Pour s'asseoir sur les fondations

Fig. 1. Jusqu'où un intellectuel assis peut-il aller ?

Pierre-Marie Pédrot
INRIA

Collège de France, 02/06/25

Tout un programme

« Formalisation des mathématiques et types dépendants »

Tout un programme

« Formalisation des mathématiques et types dépendants »

Une tension intrinsèque immédiate

Tout un programme

« Formalisation des mathématiques et types dépendants »

Une tension intrinsèque immédiate

$\rightsquigarrow$ À ma gauche, le monde mathématique

Tout un programme

« Formalisation des mathématiques et types dépendants »

Une tension intrinsèque immédiate

- $\rightsquigarrow$ À ma gauche, le monde mathématique
- $\rightsquigarrow$ À ma droite, le monde informatique

Tout un programme

« Formalisation des mathématiques et types dépendants »

Une tension intrinsèque immédiate

- $\rightsquigarrow$ À ma gauche, le monde mathématique
- $\rightsquigarrow$ À ma droite, le monde informatique

L'interprétation du et est heureusement laissée au lecteur !

- Intersection ? $A \cap B$
- Juxtaposition ? $A \,\&\, B$
- Superposition ? $A \otimes B$

Tout un programme

« Formalisation des mathématiques et types dépendants »

Une tension intrinsèque immédiate

- $\rightsquigarrow$ À ma gauche, le monde mathématique
- $\rightsquigarrow$ À ma droite, le monde informatique

L'interprétation du et est heureusement laissée au lecteur !

- Intersection ? $A \cap B$
- Juxtaposition ? $A \,\&\, B$
- Superposition ? $A \otimes B$

Comment en est-on arrivé là ?

Le chaînon manquant

L'équivalence preuve-programme, alias Curry-Howard

Le chaînon manquant

L'équivalence preuve-programme, alias Curry-Howard

Le mathématicien

Théorème. Pour tout $n \in \mathbb{N}$, il existe $p \in \mathbb{N}$ t.q. $n = 2p$ ou $n = 2p + 1$.

Preuve. Par induction sur n.

- Si $n = 0$, trivial car $p = 2 \times 0$.
- Sinon, soit $n = m + 1$. Par induction, soit p t.q. $m = 2p$ ou $m = 2p + 1$.
  - Dans le cas 1, $n = 2p + 1$.
  - Sinon, $n = 2(p + 1)$.

Le programmeur

```
val div2 : int -> int * bool
(* [div2 n] renvoie [n / 2] et
la parité de [n]. *)

let rec div2 n = match n with
| 0 -> (0, true)
| m + 1 ->
  let (p, even) = div2 m in
    if even then (p, false)
    else (p + 1, true)
```

Le chaînon manquant

L'équivalence preuve-programme, alias Curry-Howard

Le mathématicien

Théorème. Pour tout $n \in \mathbb{N}$, il existe $p \in \mathbb{N}$ t.q. $n = 2p$ ou $n = 2p + 1$.

Preuve. Par induction sur n.

- Si $n = 0$, trivial car $p = 2 \times 0$.
- Sinon, soit $n = m + 1$. Par induction, soit p t.q. $m = 2p$ ou $m = 2p + 1$.
  - Dans le cas 1, $n = 2p + 1$.
  - Sinon, $n = 2(p + 1)$.

Le programmeur

```
val div2 : int -> int * bool
(* [div2 n] renvoie [n / 2] et
la parité de [n]. *)

let rec div2 n = match n with
| 0 -> (0, true)
| m + 1 ->
  let (p, even) = div2 m in
    if even then (p, false)
    else (p + 1, true)
```

C'est la même chose !

Constructions dans un monde qui bouge

Cette équivalence est à la base de la théorie des types

Constructions dans un monde qui bouge

Cette équivalence est à la base de la théorie des types

Un **système logique** très puissant :

- On peut y écrire toutes les mathématiques conventionnelles
- Notion de calcul de première classe et inductifs de l'enfer

Constructions dans un monde qui bouge

Cette équivalence est à la base de la théorie des types

Un **système logique** très puissant :

- On peut y écrire toutes les mathématiques conventionnelles
- Notion de calcul de première classe et inductifs de l'enfer

Un **langage de programmation** très expressif :

- Des types extrêmement fins pour éliminer les *bugs*
- Pas de séparation nette entre typage et *runtime*

Constructions dans un monde qui bouge

Cette équivalence est à la base de la théorie des types

Un **système logique** très puissant :

- On peut y écrire toutes les mathématiques conventionnelles
- Notion de calcul de première classe et inductifs de l'enfer

Un **langage de programmation** très expressif :

- Des types extrêmement fins pour éliminer les *bugs*
- Pas de séparation nette entre typage et *runtime*

MLTT / CIC : l'incarnation de l'équivalence preuve-programme

Constructions dans un monde qui bouge

Cette équivalence est à la base de la théorie des types

Un **système logique** très puissant :

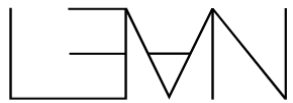
- On peut y écrire toutes les mathématiques conventionnelles
- Notion de calcul de première classe et inductifs de l'enfer

Un **langage de programmation** très expressif :

- Des types extrêmement fins pour éliminer les *bugs*
- Pas de séparation nette entre typage et *runtime*

MLTT / CIC : l'incarnation de l'équivalence preuve-programme

Des systèmes implémentés dans un certain nombre d'assistants à la preuve



D'où parles-tu, camarade ?

Qui suis-je ?

D'où parles-tu, camarade ?

Qui suis-je ?

UN INFORMATICIEN !

D'où parles-tu, camarade ?

Qui suis-je ?

UN INFORMATICIEN !

- Thèse soutenue en 2015
- Mon sujet d'étude est la théorie des types
- Un des principaux développeurs de l'assistant à la preuve Rocq

D'où parles-tu, camarade ?

Qui suis-je ?

UN INFORMATICIEN !

- Thèse soutenue en 2015
- Mon sujet d'étude est la théorie des types
- Un des principaux développeurs de l'assistant à la preuve Rocq

ATTENTION

Cet exsopé n'est pas un exsopé sur les isomorphimses

- Volontairement très haut niveau (pas de λ-termes !)
- Beaucoup d'enfonçage de portes ouvertes
- Public cible : les mathématiciens et les profanes

Un peu d'histoire

L'origine précise de la théorie des types est débattue

Un peu d'histoire

L'origine précise de la théorie des types est débattue

Selon la personne à qui on demande :

- Une structuralisation des ensembles (192X)
- Une origine philosophique constructiviste (197X)
- Un raffinement du λ-calcul (197X)
- L'algèbre universelle pour de vrai

Un peu d'histoire

L'origine précise de la théorie des types est débattue

Selon la personne à qui on demande :

- Une structuralisation des ensembles (192X)
- Une origine philosophique constructiviste (197X)
- Un raffinement du λ-calcul (197X)
- L'algèbre universelle pour de vrai

Moins de débat sur les premiers utilisateurs

Un peu d'histoire

L'origine précise de la théorie des types est débattue

Selon la personne à qui on demande :

- Une structuralisation des ensembles (192X)
- Une origine philosophique constructiviste (197X)
- Un raffinement du λ-calcul (197X)
- L'algèbre universelle pour de vrai

Moins de débat sur les premiers utilisateurs

LES INFORMATICIENS !

