Logipedia :
vers un wikipédia des démonstrations formelles

I. La révolution des formats

Jusque dans les années 1980 $\pm\varepsilon$

On écrit un logiciel de traitement de texte
Chaque caractère peut être en italique ou en romain...

On décide

- de coder chaque caractère sur deux octets : l'un pour son code ascii, l'autre pour sa fonte
- ou de coder le passage en italique par le code 128, le passage en romain par le code 129

En écrivant le logiciel on définit involontairement un format

Et ensuite

On définit d'abord un format : par exemple HTML

Puis les logiciels se soumettent au format

Une idée ancienne

- 1954 : Fortran, mais surtout 1958 : Algol
- 1963 : ascii
- 1974 : tcp / ip
- 1990 : http, html
- 1991 : unicode
- ...

Mais qui n'est pas (encore) arrivée dans le domaine des démonstrations formelles

« Une démonstration Coq du théorème des quatre couleurs »

Les problèmes que cela pose

Interopérabilité

Durabilité

II. Les mathématiques inverses

Pourquoi est-ce plus difficile pour les démonstrations formelles ?

Parce qu'on ne peut pas aller trop loin

Géométrie euclidienne $\not\leftrightarrow$ géométrie riemannienne

ZF $\not\leftrightarrow$ ZFC

Mais...

Une démonstration en ZF peut être « traduite » en ZFC

Une démonstration en ZFC qui n'utilise pas l'axiome du choix peut être « traduite » en ZF

Transformer les démonstrations

Il existe une base de $\mathbb{R}^2$

- par le théorème de la base incomplète (axiome du choix)
- $\langle 1, 0 \rangle$, $\langle 0, 1 \rangle$

Automatiquement (par exemple : constructivisation) ou à la main

Une reformulation du projet des mathématiques inverses

- Démonstrations formelles et non papier-crayon-LaTeX
- Sur des théories expressives (théories des ensembles, théories des types...) et non des fragments de l'arithmétique
- Analyser les démonstrations avant de (possiblement) les transformer

Pourquoi cette interopérabilité ZF / ZFC est-elle possible ?

Parce que ZF et ZFC sont exprimés dans un même cadre logique : la logique des prédicats

En logique des prédicats une théorie est exprimée par plusieurs axiomes

Permet la question : quels axiomes sont utilisés dans la démonstration π

III. La révolution des cadres logiques

Depuis Euclide : la géométrie, la théorie des ensembles... chacun son langage, chacun sa notion de démonstration...

Hilbert et Ackermann (1928) : la logique des prédicats

Un cadre logique dans lequel on peut définir des théories (géométrie, théorie des ensembles...)
Pour chacune : des symboles et des axiomes

Une révolution brève

Logique des prédicats : simplification de la Théorie des types (Russell, Whitehead, Ramsey)
Mais pas de reformulation de la Théorie des types dans la logique des prédicats

Dès 1940 (Church) : une nouvelle formulation de la Théorie des types difficile (impossible ?) à exprimer dans la logique des prédicats

1970, 1985... La théorie des types de Martin-Löf, le Calcul des constructions... pas dans la logique des prédicats

Trois attitudes

- Considérer la notion de cadre logique comme morte
- Exprimer la théorie des types de Russell, de Church, de Martin-Löf, le Calcul des constructions... dans la logique des prédicats de gré ou de force (Henkin, Davis, D...)
- Étendre la logique des prédicats en un meilleur cadre logique

Les limites de la logique des prédicats

- Pas de variables liées ($x \mapsto x$)
- Pas de bonne syntaxe pour les démonstrations
- Pas de notion de calcul
- Pas de bonne notion de coupure
- Classique et non constructive

De nouveaux cadres logiques

- Pas de variables liées ($x \mapsto x$) : λ-Prolog, $\lambda\Pi$-calcul
- Pas de bonne syntaxe pour les démonstrations : $\lambda\Pi$-calcul
- Pas de notion de calcul : Déduction modulo théorie
- Pas de bonne notion de coupure : Déduction modulo théorie
- Classique et non constructif : Logique œcuménique

