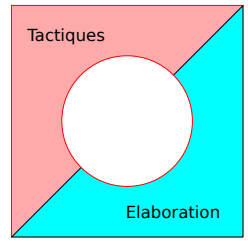
- Des types pour garantir le bon fonctionnement
  - en entrée et en sortie
- Utilisable également pour vérifier les preuves
  - Les programmes consomment des données
  - Les théorèmes consomment des hypothèses

Le calcul des constructions inductives

- Une variante du λ-calcul typé (Church ca. 1940)
- Calcul des constructions (Coquand 1984)
- Types inductifs (Paulin-Mohring 1993)
- Les termes représentent des programmes ou des preuves
- Les types sont des types de données ou des propositions
- Beaucoup d'informations sont obligatoires
  - Beaucoup de répétitions
  - La vérification est décidable
  - Un "assembleur" de la logique
- Acronyme : CIC

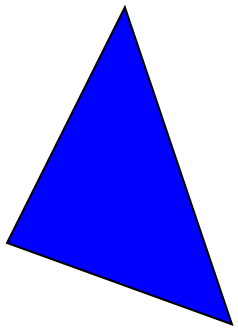
Faciliter la construction des preuves

- Les preuves sont trop laborieuses pour un humain
- Première approche : preuve par tactique
  - Ne pas écrire les preuves dans le langage CIC
  - Appeler des fonctions qui font cette construction
  - Disposer d'un langage qui peut faire des essais
- Deuxième approche : élaboration
  - Ecrire directement la structure des expressions
  - Le reste est rempli par Coq
  - Applicables pour les preuves et les formules
- Ces deux approches sont programmables

