Bibliothèques de Composants Mathématiques

Journée ADT Coq - Bibliothèques

Assia Mahboubi

INRIA - TypiCal
INRIA - Microsoft Research Joint Centre

11 Décembre 2008

Objectifs du corpus de bibliothèques

- Aujourd'hui les bibliothèques formalisées :
  - partagent peu de structures de données
  - sont souvent munies d'outils d'automatisation ad'hoc
  - sont peu réutilisées d'un développement à l'autre

Objectifs du corpus de bibliothèques

- Aujourd'hui les bibliothèques formalisées :
  - partagent peu de structures de données
  - sont souvent munies d'outils d'automatisation ad'hoc
  - sont peu réutilisées d'un développement à l'autre
- La correspondance de Curry-Howard en pratique:
  - Quelles structures de données ?
  - Quelles abstractions ?
  - Quelle automatisation ?
  - Quelle modularité ?
  - Quelle discipline ?

  pour les bibliothèques de preuves formelles.

Motivations concrètes et point de départ

- Terrain de jeu : formaliser une preuve du th. de Feit-Thompson (1963)
  - Théories mathématiques avancées, variées et inter-dépendantes
  - Discrètes, dénombrables, algébriques
  - Beaucoup de bas niveau très générique

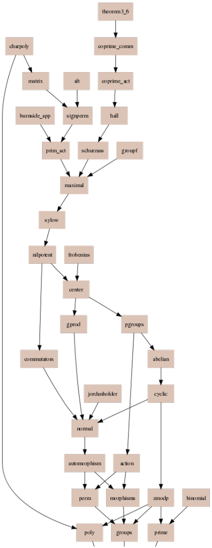
Motivations concrètes et point de départ

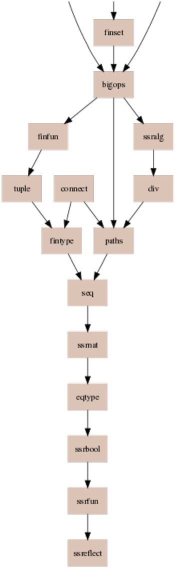
- Terrain de jeu : formaliser une preuve du th. de Feit-Thompson (1963)
  - Théories mathématiques avancées, variées et inter-dépendantes
  - Discrètes, dénombrables, algébriques
  - Beaucoup de bas niveau très générique
- Point de départ : les bibliothèques de la preuve du th. des 4 couleurs

Motivations concrètes et point de départ

- Terrain de jeu : formaliser une preuve du th. de Feit-Thompson (1963)
  - Théories mathématiques avancées, variées et inter-dépendantes
  - Discrètes, dénombrables, algébriques
  - Beaucoup de bas niveau très générique
- Point de départ : les bibliothèques de la preuve du th. des 4 couleurs
- Objet de cet exposé :
  - état des lieux du développement
  - pourquoi les bibliothèques sont agréables à (ré)utiliser

Contenu actuel de l'archive





Contenu actuel de l'archive

- Infrastructure bas niveau
  - entiers, booléens, fonctions, relations, ensembles finis, ...

Contenu actuel de l'archive

- Infrastructure bas niveau
  - entiers, booléens, fonctions, relations, ensembles finis, ...
- Infrastructure algébrique
  - arithmétique, polynômes, matrices, ...
  - Hiérarchie algébrique : des groupes commutatifs aux corps algébriquement clos

Contenu actuel de l'archive

- Infrastructure bas niveau
  - entiers, booléens, fonctions, relations, ensembles finis, ...
- Infrastructure algébrique
  - arithmétique, polynômes, matrices, ...
  - Hiérarchie algébrique : des groupes commutatifs aux corps algébriquement clos
- Théories algébriques
  - Groupes finis : (iso)morphismes, actions, quotients, Sylow, permutations, $\mathbb{Z}/n\mathbb{Z}$, Jordan-Hölder, ...
  - Algèbre linéaire/multi-linéaire : Maschke, théorie du déterminant, Cayley-Hamilton, pivot de Gauss,...

Contenu actuel de l'archive

- Infrastructure bas niveau
  - entiers, booléens, fonctions, relations, ensembles finis, ...
- Infrastructure algébrique
  - arithmétique, polynômes, matrices, ...
  - Hiérarchie algébrique : des groupes commutatifs aux corps algébriquement clos
- Théories algébriques
  - Groupes finis : (iso)morphismes, actions, quotients, Sylow, permutations, $\mathbb{Z}/n\mathbb{Z}$, Jordan-Hölder, ...
  - Algèbre linéaire/multi-linéaire : Maschke, théorie du déterminant, Cayley-Hamilton, pivot de Gauss,...
- Théories plus spécifiques à la preuve de Feit-Thompson

Un peu d'algèbre

- En profondeur, des choix très techniques, dictés par toute la construction supportée au dessus

Un peu d'algèbre

- En profondeur, des choix très techniques, dictés par toute la construction supportée au dessus
- Chaque anneau d'infrastructure peut être considéré comme “ étanche”