Le $\lambda\Pi$-calcul modulo théorie qui les généralise tous

Dedukti

Une implémentation du $\lambda\Pi$-calcul modulo théorie

Uniquement un vérificateur de démonstrations
Réimplémentable en deux semaines

La Théorie des types simples en Dedukti

$$
\begin{array}{rcl}
type & : & Type \\
\eta & : & type \to Type \\
o & : & type \\
nat & : & type \\
arrow & : & type \to type \to type \\
\varepsilon & : & (\eta\ o) \to Type \\
\Rightarrow & : & (\eta\ o) \to (\eta\ o) \to (\eta\ o) \\
\forall & : & \Pi a : type\ (((\eta\ a) \to (\eta\ o)) \to (\eta\ o))
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (arrow\ x\ y)) & \longrightarrow & (\eta\ x) \to (\eta\ y) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z))
\end{array}
$$

Le Calcul des constructions en Dedukti

$$
\begin{array}{rcl}
\mathit{type} & : & \mathit{Type} \\
\eta & : & \mathit{type} \to \mathit{Type} \\
o & : & \mathit{type} \\
\mathit{nat} & : & \mathit{type} \\
\mathit{arrow} & : & \Pi x : \mathit{type}\ ((\eta\ x) \to \mathit{type}) \to \mathit{type}) \\
\varepsilon & : & (\eta\ o) \to \mathit{Type} \\
\Rightarrow & : & \Pi x : (\eta\ o)\ (((\varepsilon\ x) \to (\eta\ o)) \to (\eta\ o)) \\
\forall & : & \Pi x : \mathit{type}\ (((\eta\ x) \to (\eta\ o)) \to (\eta\ o)) \\
\pi & : & \Pi x : (\eta\ o)\ (((\varepsilon\ x) \to \mathit{type}) \to \mathit{type})
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (\mathit{arrow}\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\eta\ (y\ z)) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & \Pi z : (\varepsilon\ x)\ (\varepsilon\ (y\ z)) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z)) \\
(\eta\ (\pi\ x\ y)) & \longrightarrow & \Pi z : (\varepsilon\ x)\ (\eta\ (y\ z))
\end{array}
$$