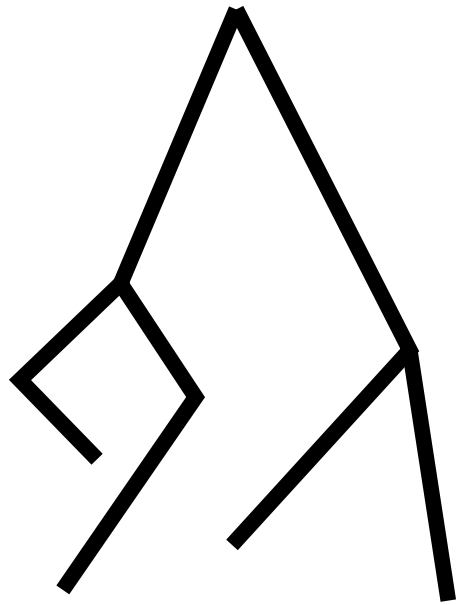


Deux approches pour construire les preuves

Tactiques

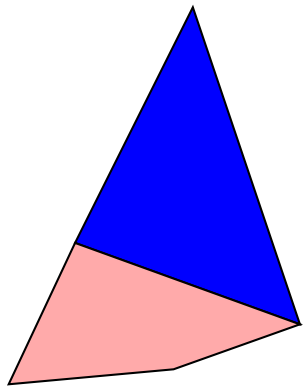


Elaboration

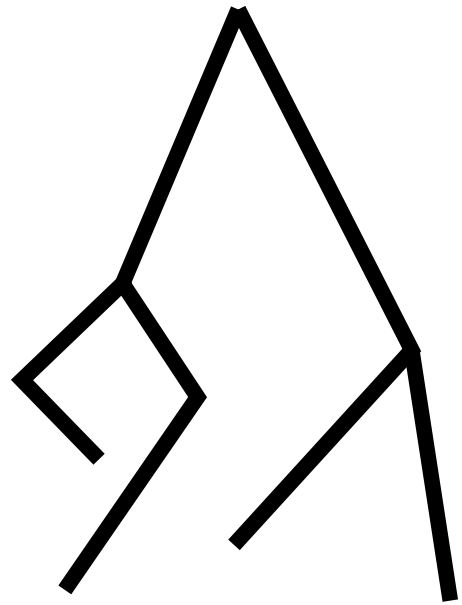


Deux approches pour construire les preuves

Tactiques

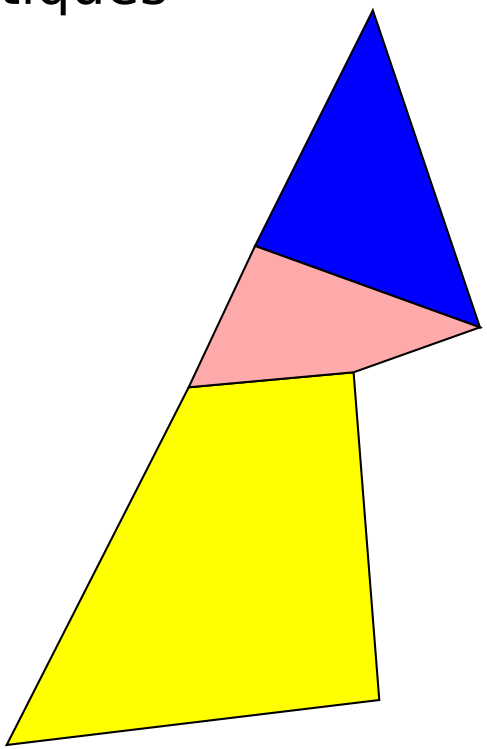


Elaboration

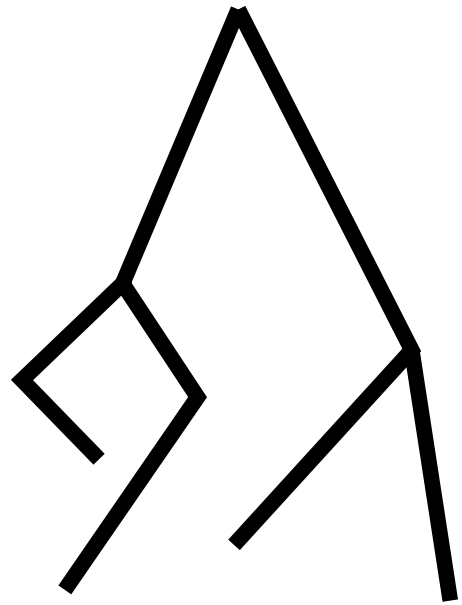


Deux approches pour construire les preuves

Tactiques

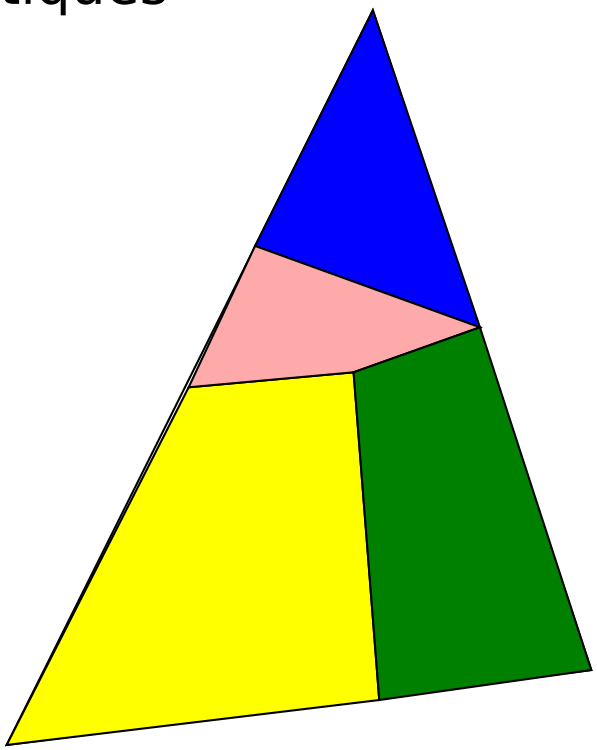


Elaboration

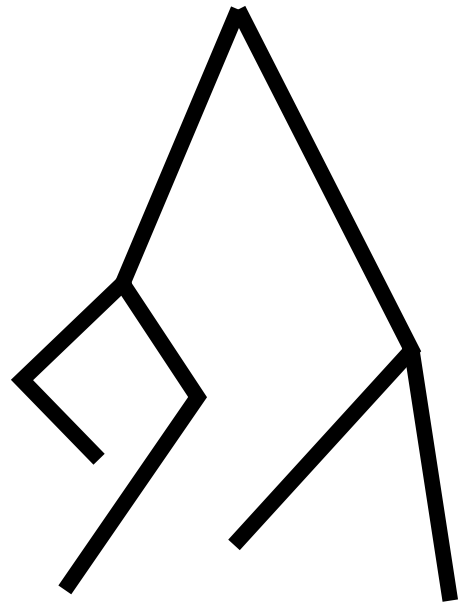


Deux approches pour construire les preuves

Tactiques