Un peu d'algèbre

- En profondeur, des choix très techniques, dictés par toute la construction supportée au dessus
- Chaque anneau d'infrastructure peut être considéré comme " étanche"
- Des bibliothèques faciles à prendre en main

Un peu d'algèbre

- En profondeur, des choix très techniques, dictés par toute la construction supportée au dessus
- Chaque anneau d'infrastructure peut être considéré comme “ étanche”
- Des bibliothèques faciles à prendre en main

Démo : Matrices

Ingrédients

Éléments de bilan d'une expérience de développement "autarcique":

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

Ingrédients

Éléments de bilan d'une expérience de développement "autarcique":

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

Démo : une version révisée d'Arith

Ingrédients

Éléments de bilan d'une expérience de développement “autarcique”:

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

  Démo : une version révisée d'Arith
- Discipline et révisions successives doivent assurer:
  - Cohérence dans les choix de noms
  - Systématisme dans la présence des lemmes
  - Uniformité des notations, dans leur définition et leur choix

Ingrédients

Éléments de bilan d'une expérience de développement "autarcique":

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

  Démo : une version révisée d'Arith
- Discipline et révisions successives doivent assurer:
  - Cohérence dans les choix de noms
  - Systématisme dans la présence des lemmes
  - Uniformité des notations, dans leur définition et leur choix

  Démo : la bibliothèque de listes

Ingrédients

Éléments de bilan d'une expérience de développement "autarcique":

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

  Démo : une version révisée d'Arith
- Discipline et révisions successives doivent assurer:
  - Cohérence dans les choix de noms
  - Systématisme dans la présence des lemmes
  - Uniformité des notations, dans leur définition et leur choix

  Démo : la bibliothèque de listes
- Complémentarités des mécanismes de coercions et d'inférence de types:

Ingrédients

Éléments de bilan d'une expérience de développement "autarcique":

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

  Démo : une version révisée d'Arith
- Discipline et révisions successives doivent assurer:
  - Cohérence dans les choix de noms
  - Systématisme dans la présence des lemmes
  - Uniformité des notations, dans leur définition et leur choix

  Démo : la bibliothèque de listes
- Complémentarités des mécanismes de coercions et d'inférence de types:
  - Définition de la hiérarchie algébriques en termes de "mixins"

Ingrédients

Éléments de bilan d'une expérience de développement "autarcique":

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

  Démo : une version révisée d'Arith
- Discipline et révisions successives doivent assurer:
  - Cohérence dans les choix de noms
  - Systématisme dans la présence des lemmes
  - Uniformité des notations, dans leur définition et leur choix

  Démo : la bibliothèque de listes
- Complémentarités des mécanismes de coercions et d'inférence de types:
  - Définition de la hiérarchie algébriques en termes de "mixins"
  - Notations génériques, et notations itérées

Ingrédients

Éléments de bilan d'une expérience de développement “autarcique”:

- Choix de styles de formalisation:
  - Propriété décidable $\Rightarrow$ prédicat booléen: exploiter le contenu calculatoire des objets pour les preuves,
  - mais avec du contrôle,
  - peu de définitions inductives, mais des spec inductives

  Démo : une version révisée d'Arith
- Discipline et révisions successives doivent assurer:
  - Cohérence dans les choix de noms
  - Systématisme dans la présence des lemmes
  - Uniformité des notations, dans leur définition et leur choix

  Démo : la bibliothèque de listes
- Complémentarités des mécanismes de coercions et d'inférence de types:
  - Définition de la hiérarchie algébriques en termes de “mixins”
  - Notations génériques, et notations itérées
  - Inférence de preuves par inférence de types

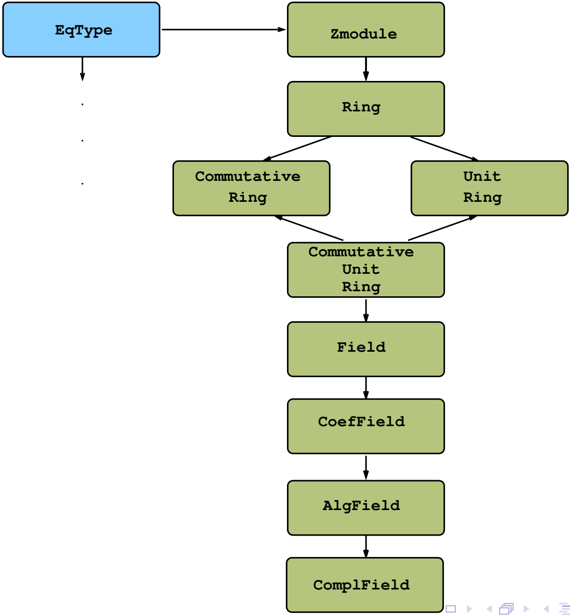
Une utilisation massive de l'inférence de type

Utiliser l'inférence de type du système comme moteur d'inférence de preuves.
Actuellement deux mécanismes complémentaires pour supporter ces techniques:

- Les classes de types, introduites par M. Sozeau (2008 cf. manuel v8.2), basées sur des obligations de preuves traitées par `eauto`.
- Les structures canoniques, introduites par A. Saïbi ($\sim$ 1996? cf. crédits v6.1), basées sur une base de données pour l'unification.