Les mathématiques inverses en Dedukti

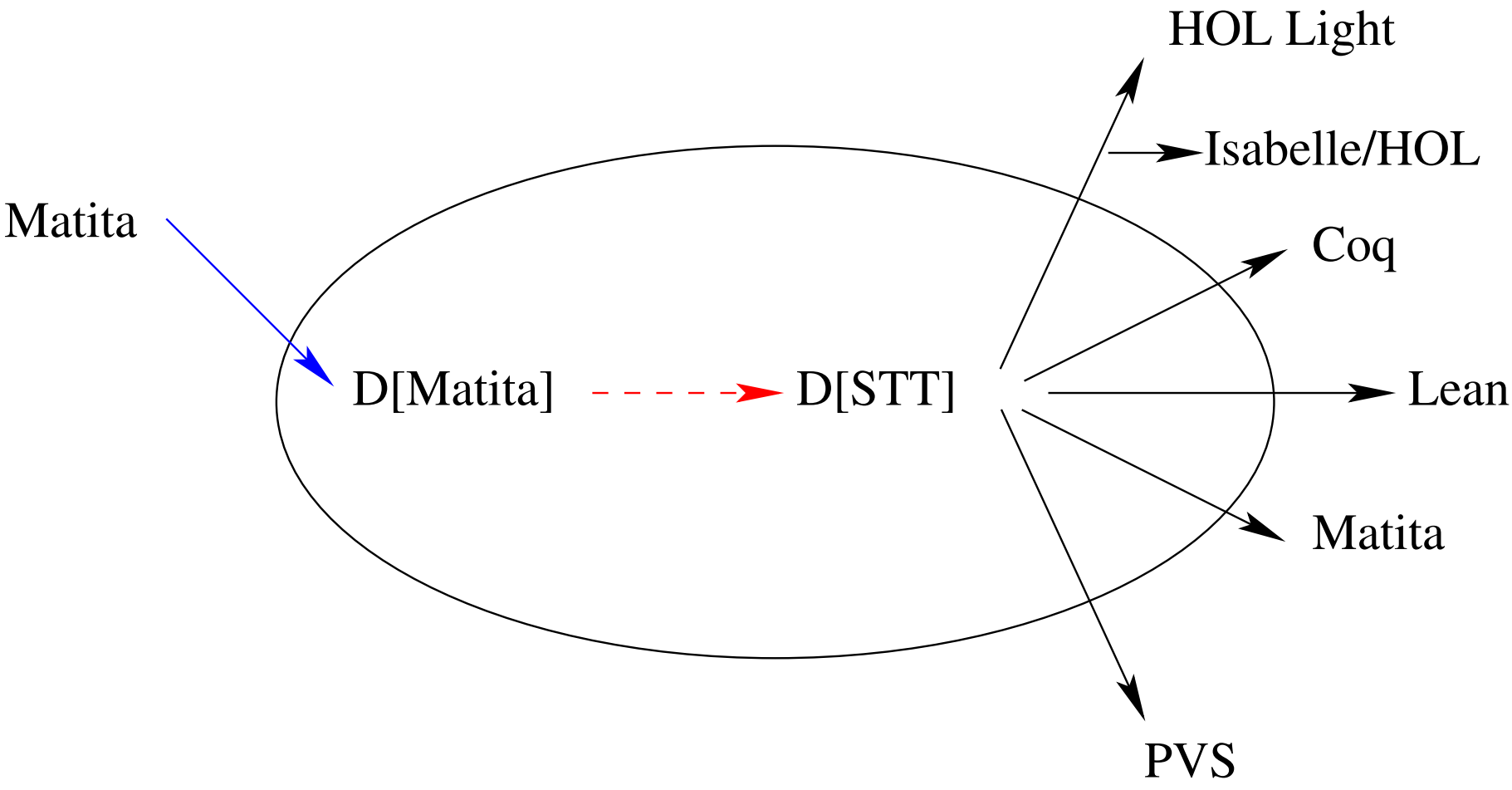
- arrow dépendente dans le Calcul des constructions mais pas dans la Théorie des types simples
- idem pour $\Rightarrow$
- un symbole supplémentaire π dans le Calcul des constructions : exprimer des fonctions des démonstrations dans les termes

Toutes les démonstrations de la Théorie des types simples peuvent être traduites dans le Calcul des constructions

Les démonstrations dans le Calcul des constructions qui n'utilisent pas ces trois fonctionnalités peuvent être traduites en Théorie des types simples

Par exemple : toutes les démonstrations de la librairie d'arithmétique de Matita

Interopérabilité



Pourquoi cela marche-t-il si bien ?