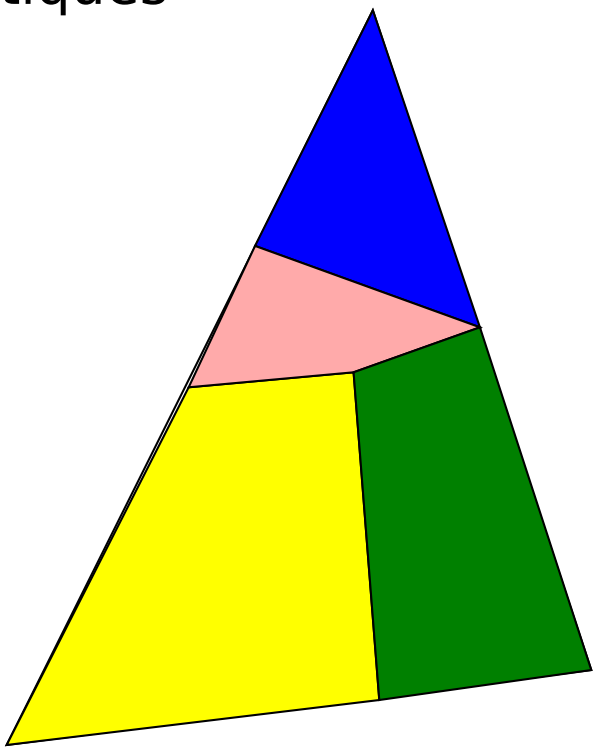


Elaboration

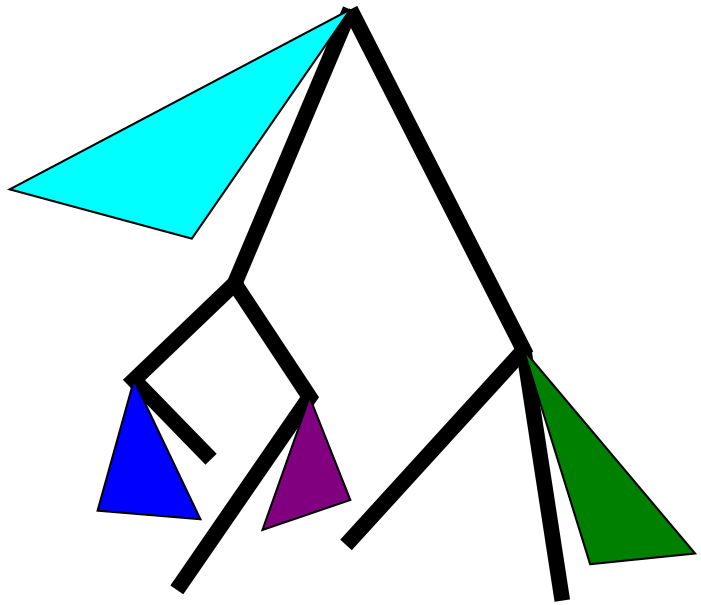


Deux approches pour construire les preuves

Tactiques

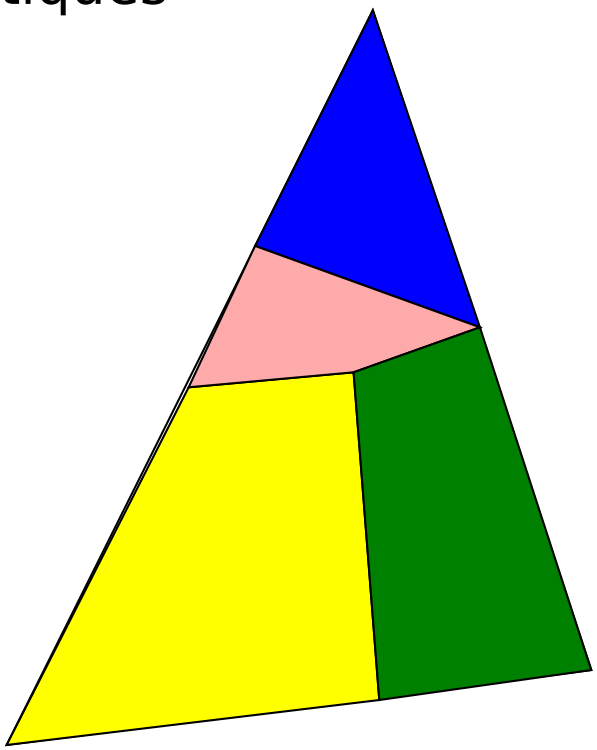


Elaboration

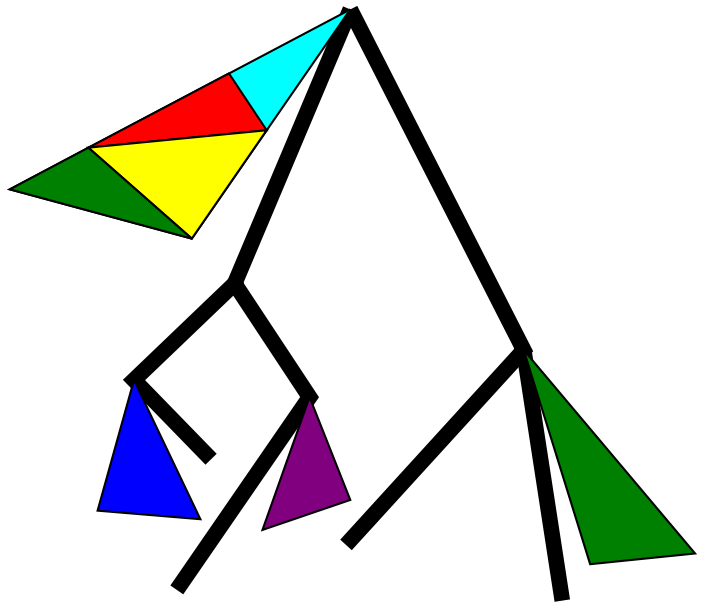


Deux approches pour construire les preuves

Tactiques

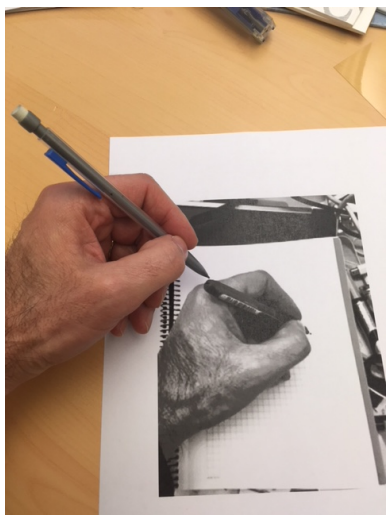


Elaboration



Utiliser les algorithmes que l'on prouve

- Décrire une théorie mathématique
  - Formules, algorithme, et preuve
- Faire travailler l'algorithme et sa preuve
  - Traduire un problème en une formule
  - Appeler l'algorithme sur cette formule
  - Invoquer la preuve sur le résultat
- Terme consacré : *réflexion*
- Rêve ultime : Décrire Coq lui-même dans Coq (Barras 1996)



Exemple de preuve par réflexion

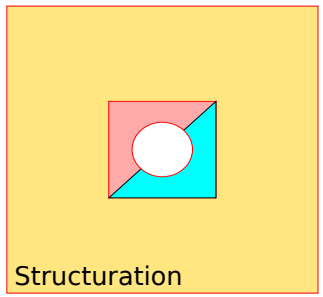
- Formules : $\Leftrightarrow$, variables propositionnelles,
- Algorithme : compter le nombre d'occurrences de chaque variable, s'il est pair, la formule est valide
- Preuve : à faire en Coq
- $(A \Leftrightarrow B) \Leftrightarrow (B \Leftrightarrow (A \Leftrightarrow C)) \Leftrightarrow C$ est valide !