Un peu d'histoire

L'origine précise de la théorie des types est débattue

Selon la personne à qui on demande :

- Une structuralisation des ensembles (192X)
- Une origine philosophique constructiviste (197X)
- Un raffinement du λ-calcul (197X)
- L'algèbre universelle pour de vrai

Moins de débat sur les premiers utilisateurs

LES INFORMATICIENS !

Automath (1968) et surtout dans sa version moderne **Coq** (1984).

Pendant ce temps à Vera Cruz

L'engouement des mathématiciens pour les assistants à la preuve est récent

- Moins de 10 ans
- Encore très niche
- Surreprésentation de certains domaines

Pendant ce temps à Vera Cruz

L'engouement des mathématiciens pour les assistants à la preuve est récent

- Moins de 10 ans
- Encore très niche
- Surreprésentation de certains domaines

Un certain choc des cultures

- Pas les mêmes priorités
- Pas le même *background*
- Pas le même point de vue

Le pourquoi du comment

Un assistant à la preuve force à se poser des questions

- Quelle structure de donnée utiliser ?
- Comment passer d'une représentation à une autre ?

Le pourquoi du comment

Un assistant à la preuve force à se poser des questions

- Quelle structure de donnée utiliser ?
- Comment passer d'une représentation à une autre ?

... et très rapidement, des questions sur le langage

- Existe-t-il des types satisfaisant telle propriété ?
- Y a-t-il une représentation directe d'un encodage ?
- Puis-je transformer des isomorphismes en égalités ?

Le pourquoi du comment

Un assistant à la preuve force à se poser des questions

- Quelle structure de donnée utiliser ?
- Comment passer d'une représentation à une autre ?

... et très rapidement, des questions sur le langage

- Existe-t-il des types satisfaisant telle propriété ?
- Y a-t-il une représentation directe d'un encodage ?
- Puis-je transformer des isomorphismes en égalités ?

Ce sont des questions sur les fondations

Réponse générique

Triste topique

En général, les mathématiciens ont une aversion pour la logique

$\rightsquigarrow$ surtout les questions fondationnelles

$\rightsquigarrow$ très particulièrement en France

Triste topique

En général, les mathématiciens ont une aversion pour la logique

$\rightsquigarrow$ surtout les questions fondationnelles

$\rightsquigarrow$ très particulièrement en France

Arguments usuels par ordre d'acceptabilité sociale :

- Ça ne m'intéresse pas

Triste topique

En général, les mathématiciens ont une aversion pour la logique

$\rightsquigarrow$ surtout les questions fondationnelles

$\rightsquigarrow$ très particulièrement en France

Arguments usuels par ordre d'acceptabilité sociale :

- Ça ne m'intéresse pas
- Ça ne sert à rien

Triste topique

En général, les mathématiciens ont une aversion pour la logique

$\rightsquigarrow$ surtout les questions fondationnelles

$\rightsquigarrow$ très particulièrement en France

Arguments usuels par ordre d'acceptabilité sociale :

- Ça ne m'intéresse pas
- Ça ne sert à rien
- « La logique n'est plus stérile, elle engendre des paradoxes. » (Poincaré)

Triste topique

En général, les mathématiciens ont une aversion pour la logique

$\rightsquigarrow$ surtout les questions fondationnelles

$\rightsquigarrow$ très particulièrement en France

Arguments usuels par ordre d'acceptabilité sociale :

- Ça ne m'intéresse pas
- Ça ne sert à rien
- « La logique n'est plus stérile, elle engendre des paradoxes. » (Poincaré)

Il semble y avoir une cause historico-sociologique

Hilbert, Bourbaki and the scorning of logic (A. Mathias, 2013)

Thèse

Quand on formalise, les fondations c'est important...

Thèse

Quand on formalise, les fondations c'est important...

- 10 raisons méconnues, la $9^{\text{ème}}$ va vous surprendre !

Thèse

Quand on formalise, les fondations c'est important...

- ~~10 raisons méconnues, la 9ème va vous surprendre !~~

Thèse

Quand on formalise, les fondations c'est important...

- ~~10 raisons méconnues, la $9^{\text{ème}}$ va vous surprendre !~~
- ... pour pouvoir s'en passer.

Thèse

Quand on formalise, les fondations c'est important...

- ~~10 raisons méconnues, la 9ème va vous surprendre !~~
- ... pour pouvoir s'en passer.

Dans le reste de l'exposé, je vais tenter de vous en convaincre

- à travers la correspondance de Curry-Howard
- en surjouant le côté turbo-informaticien
- j'assume mes biais et autres monomanies

Pourquoi tant de n ?

Radotage

La théorie des types est un langage de programmation

Pourquoi tant de n ?

Radotage

La théorie des types est un langage de programmation

Observation

Il existe **beaucoup** de langages de programmation.

Pourquoi tant de n ?

Radotage

La théorie des types est un langage de programmation

Observation

Il existe **beaucoup** de langages de programmation.

- ... Mais alors, beaucoup.
- Vraiment beaucoup.
- J'insiste.

Pourquoi tant de n ?

Radotage

La théorie des types est un langage de programmation

Observation

Il existe **beaucoup** de langages de programmation.

- ... Mais alors, beaucoup.
- Vraiment beaucoup.
- J'insiste.

Une recherche rapide suggère plus de 8000 sur des critères incertains.

$\rightsquigarrow$ Tout étudiant en informatique implémente son propre langage un jour.

Tout est permis, mais rien n'est possible

Essentiellement tous équivalents (Turing-complétude)

Tout est permis, mais rien n'est possible

Essentiellement tous équivalents (Turing-complétude)

Pourquoi y a-t-il autant de langages de programmation ?

Tout est permis, mais rien n'est possible

Essentiellement tous équivalents (Turing-complétude)

Pourquoi y a-t-il autant de langages de programmation ?

> "— Beware of the Turing tarpit in which everything is possible but nothing of interest is easy."
>
> (Perlis)

Tout est permis, mais rien n'est possible

Essentiellement tous équivalents (Turing-complétude)

Pourquoi y a-t-il autant de langages de programmation ?

> "— Beware of the Turing tarpit in which everything is possible but nothing of interest is easy."
>
> (Perlis)

La notion d'*intéressant* n'est pas absolue !

- Peu de points en commun entre les développeurs
- Pléthore de langages pour chaque besoin
- Il n'existe pas de *meilleur* langage (mais des langages *meilleurs* ?...)

Cantor's Tarpit Hell

Quel est le paysage du côté de la logique ?

Cantor's Tarpit Hell

Quel est le paysage du côté de la logique ?

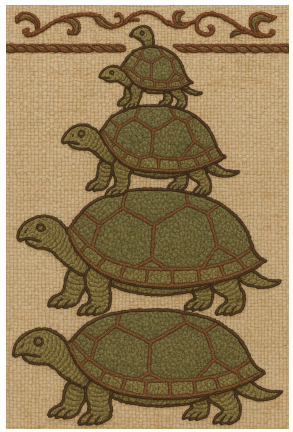
Pas de logique universelle (incomplétude de Gödel)

Cantor's Tarpit Hell

Quel est le paysage du côté de la logique ?

Pas de logique universelle (incomplétude de Gödel)

- les fondations sont fondamentalement non-équivalentes
- mais en pratique cette question n'est pas pertinente
- expressivité logique $\neq$ expressivité linguistique

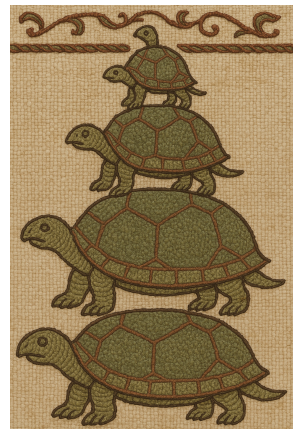


Cantor's Tarpit Hell

Quel est le paysage du côté de la logique ?