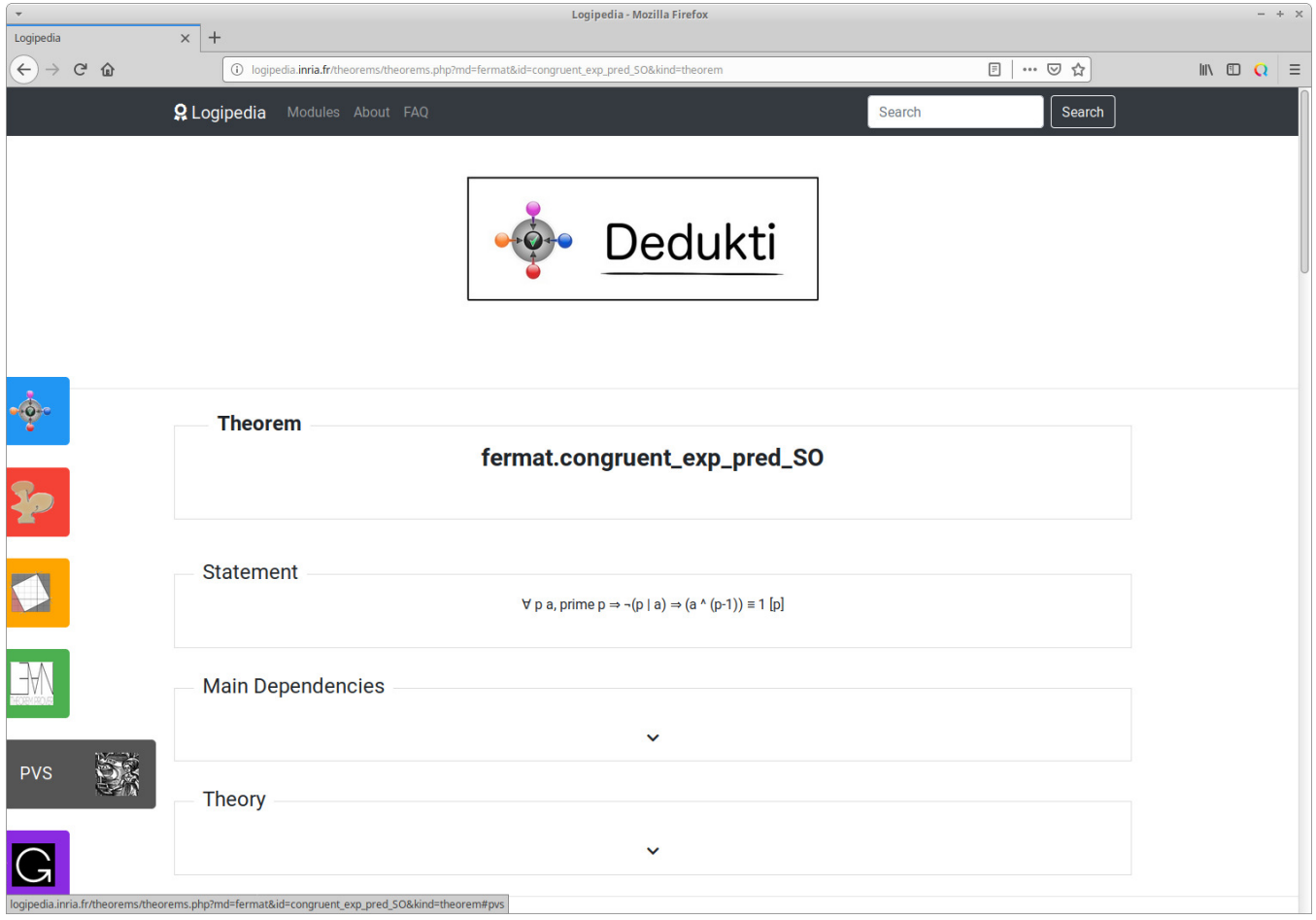
Parce que les systèmes de démonstrations implémentent des théories très expressives... dont on n'utilise qu'une petite partie dans les démonstrations

Trois constatations empiriques antérieures

- Systèmes de démonstrations : théories très différentes, mais librairies très proches
- Les mathématiciens s'intéressent peu aux axiomes utilisés dans une démonstration : n'importe lesquels semblent leur convenir
- Les mathématiciens ne s'intéressent même pas aux définitions (il faut bien construire les réels, mais peu importe comment)

IV. De Dedukti à Logipedia

Regrouper les démonstrations dans une unique base de données



http://logipedia.science

Importer et exporter des démonstrations

Une fois défini, en Dedukti, la logique d'un système de démonstrations (logique des prédicats, HOL Light, Matita...)
Importer et exporter une démonstration de T dans $D[T]$ devrait être simple

