Que faire avec la bibliothèque standard de Coq ?

Un aveu d'échec

Des bibliothèques hétérogènes, avec peu de cohérence, ni globale, ni interne

– dans le nommage (différents noms pour la même propriété, noms des variables liées, ...)
– dans la manière d'énoncer les théorèmes (sens des égalités, ordre et place des variables liées, ...)
– dans le design (p.ex. $\leq$ proposition dans `Arith`, fonction dans `ZArith`, ...)
– des redondances (propriétés des relations à la fois dans `Sets` et `Relations`, ...)
– asymétrie dans les lemmes (une propriété peut être exprimée sur $\leq$, une autre sur $<$, une sur `nat`, l'autre sur `N`, ...)
– dans le chaînage des bibliothèques (`Require`)
– dans les fonctionnalités associées (arguments implicites, hints, ...)

Un manque de moyens humains et de cadre pour étendre les bibliothèques

– de nombreux utilisateurs proposent des amendements, de nouveaux lemmes, de nouvelles bibliothèques (Frédéric Blanqui avec les listes de taille donnée, Xavier Leroy avec les listes, Nimègue, ...), mais pas de disponibilité matérielle pour

arbitrer entre ces propositions et pour s'assurer que cela n'engendre pas plus de cohérence

– au bilan, les utilisateurs refont souvent les mêmes choses dans leur coin (cf les Zaux et listaux dans les contribs, ...)

Comment avancer ?

Quelles solutions pour des bibliothèques vivantes ?

- mise en place de lignes directrices dans lesquelles doivent s'inscrire les extensions proposées par les utilisateurs ?
- distribution des responsabilités autour des bibliothèques ?
  - utilisation d'un gestionnaire de version distribué pour que la bibliothèque standard soit en fait une collection de sous-archives autonomes chacune hébergées par leur responsable ?
  - des responsables parmi les utilisateurs ?

Quelles solutions pour uniformiser la bibliothèque actuelle ?

- renoncer et repartir de zéro en déplaçant les biblios actuelles vers les contribs ? avec quels moyens matériels ? avec quelle politique de validation des nouvelles bibliothèques ?
- comment gérer la compatibilité ? modules de compatibilité ? une nouvelle phase de traduction de la V8 vers une V9 ?

Bloquer les personnes volontaires sur une île déserte jusqu'à ce qu'un projet de bibliothèque uniforme et de principes compatibles avec les expériences de chacun soit adopté ?

Objectifs pour aujourd'hui ?

- mode de fonctionnement général (centralisation, modèle distribué, mode mixte) ?
- mode de validation des bibliothèques (sur une base globale, sur une base de délégation de responsabilités, éventuellement hors é.d.c.) ?
- statuer sur la place et les objectifs de chaque bibliothèque prise individuellement
- avancer en particulier sur les objectifs pour une bibliothèque arithmétique (place de `Numbers`, place de `Arith`, ...)
- principe d'une charte pour faciliter l'uniformité et les extensions venant de l'extérieur
- avancer sur les solutions pour uniformiser la bibliothèque actuelle

Solutions techniques pour la compatibilité ?

À court terme

– utilisation de notations « syntaxiques » pour les changements de nom ? (dans le corps du .v, en fin de fichier ? les documenter dans coqdoc ?)
– table interne de redirection pour les changements de noms ou le réordonnement des variables liés (comme pour le passage V7→V8) ?

À plus long terme

– Avancer sur le principe de raisonnement modulo équivalence (ainsi, à long terme, $\leq$ ne serait plus une définition particulière mais une classe de définitions équivalentes dont un des représentant serait utilisé selon le contexte d'utilisation) ?

Quelques aspects d'une politique globale de conception des bibliothèques

- toutes les définitions ensembles puis les propriétés ou alternance définitions et propriétés ?
- où mettre les Hints ? (séparément, ce qui permet d'avoir des fichiers arith1, arith2, arith3, ... ou dans la foulée des lemmes, mais les bases de hints deviennent rigides)
- quelle stratégie pour les implicites ?
- politique de nommage et de notation ?
  - étendre le principe actuel (cf document « stdlib project ») mais vite limité dès qu'on sort des propriétés algébriques standard
  - comment gérer la présence de plusieurs implantations de $\mathbb{N}$?
  - faut-il donner à `Arith` un statut « pédagogique » à part ? (par exemple garder O et S plutôt que zero et succ, ou Nat.O et Nat.S)
  - réfléchir à une notion d'algèbre de noms (style « distributivity of $\times$ over $+$ on N » étant un nom à part entière)
- place des propriétés calculatoires ? p.ex. : vers quoi pointe la notation $\leq$? un

meilleur support pour raisonner avec `sumbool` ?
– place de l'égalité hétérogéne ? un meilleur support pour manipuler `JMeq` ? pour manipuler les coercions de types à base de `eq_rect` ?
– place à donner aux classes de type, aux coercions, ...

Message de Pierre Courtieu

Un point à défendre : L'unification derrière les modules types. Toutes les librairies standards (au moins celles de type "Data") devraient être présentées sous forme de modules instanciant des module type. Toute librairie devrait avoir cette forme et fournir les foncteurs permettant de traduire vers toutes les autres structures, le graphe devant toujours être aussi complet que possible. Bien sûr le travail d'ajout d'une librairie ira en explosant à cause des traductions à fournir, mais cela permettra aussi de ne pas partir dans tous les sens.

Les changements de représentation (au milieu des preuves par exemple) seraient facilités ou inutiles, la conception de tactiques basées sur les lib standards seraient plus universelles etc...

Pierre Letouzey fait un super boulot dans ce sens (FSet, et j'imagine numbers part dans la même direction) et je pense que cela devrait être érigé plus ou moins en principe de développement des librairies standards.