Pas de logique universelle (incomplétude de Gödel)

- les fondations sont fondamentalement non-équivalentes
- mais en pratique cette question n'est pas pertinente
- expressivité logique $\neq$ expressivité linguistique



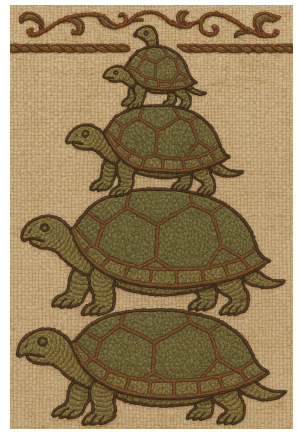
Assistants à la preuve matures basés sur la théorie des types : environ $3^{\dagger}$

Cantor's Tarpit Hell

Quel est le paysage du côté de la logique ?

Pas de logique universelle (incomplétude de Gödel)

- les fondations sont fondamentalement non-équivalentes
- mais en pratique cette question n'est pas pertinente
- expressivité logique $\neq$ expressivité linguistique



Assistants à la preuve matures basés sur la théorie des types : environ $3^{\dagger}$

Il y a ici un paradoxe majeur !

De pire en pire

En fait, c'est bien pire que ça.

De pire en pire

En fait, c'est bien pire que ça.

« La » mathématique n'existe pas

- Une unité de surface qui se craquelle très vite
- Des objets manipulés très différents
- Des techniques de preuves extrêmement variables

De pire en pire

En fait, c'est bien pire que ça.

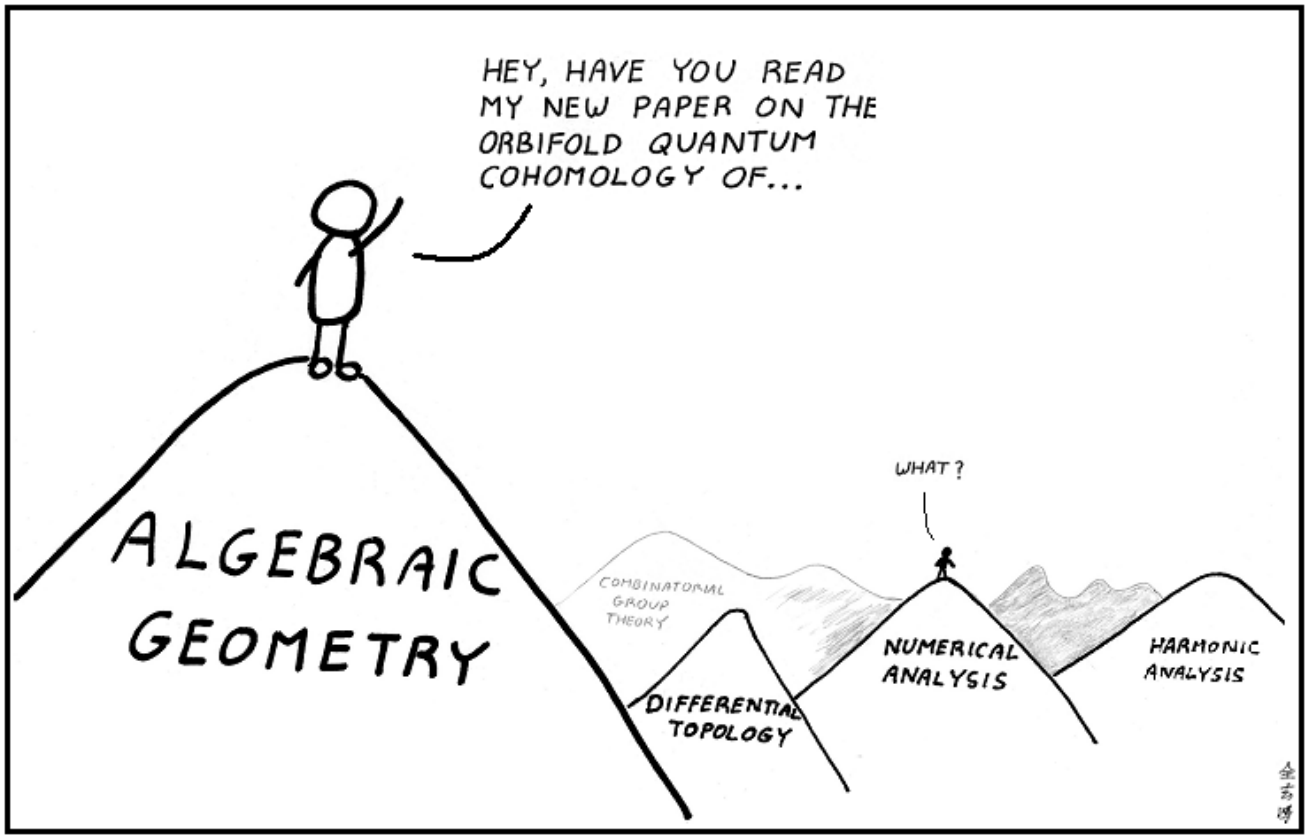
« La » mathématique n'existe pas

- Une unité de surface qui se craquelle très vite
- Des objets manipulés très différents
- Des techniques de preuves extrêmement variables

Même les langages de surface sont incomparables

- algèbre vs. analyse
- symboles vs. représentation pictoriale
- « modèle intuitif »

C'est pas moi qui le dit



The Landscape of Modern Mathematics

Abstruse Goose (CC-BY-NC 3.0)

C'est une feature, pas un bug

- Une sorte de Sapir-Whorf à l'envers
- Le langage **doit** évoluer pour mieux manipuler les concepts
- *Premier Contact* vs. platonisme ?

Loquor Ergo Est

C'est une feature, pas un bug

- Une sorte de Sapir-Whorf à l'envers
- Le langage **doit** évoluer pour mieux manipuler les concepts
- *Premier Contact* vs. platonisme ?

Cette diversité doit être portée à la formalisation des mathématiques

- Une condition *sine qua non* pour passer à l'échelle
- Langages de haut niveau $\Rightarrow$ faisabilité
- Nous en sommes encore aux cartes perforées

Loquor Ergo Est

C'est une feature, pas un bug

- Une sorte de Sapir-Whorf à l'envers
- Le langage **doit** évoluer pour mieux manipuler les concepts
- *Premier Contact* vs. platonisme ?