Ici on utilise exclusivement des structures canoniques, et on en utilise beaucoup.

Hiérarchie algébrique (en construction)



Des téléscopes aux mixins

Le problème : formaliser l'héritage des structures

- Dans les modèles *à la* C-CORN, on construit des téléscopes:
  - Chaque structure est un record
  - Une projection de ce record est la structure inférieure
  - Cette projection est une coercion
- Problèmes:
  - Héritage multiple
  - Empilement de projections qui fait ployer la comparaison de terme
- Alternative:
  - Découper les spécifications en tranches (mixins)
  - Utiliser conjointement coercions et inférence

Opérateurs itérés

Le problème : comment gérer des opérateurs itérés, comme : $\sum, \prod$ ou $\bigcup, \bigcap$...

On veut:

- Des notations pratiques et uniformes: $\sum_{i=0}^{n}, \bigcup_{A \notin E}, \bigcap_{A | P(A)} \dots$
- Des outilsgénériques pour l'associativité: $\bigcap_{A \in E} = \bigcap_{A \in E \cap B} \cap \bigcap_{A \in E \cap B^c} \dots$
- Des outils génértique pour les paires d'opérateurs ayant des propriétés relatives (distributivité, ...)

$$\prod_{i=1}^{n} \sum_{j=1}^{m} = \sum \prod$$

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I → R) : R :=
  foldr (fun i x ⇒ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  Structure law : Type := Law {
  operator : T → T → T;
  _ : associative operator;
  _ : left_unit unit operator;
  _ : right_unit unit operator
  }

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)
- Des notations pour distiguer les différentes formes d'itération, en fonction aussi des propriétés de l'opérateur

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)
- Des notations pour distiguer les différentes formes d'itération, en fonction aussi des propriétés de l'opérateur

```
Notation "\big [ op / nil ]_( m ≤ i < n ) F" :=
(bigop
nil op (index_iota m n)(fun _ ⇒ true)(fun i:nat ⇒ F))
```

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I → R) : R :=
  foldr (fun i x ⇒ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)
- Des notations pour distiguer les différentes formes d'itération, en fonction aussi des propriétés de l'opérateur

```
Notation "\big [ op / nil ]_( m ≤ i < n ) F" :=
(bigop
nil op (index_iota m n)(fun _ ⇒ true)(fun i:nat ⇒ F))

Notation "\sum_(m ≤i< n) F" :=
(\big[+%R / 0%R]_(m ≤ i < n) F%R)
```

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)
- Des notations pour distiguer les différentes formes d'itération, en fonction aussi des propriétés de l'opérateur
- Une théorie incrémentale des opérateurs itérés:

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)
- Des notations pour distiguer les différentes formes d'itération, en fonction aussi des propriétés de l'opérateur
- Une théorie incrémentale des opérateurs itérés:
  - D'abord des lemmes d'extentionnalité

Opérateurs itérés

- Une définition générique d'itération:
  Definition bigop R I nil op r (P : pred I) (F : I $\rightarrow$ R) : R :=
  foldr (fun i x $\Rightarrow$ if P i then op (F i) x else x) nil r.
- Des structures algébriques pour les propriétés d'opérateurs:
  - Des lois abéliennes, multiplicatives,...
  - Des structures sur les propriétés relatives des opérateurs (anneau,...)
- Des notations pour distiguer les différentes formes d'itération, en fonction aussi des propriétés de l'opérateur
- Une théorie incrémentale des opérateurs itérés:
  - D'abord des lemmes d'extentionnalité
  - Plus de propriétés sur l'opérateur itéré donne plus de propriétés à l'itération:
    Lemma sum_predU : forall (I : finType) (N : I $\rightarrow$ nat) (a b : pred I),
    [disjoint a b] $\rightarrow$
    \sum_(i | (a i || b i)) N i = (\sum_(i | a i) N i) + (\sum_(i | b i) N i).

Opérateurs itérés

Et maintenant, après les déclarations de structures canoniques appropriées:

- Déterminants :
  Definition determinant n (A : M_n) :=
  \sum_(s : S_n) $(-1)^{+s} *$ \prod_i A i (s i).
- Sous groupes engendrés:
  Definition generated A := \bigcap_(G : group | A \subset G) G.
- Instanciantion de propriétés génériques:
  Lemma subset_gen : forall A : group elt, A \subset $<< A >>$.
  Proof. intros A; exact: bigcapsP. Qed.

Inférence de preuves

En plus de la surcharge ou de l'inférence de structure, les mécanismes de structures canoniques permettent de l'inférence de preuves.

Inférence de preuves

En plus de la surcharge ou de l'inférence de structure, les mécanismes de structures canoniques permettent de l'inférence de preuves.

Démo : des vecteurs dépendants (presque) sans douleur

Conclusion

- Des objets non calculatoires
- Des bibliothèques constructives
- Une part significative d'infrastructure: notations, assemblage de structures, exhausitivités des lemmes,...
- Des techniques de développement utilisant les outils modernes proposés par le système