Efficacité du calcul symbolique

- Plusieurs niveaux d'efficacité
  1. calcul symbolique basé sur la théorie du λ-calcul
  2. calcul plus rapide basé sur une machine virtuelle (Grégoire 2002), (Grégoire & Mahboubi 2005)
  3. calcul "natif" basé sur le compilateur Ocaml (Dénès 2011)
- Remplacement de calculs symboliques par des calculs en circuit matériel

Structuration

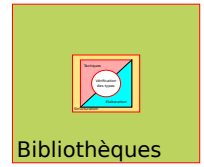
- Regrouper les connaissances en modules
- Proposer des théorèmes *généraux*
- Exploiter les similitudes entre $x + (y + z)$ et $x \times (y \times z)$
  - Propriétés d'anneaux, de groupes
  - Théories abstraites instanciables
  - Algorithmes pour passer du cas particulier au cas général
    - Type classes, (Sozeau 2008), Canonical structures (Saïbi 1997)
    - Modules et foncteurs (Chrzaszcz 2003)
- Intégration dans le mécanisme d'élaboration



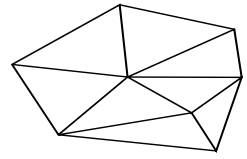
Bibliothèques et efforts collaboratifs

Nombreux sujets abordés

- Technologies de programmation
  - Compilateur, système d'exploitation
  - Algorithmique
- Calcul numérique, arithmétique des ordinateurs
  - Maîtrise des erreurs d'arrondi
  - Calcul de très grands nombres
  - Simulation numérique
- Cryptographie, commerce électronique
- Programmation concurrente et distribuée
- Mathématiques
  - Algèbre, topologie, combinatoire, géométrie