Cette diversité doit être portée à la formalisation des mathématiques

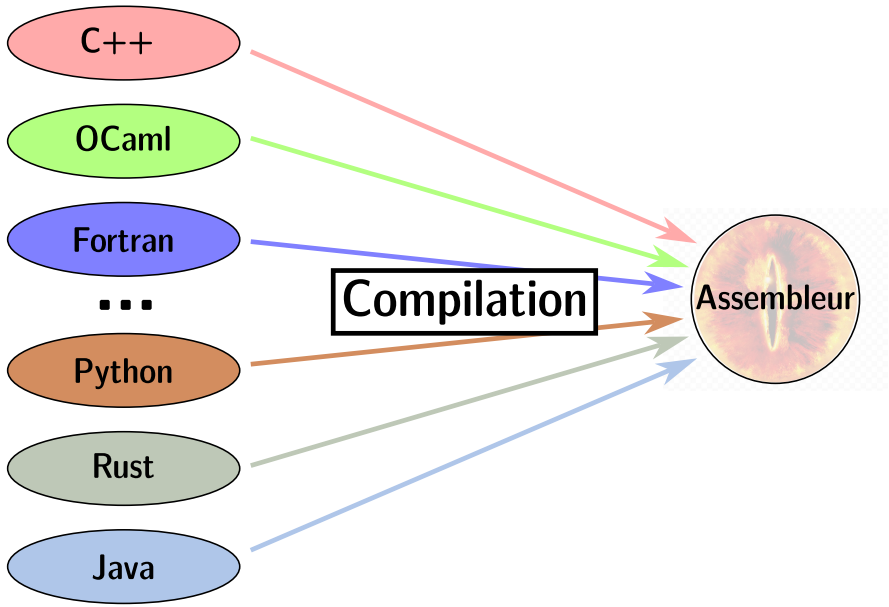
- Une condition *sine qua non* pour passer à l'échelle
- Langages de haut niveau $\Rightarrow$ faisabilité
- Nous en sommes encore aux cartes perforées

Comment éviter les schismes ?

Pluralité dans le monisme

En informatique

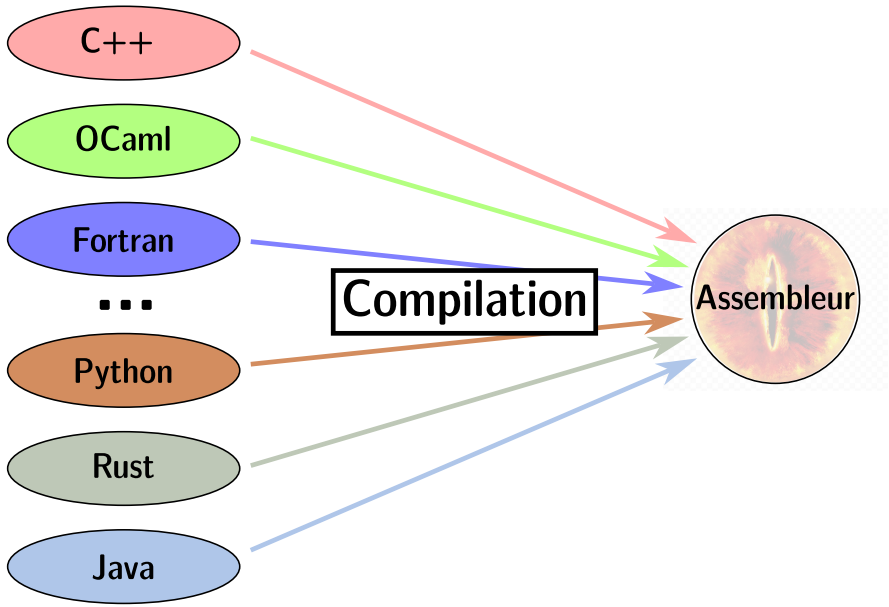
Le multilinguisme est résolu par la **compilation**



Pluralité dans le monisme

En informatique

Le multilinguisme est résolu par la **compilation**

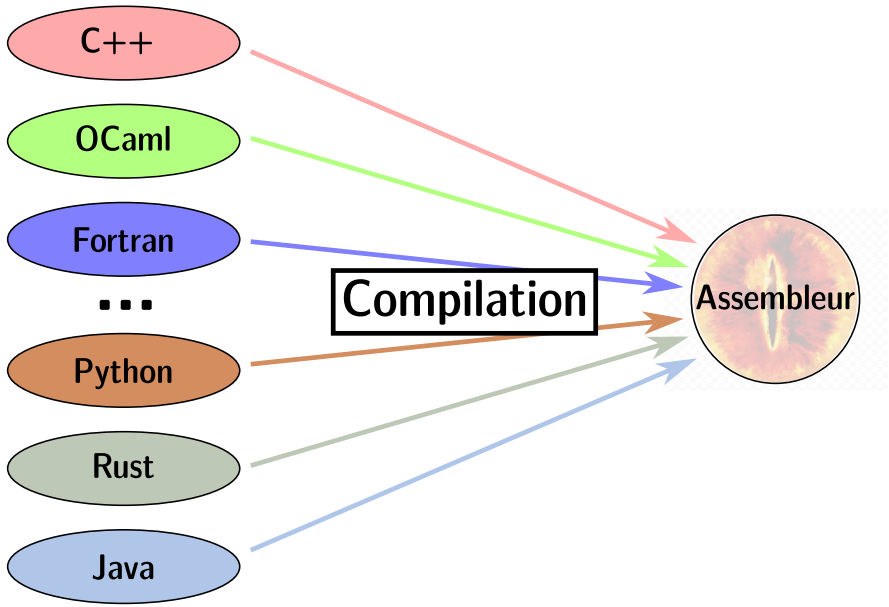


(* Photo non-contractuelle. Peut contenir des traces de fruits à coque et de bytecode VM.)

Pluralité dans le monisme

En informatique

Le multilinguisme est résolu par la **compilation**

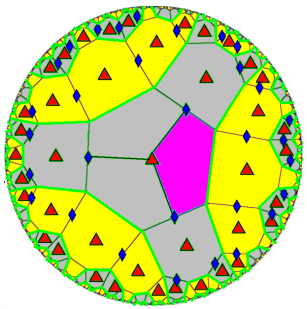


(* Photo non-contractuelle. Peut contenir des traces de fruits à coque et de bytecode VM.)

L'assembleur de notre machine sont nos fondations à nous

Le temps est venu de compiler...

...
à bec frisé



COHRQUI
Cohomologie des orbivariétés quantiques

COMPILATION

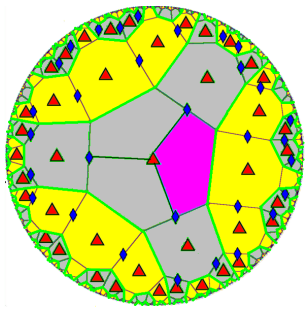
Théorie des types

...
à poils bleus

```
6a6:    66 2e 0f 1f 84 00 00    cs nopw 0x0(%rax,%rax,1)
6ad:    00 00 00
6b0:    4c 89 f0                mov    %r14,%rax
6b3:    4d 8d 76 f8             lea    -0x8(%r14),%r14
6b7:    4c 89 68 f8             mov    %r13,-0x8(%rax)
6bb:    4c 89 f8                mov    %r15,%rax
6be:    4d 8d 7f 04             lea    0x4(%r15),%r15
6c2:    48 63 10                movslq (%rax),%rdx
6c5:    48 8b 85 70 ff ff ff    mov    -0x90(%rbp),%rax
6cc:    4c 8b 2c d0             mov    (%rax,%rdx,8),%r13
6d0:    4c 89 f0                mov    %r14,%rax
6d3:    49 8b 16                mov    (%r14),%rdx
6d6:    49 83 c7 01             add    $0x1,%r15
6da:    4d 8d 76 e8             lea    -0x18(%r14),%r14
6de:    48 89 58 f8             mov    %rbx,-0x8(%rax)
6e2:    48 8b 9d 68 ff ff ff    mov    -0x98(%rbp),%rbx
6e9:    48 89 50 e8             mov    %rdx,-0x18(%rax)
6ed:    48 8d 54 1b 01          lea    0x1(%rbx,%rbx,1),%rdx
6f2:    4c 89 78 f0             mov    %r15,-0x10(%rax)
6f6:    4c 89 eb                mov    %r13,%rbx
6f9:    48 89 10                mov    %rdx,(%rax)
6fc:    4d 8b 7d 00             mov    0x0(%r13),%r15
700:    48 c7 85 68 ff ff ff    movq   $0x0,-0x98(%rbp)
707:    00 00 00 00
```

Reductio ad Assemblatorem

Le temps est venu de compiler...