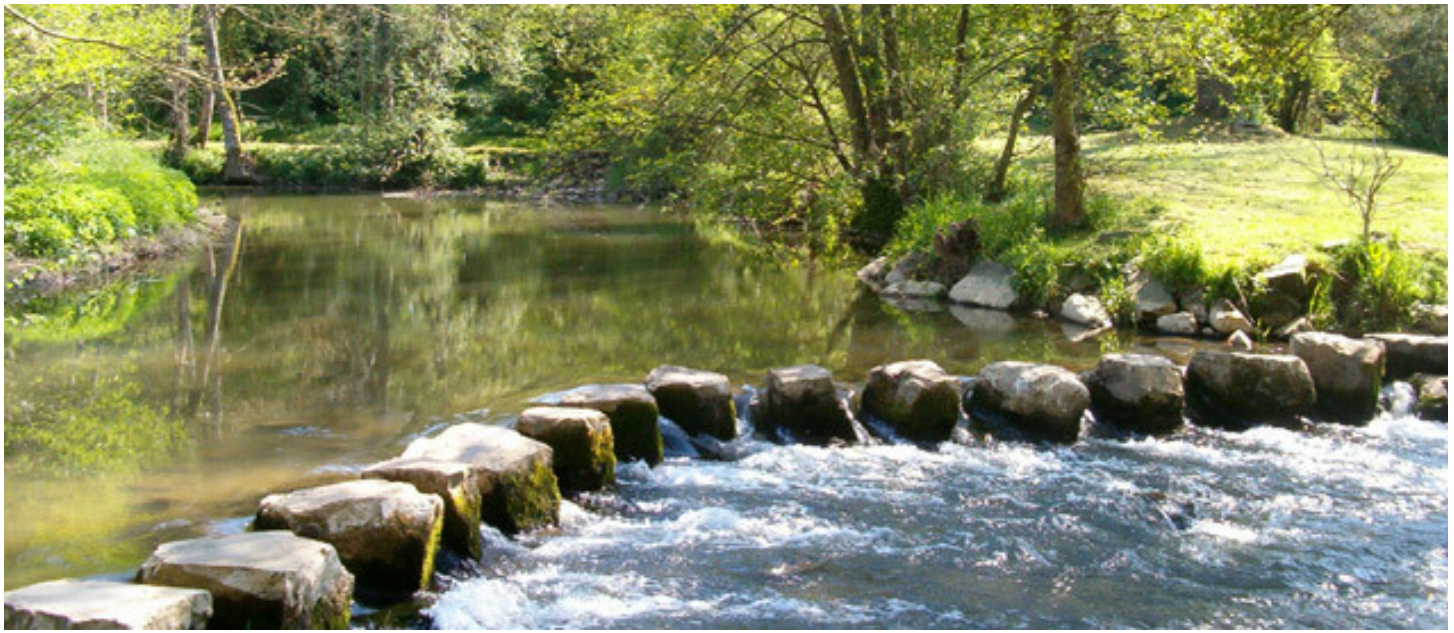
Mais les systèmes sont très divers sur ce plan

- Automath, Matita, Coq, Lean, Agda, Zenon, iProver, ArchSAT... ont une syntaxe pour les démonstrations : il suffit d'un analyseur syntaxique et d'un *printer*
- LCF, HOL, HOL Light (petits noyaux) faciles à instrumenter
- PVS, Mizar... (gros noyaux) difficiles à instrumenter

Pour les plus difficiles

Exporter une trace et combler les vides avec un système de démonstration automatique

Quand il n'y a pas de pont



Il faut passer à gué

Encore beaucoup à faire

- Expression dans Dedukti de Coq, PVS, Mizar...
- Alignement des concepts, élimination des isomorphismes...
- Transformation des démonstrations (mathématiques inverses)
- Interface, BD, recherche...

Mais déjà des résultats concrets

Alors que Qed (1993) n'était pas allé très loin

- Meilleure compréhension des théories implémentées dans les systèmes de démonstrations
- Un nouveau cadre logique pour exprimer ces théories
- Qui permet une meilleure analyse individuelle des démonstrations

L’interopérabilité n’est pas uniquement une question de comités, de négociations et de standards. C’est, avant tout, une question de recherche