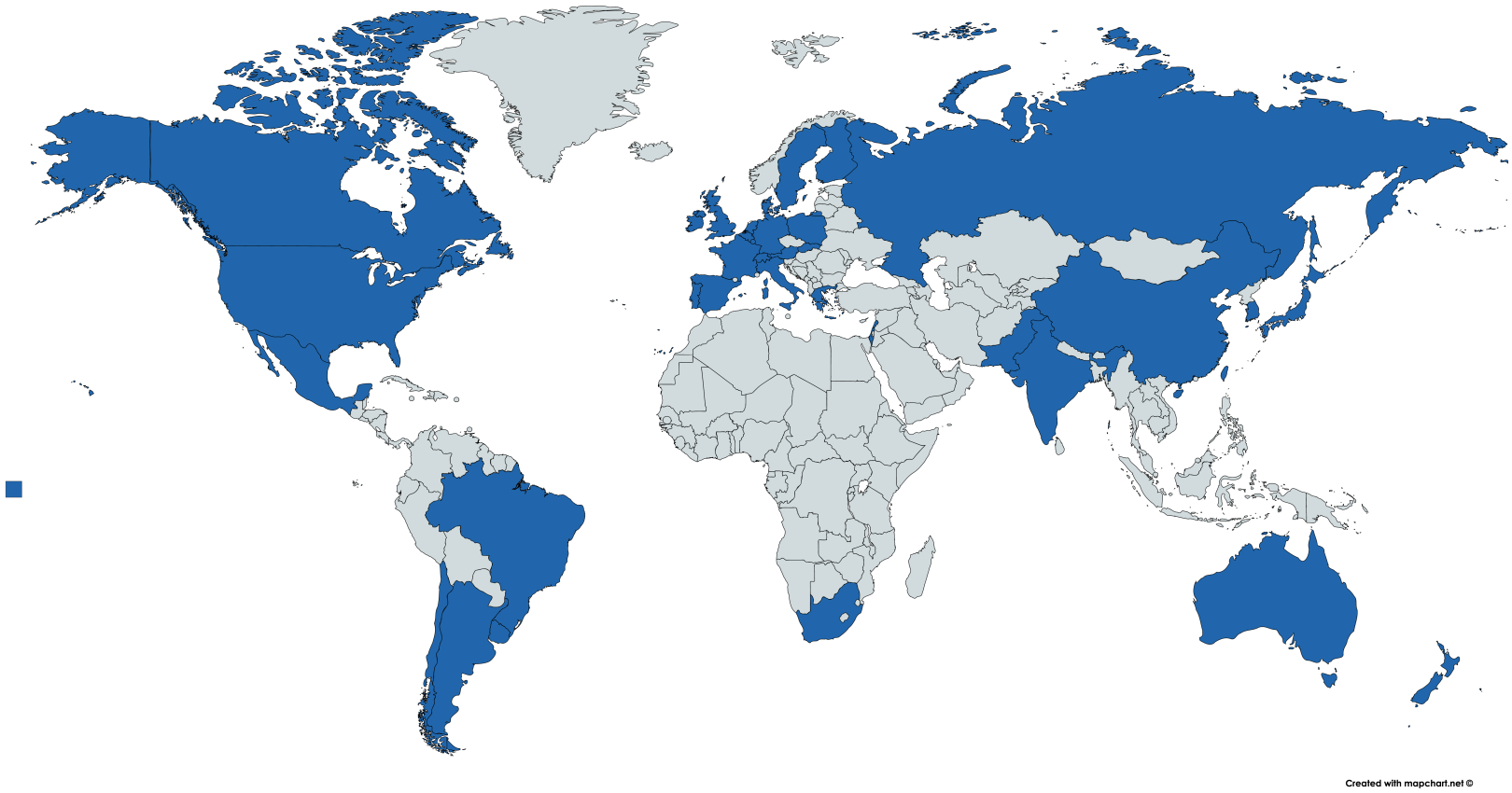


π = 3.1415926...



La communauté du système Coq

- Une dizaine de développeurs
- Une centaine de contributeurs
- Utilisation mondiale



Un exemple d'utilisation

Calculer les décimales de π

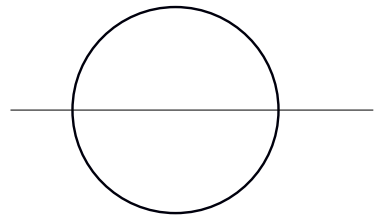
- Un exercice théorique
- Utilisé pour illustrer la puissance de super-ordinateurs

Une distinction entre nombres théoriques et pratiques

- Les nombres réels sont des nombres « limites »
- Les calculs sur ordinateurs ont une précision limitée
  - Calcul avec des nombres flottants
  - Calcul en précision fixe

Quelle définition pour π ?

- La surface d'un cercle de rayon 1
  - Valeur approchée calculée par Archimède
- La première racine positive de sin
- Plus précisément $\sin x = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{2i+1!}$
- Les deux notions sont bien reliées par le calcul intégral



Calculer π dans Coq

- Avec une calculette, on obtient une approximation
- Avec Coq, on prouve la correction d'une approximation (Melquiond 2008)

```
Require Import Reals Coquelicot.Coquelicot
  Interval.Interval_tactic.

Open Scope R_scope.

Lemma a_pi_approx : 314/100 <= PI <= 315/100.
Proof.
interval.
Qed.
```

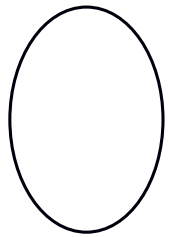
Calculer beaucoup de décimales de π

- Première approche, calculer $\pi \times 10^N$
- Deuxième approche, calculer $\pi \times 10^N \bmod 10$
  - plutôt $\pi \times 16^N \bmod 16$ (Bailey et al. 1997)
- Un bon algorithme est nécessaire
  - Exemple : algorithme basé sur les moyennes arithmetico géométriques

Une idée de Gauss : Moyenne arithmético-géométrique

Un couple de séquences de nombres réels

- $a_0 = 1 \quad b_0 = x \quad a_{n+1} = \frac{a_n + b_n}{2} \quad b_{n+1} = \sqrt{a_n b_n}$
- a_n et b_n convergent rapidement vers une valeur $f(x)$
- Il existe A tel que $a_n - b_n < A^{-2^n}$
- Le nombre de décimales double à chaque itération



Un outil pour calculer les *intégrales elliptiques*

- $\int_{-\infty}^{\infty} \frac{\mathrm{d}t}{\sqrt{(1^2+t^2)(x^2+t^2)}} = \int_{-\infty}^{\infty} \frac{\mathrm{d}t}{\sqrt{(a_n^2+t^2)(b_n^2+t^2)}}$