...
à bec frisé

COHRQUI
Cohomologie des orbivariétés quantiques

COMPILATION

Théorie des types

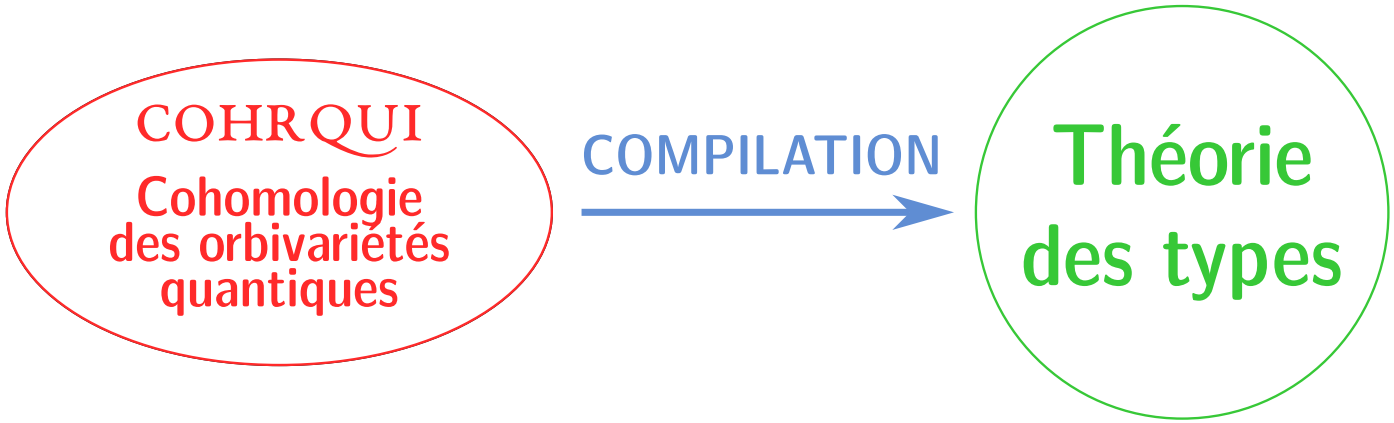
...
à poils bleus

```
6a6:    66 2e 0f 1f 84 00 00    cs nopw 0x0(%rax,%rax,1)
6ad:    00 00 00
6b0:    4c 89 f0                mov    %r14,%rax
6b3:    4d 8d 76 f8             lea    -0x8(%r14),%r14
6b7:    4c 89 68 f8             mov    %r13,-0x8(%rax)
6bb:    4c 89 f8                mov    %r15,%rax
6be:    4d 8d 7f 04             lea    0x4(%r15),%r15
6c2:    48 63 10                movslq (%rax),%rdx
6c5:    48 8b 85 70 ff ff ff    mov    -0x90(%rbp),%rax
6cc:    4c 8b 2c d0             mov    (%rax,%rdx,8),%r13
6d0:    4c 89 f0                mov    %r14,%rax
6d3:    49 8b 16                mov    (%r14),%rdx
6d6:    49 83 c7 01             add    $0x1,%r15
6da:    4d 8d 76 e8             lea    -0x18(%r14),%r14
6de:    48 89 58 f8             mov    %rbx,-0x8(%rax)
6e2:    48 8b 9d 68 ff ff ff    mov    -0x98(%rbp),%rbx
6e9:    48 89 50 e8             mov    %rdx,-0x18(%rax)
6ed:    48 8d 54 1b 01          lea    0x1(%rbx,%rbx,1),%rdx
6f2:    4c 89 78 f0             mov    %r15,-0x10(%rax)
6f6:    4c 89 eb                mov    %r13,%rbx
6f9:    48 89 10                mov    %rdx,(%rax)
6fc:    4d 8b 7d 00             mov    0x0(%r13),%r15
700:    48 c7 85 68 ff ff ff    movq   $0x0,-0x98(%rbp)
707:    00 00 00 00
```

... pour s'asseoir sur les fondations !

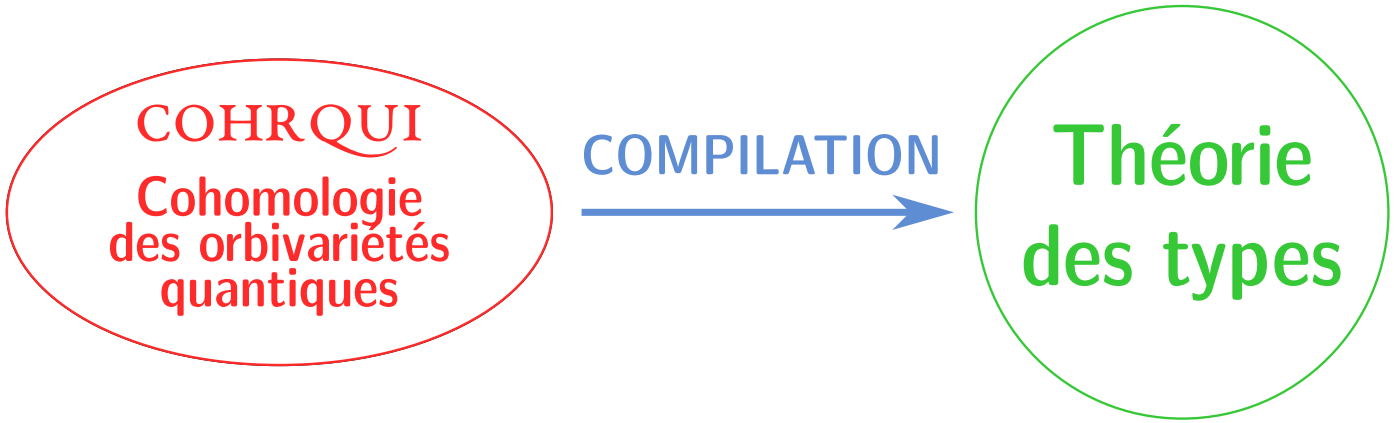
Mathématiques ⊗ informatique

Plus facile à dire qu'à faire



Mathématiques ⊗ informatique

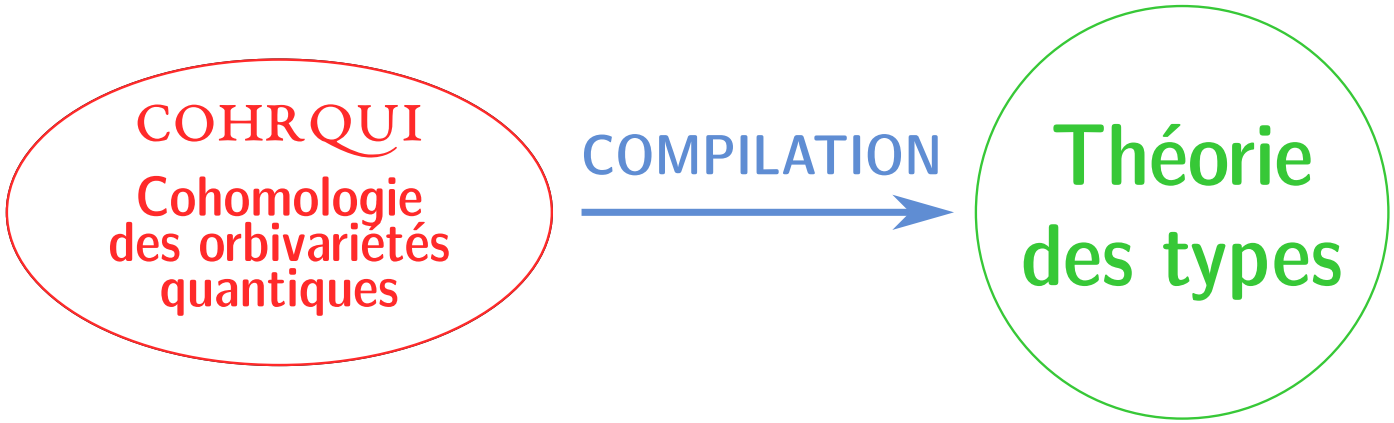
Plus facile à dire qu'à faire



- La source : c'est (en partie) le boulot des matheux
- La compilation : de vrais morceaux de mathématiques
- La théorie cible : un sujet à part entière

Mathématiques ⊗ informatique

Plus facile à dire qu'à faire



- La source : c'est (en partie) le boulot des matheux
- La compilation : de vrais morceaux de mathématiques
- La théorie cible : un sujet à part entière

Appel à contribution : du boulot pour les décennies à venir

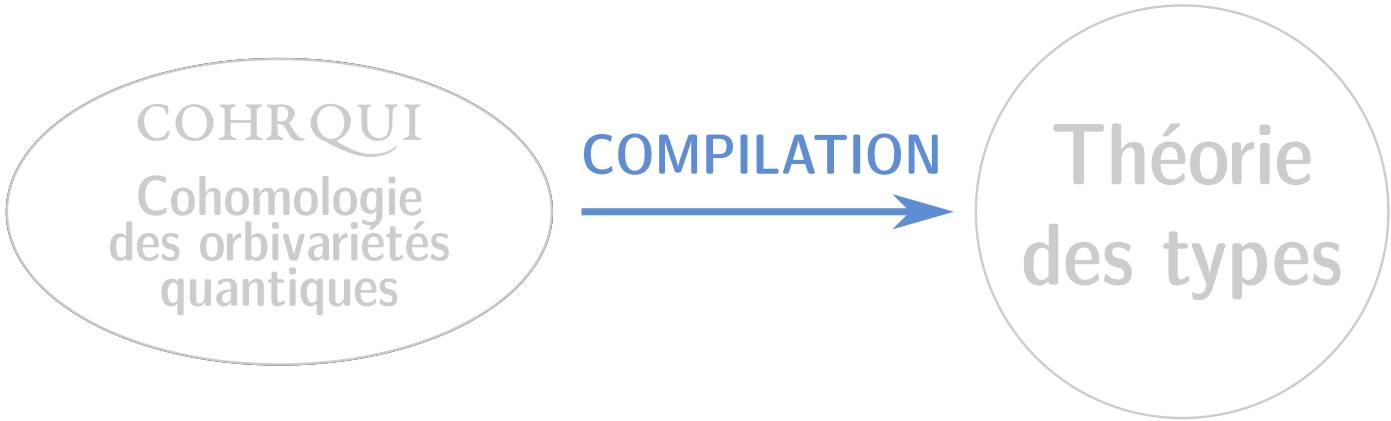
Moi, jamais je pipote

Dans la suite de cet exposé

Un échantillon absolument objectif des défis

1. Compilation : les calculs, c'est douloureux
2. Assembleur : CIC citron ou CIC orange ?
3. Pas de côté : et les assistants à la preuve dans tout ça

1. Les calculs, c'est douloureux



Rencontre du troisième type

Cas idéal : le langage source est **aussi** une théorie des types

Rencontre du troisième type

Cas idéal : le langage source est **aussi** une théorie des types

COMPILATION

Rencontre du troisième type

Cas idéal : le langage source est **aussi** une théorie des types

COMPILATION

- étend la puissance expressive
- permet de réutiliser les bibliothèques de la cible
- réduit l'impédance entre les théories

Rencontre du troisième type

Cas idéal : le langage source est **aussi** une théorie des types

COMPILATION

- étend la puissance expressive
- permet de réutiliser les bibliothèques de la cible
- réduit l'impédance entre les théories

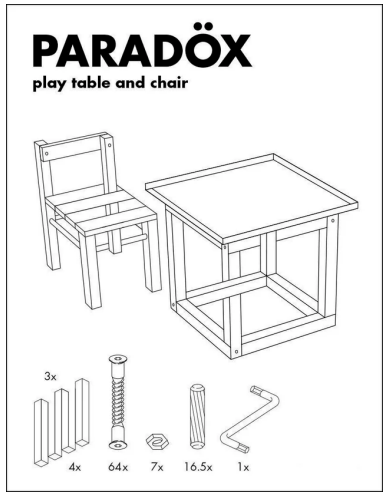
Mot-clef : approche **synthétique**

- Homotopie synthétique $\sim$ MLTT + univalence
- Calculabilité synthétique $\sim$ MLTT + CT
- ...

La logique mène à tout à condition d'en sortir

Bonne nouvelle 1 : techniques archi-connues dans les topoi

(Pré)faisceaux alias LA LOGIQUE IKEA

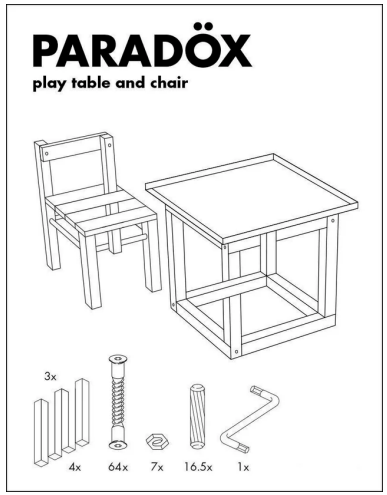


- usine à gaz catégorique $\Rightarrow$ compilation à la pogne
- universel pour les logiques « positives »
- ma logique en kit

La logique mène à tout à condition d'en sortir

Bonne nouvelle 1 : techniques archi-connues dans les topoi

(Pré)faisceaux alias LA LOGIQUE IKEA



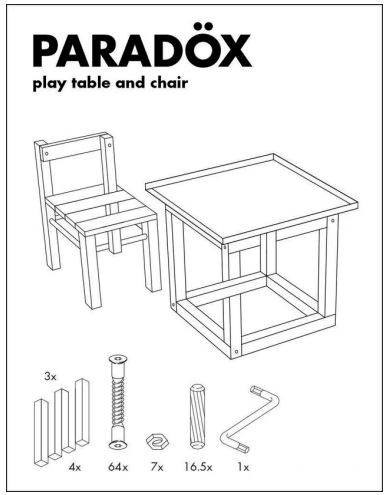
- usine à gaz catégorique $\Rightarrow$ compilation à la pogne
- universel pour les logiques « positives »
- ma logique en kit

Bonne nouvelle 2 : Théorie des types $\sim$ topoi

La logique mène à tout à condition d'en sortir

Bonne nouvelle 1 : techniques archi-connues dans les topoi

(Pré)faisceaux alias LA LOGIQUE IKEA



- usine à gaz catégorique $\Rightarrow$ compilation à la pogne
- universel pour les logiques « positives »
- ma logique en kit

Bonne nouvelle 2 : Théorie des types $\sim$ topoi

Conclusion ?