Le lien avec π

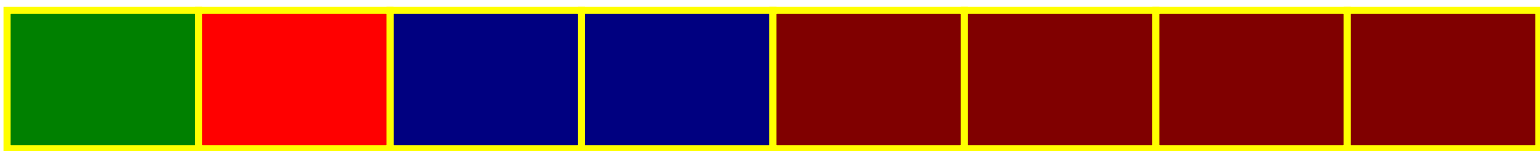
- Propriété centrale $\pi = 2\sqrt{2}\dfrac{f(1/\sqrt{2})^3}{f'(1/\sqrt{2})}$
- Si on définit $y_n(x) = \dfrac{a_n(x)}{b_n(x)}$ et $z_n(x) = \dfrac{b'_n(x)}{a'_n(x)}$
- Algorithme de Borwein & Borwein

$$y_0 = \sqrt{2} \quad y_{n+1} = \frac{1+y_n}{2\sqrt{y_n}} \quad z_1 = \sqrt{\sqrt{2}} \quad z_{n+1} = \frac{1+z_n y_n}{(1+z_n)\sqrt{y_n}}$$

$$\pi_n = (2+\sqrt{2}) \prod_{i=1}^{n} \frac{1+y_i}{1+z_i}$$

Convergence quadratique

$$0 \leq \pi_{n+1} - \pi \leq 8\sqrt{2} \times 531^{-2^n}$$



Les preuves en Coq sur les moyennes arithmético-géométrique

Grâce à une bibliothèque existante : Coquelicot (Boldo, Lelay, Melquiond, 2015)

- Limites
- Dérivées
- Intégrales impropres
- Convergence uniforme

En suivant le plan donné par une composition du CAPES

Tout ça, c'est de la théorie

L'ordinateur calcule avec des nombres de précision limitée
Chaque opération contient un arrondi

- On perd de la précision

Besoin de prouver que la perte de précision est faible

Défi : un million de décimales

- Calcul en virgule fixe
- Equivalent à calculer la partie entière $\left\lfloor \sqrt{2} \times 10^{10^6} \right\rfloor$, puis $\left\lfloor y_1 \times 10^{10^6} \right\rfloor$ puis $\left\lfloor \pi_1 \times 10^{10^6} \right\rfloor$, etc.
- Pour un million de décimales, il suffit de calculer π_{20}

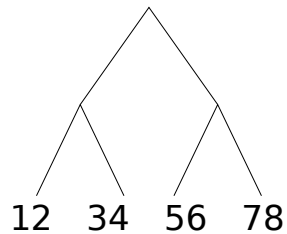
Calculer avec des entiers

- Programmer une « version » entière de l'algorithme qui calcule y_n, z_n, et π_n
- Chaque calcul est une approximation
- Les erreurs s'accumulent, mais on sait les quantifier
- Deux sources d'erreur à chaque étape
  1. Les erreurs sur les calculs précédents
  2. L'erreur d'arrondi introduite à cette étape
- L'influence de la première source est limitée par la dérivée
- Borne finale sur l'erreur où e est l'erreur élémentaire

$$|v_n - \pi| < (21 \times n + 3)e$$

Calculer avec des grands entiers

- On calcule mieux sur les grands entiers avec une représentation en arbre binaire
- Coq fournit une bibliothèque pour ça
- Toutes les operations sont prouvées par rapport aux entiers « standards »
- Raffinement progressif
  - Calcul réel $\rightarrow$ Calcul en nombres entiers $\rightarrow$ Calcul en arbres binaires
- Calcul en approximativement une heure
  - Exécution native



Travail collectif

- L'algorithme de Bailey et al. a aussi été vérifié par L. Rideau et L. Théry
- Merci à tous les développeurs qui ont fourni les éléments de cette expérience
- Spécialement S. Boldo, C. Lelay, G. Melquiond, N. Magaud, L. Théry, B. Grégoire, M. Dénès

Conclusion

Voici venu le temps des grosses preuves formelles

- “Stand on the shoulders of giants”

Encore des questions de langage

- Accumuler des mathématiques formalisées
- Permettre de retrouver les résultats existants

Un travail d’éducation important

- Nombreux livres sur Coq