On peut impléménter les (pré)faiseaux en TT par compilation (?)

Et le désir s'accroit quand les faisceaux reculent

Conclusion ?

On peut impléménter les (pré)faiseaux en TT par compilation (?)

Et le désir s'accroit quand les faisceaux reculent

Conclusion ?

On peut implémenter [illegible] par compilation (?)

Et le désir s'accroit quand les faisceaux reculent

Conclusion ?

On peut implémenter l[illegible] (pré)faisceaux [illegible]TT par compilation (?)

Radotage

La théorie des types est un langage de programmation

Et le désir s'accroit quand les faisceaux reculent

Conclusion ?

On peut implémenter la (pré)faisceautique TT par compilation (?)

Radotage

La théorie des types est un langage de programmation

Deux notions d'égalité : **définitionnelle** et **propositionnelle**

$$\vdash M \equiv N : A \quad \text{vs} \quad \vdash P : M =_A N$$

jugement	type
calcul	logique

Coincident dans les topoi, pas en théorie des types !

Et le désir s'accroit quand les faisceaux reculent

Conclusion ?

On peut implémenter la (pré)faisceautique TT par compilation (?)

Radotage

La théorie des types est un langage de programmation

Deux notions d'égalité : **définitionnelle** et **propositionnelle**

$$\vdash M \equiv N : A \quad \text{vs} \quad \vdash P : M =_A N$$

jugement	type
calcul	logique

Coincident dans les topoi, pas en théorie des types !

La traduction de préfaisceaux repose sur cette identification !

La Strictitude

Solution 1 : toposifier la cible

- régression technique
- perte de propriétés critiques pour l'implémentation

La Strictitude

Solution 1 : toposifier la cible

- régression technique
- perte de propriétés critiques pour l'implémentation

Solution 2 : changer complètement la traduction

préfaisceaux ensemblistes $\Rightarrow$ types préfascistes

- inspirés par la théorie de la programmation
- préservent la conversion
- vraiment pas des préfaisceaux
- (remarque sur les faisceaux *intensionnellement* laissée vide)

La Strictitude

Solution 1 : toposifier la cible

- régression technique
- perte de propriétés critiques pour l'implémentation

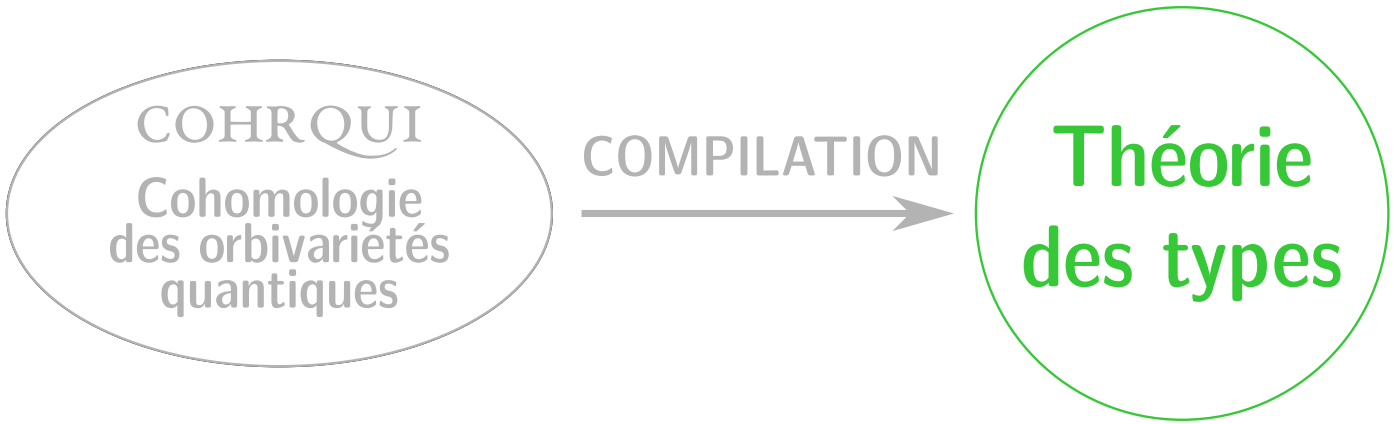
Solution 2 : changer complètement la traduction

préfaisceaux ensemblistes $\Rightarrow$ types préfascistes

- inspirés par la théorie de la programmation
- préservent la conversion
- vraiment pas des préfaisceaux
- (remarque sur les faisceaux *intensionnellement* laissée vide)

Il faut revisiter le foklore en profondeur !

2. CIC citron ou CIC orange ?



J'y suis, j'y reste

Pour s'asseoir sur les fondations, elles doivent être confortables

- Une théorie des types la plus expressive possible
- ... tout en maintenant de bonnes propriétés (cohérence, canonicité, décidabilité...)

J'y suis, j'y reste

Pour s'asseoir sur les fondations, elles doivent être confortables

- Une théorie des types la plus expressive possible
- ... tout en maintenant de bonnes propriétés (cohérence, canonicité, décidabilité...)

Deux points clairement en tension !

J'y suis, j'y reste

Pour s'asseoir sur les fondations, elles doivent être confortables

- Une théorie des types la plus expressive possible
- ... tout en maintenant de bonnes propriétés (cohérence, canonicité, décidabilité...)

Deux points clairement en tension !

- L'exploration de cet espace est le travail des informaticiens
- ... mais il doit être informé des besoins des sources
- Les fondations doivent être évaluées à l'aune de leur utilisabilité

J'y suis, j'y reste

Pour s'asseoir sur les fondations, elles doivent être confortables

- Une théorie des types la plus expressive possible
- ... tout en maintenant de bonnes propriétés (cohérence, canonicité, décidabilité...)

Deux points clairement en tension !

- L'exploration de cet espace est le travail des informaticiens
- ... mais il doit être informé des besoins des sources
- Les fondations doivent être évaluées à l'aune de leur utilisabilité

Au hasard : **Univalence ou UIP ?**

Les aventures de Poly

Et pourquoi pas les deux ?

Les aventures de Poly

Et pourquoi pas les deux ?

- Surtout ne pas répondre à la question
- Une extension conceptuellement triviale qui passe à l'échelle
- Le multivers à portée de clavier

Les aventures de Poly

Et pourquoi pas les deux ?

- Surtout ne pas répondre à la question
- Une extension conceptuellement triviale qui passe à l'échelle
- Le multivers à portée de clavier

via **le polymorphisme de sortes**

- Une addition récente à Rocq
- Compatibilité arrière par construction
- Gain d'expressivité linguistique, conservatif logiquement

Tu erreras désormais dans un monde inconnu

Rocq a déjà une notion d'univers parallèles : les sortes

Tu erreras désormais dans un monde inconnu

Rocq a déjà une notion d'univers parallèles : les sortes

- Prop / Type / SProp
- Une distinction à l'origine introduite pour l'extraction
- Chaque univers capture une logique différente

Tu erreras désormais dans un monde inconnu

Rocq a déjà une notion d'univers parallèles : les sortes

- Prop / Type / SProp
- Une distinction à l'origine introduite pour l'extraction
- Chaque univers capture une logique différente

On peut rajouter de nouvelles sortes

- Par exemple des types fibrants, des types stricts...
- Rocq est agnostique à priori sur le contenu de ces sortes

Tu erreras désormais dans un monde inconnu

Rocq a déjà une notion d'univers parallèles : les sortes

- Prop / Type / SProp
- Une distinction à l'origine introduite pour l'extraction
- Chaque univers capture une logique différente

On peut rajouter de nouvelles sortes

- Par exemple des types fibrants, des types stricts...
- Rocq est agnostique à priori sur le contenu de ces sortes

Nihil novis

cf. la théorie des types à deux niveaux

C'est une révolution™

La notion de variable

C'est une révolution™

La notion de variable

- On peut maintenant quantifier sur les sortes
- Permet de partager le code
- Une théorie polymorphe est valide dans tous les univers

C'est une révolution™

La notion de variable

- On peut maintenant quantifier sur les sortes
- Permet de partager le code
- Une théorie polymorphe est valide dans tous les univers

La question difficile : l'interaction entre sortes

C'est une révolution™

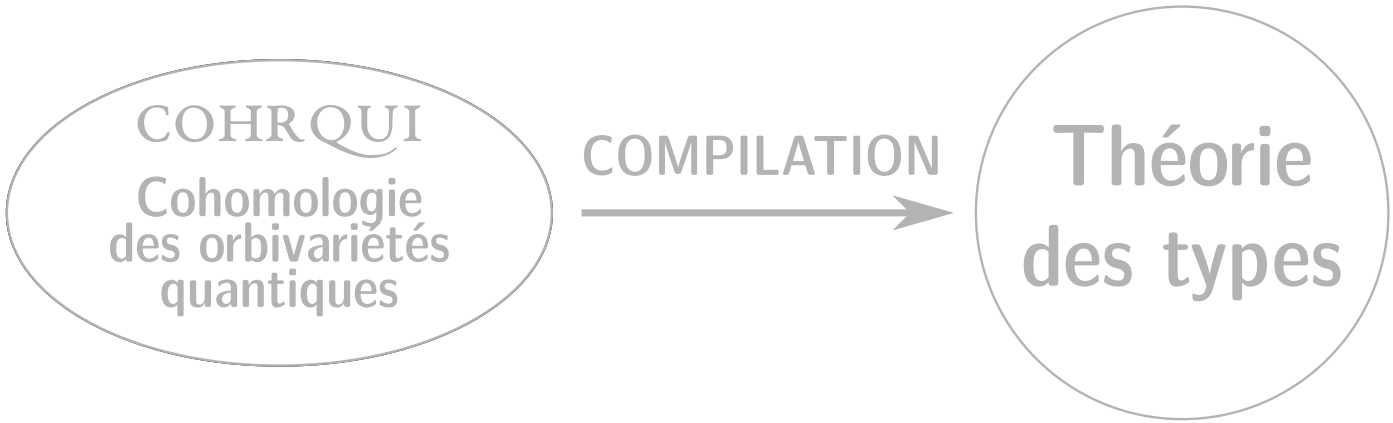
La notion de variable

- On peut maintenant quantifier sur les sortes
- Permet de partager le code
- Une théorie polymorphe est valide dans tous les univers

La question difficile : l'interaction entre sortes

Ça bouge encore dans les fondations !

Et les assistants à la preuve dans tout ça ?

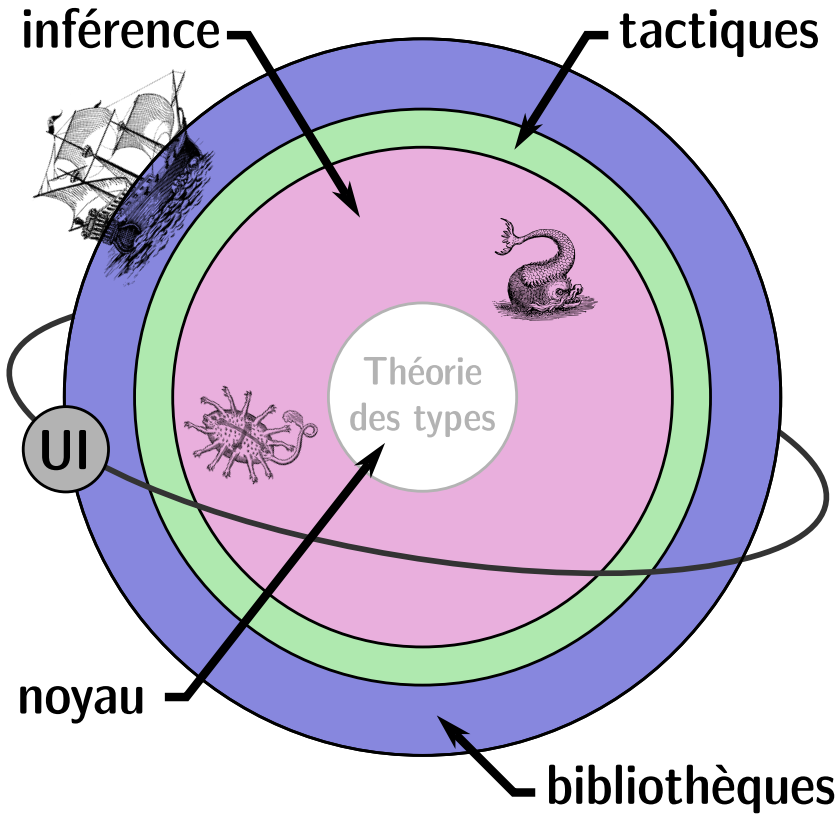


J'ai menti

Une vision très biaisée !

J'ai menti

Une vision très biaisée !



Here Be Dragons

Un assistant à la preuve, c'est bien plus que ça

Rocqblox

Encore une fois, les informaticiens ont déjà frappé

Rocqblox

Encore une fois, les informaticiens ont déjà frappé

The IRIS framework

- Un assistant à la preuve *au-dessus* de Rocq
- Bas-niveau : logique de séparation via des préfaisceaux
- Haut-niveau : un *mod* de preuve pour votre jeu vidéo favori

Le *mod* à l'IT

Les couches hautes sont entièrement redéfinies

```
  ============================
  "Hinv" : inv N (lock_inv γ l R)
  ------------------------------------□
  "HΦ" : ∀ b : bool, (if b then locked γ ∗ R else True) -∗ Φ #b
  "Hl" : ▷ l ↦ #true
  "HR" : ▷ True
  ------------------------------------∗
  WP CmpXchg #l #false #true @ T ∖ ↑N {{ v, |={T ∖ ↑N}=> ▷ lock_inv γ l R ∗ WP Snd v {{ v, Φ v }} }}

goal 2 (ID 1116) is:
 "Hinv" : inv N (lock_inv γ l R)
 ------------------------------------□
 "HΦ" : ∀ b : bool, (if b then locked γ ∗ R else True) -∗ Φ #b
 "Hl" : ▷ l ↦ #false
 "HR" : ▷ (token γ ∗ R)
 ------------------------------------∗
 WP CmpXchg #l #false #true @ T ∖ ↑N {{ v, |={T ∖ ↑N}=> ▷ lock_inv γ l R ∗ WP Snd v {{ v, Φ v }} }}
```

- Des notations de partout
- Un langage de tactiques ad-hoc
- Même le mode de preuve est un encodage !

Songe d'une nuit de MLTT

Je rêve d'une myriade de sur-assistants à la preuve

Songe d'une nuit de MLTT

Je rêve d'une myriade de sur-assistants à la preuve

À votre tour de faire fleurir les *mods*

Conclusion

- Le multilinguisme c'est la norme, en maths comme en info
- Il doit devenir systématique en théorie des types
- Un travail au long cours
- ... à l'interface entre mathématiques et informatique

Merci de votre attention.