Types quotients en *Coq*

Cyril Cohen

INRIA Saclay - Île de France
École Polytechnique

1er février 2010

Illustration

Exemple

- On a un ensemble des gens, représentés par leur numéro de sécurité sociale.

Illustration

Exemple

- On a un ensemble des gens, représentés par leur numéro de sécurité sociale.
- Relation d'équivalence : "être née la même année que".

Illustration

Exemple

- On a un ensemble des gens, représentés par leur numéro de sécurité sociale.
- Relation d'équivalence : "être née la même année que".
- Une classe d'équivalence
  $=$ ensemble de gens nées là même année
  $\hat{=}$ une génération.

Illustration

Exemple

- On a un ensemble des gens, représentés par leur numéro de sécurité sociale.
- Relation d'équivalence : "être née la même année que".
- Une classe d'équivalence
  $=$ ensemble de gens nées là même année
  $\hat{=}$ une génération.
- Quotient : l'ensemble des classes d'équivalence

Illustration

Exemple

- On a un ensemble des gens, représentés par leur numéro de sécurité sociale.
- Relation d'équivalence : "être née la même année que".
- Une classe d'équivalence
  = ensemble de gens nées là même année
  $\hat{=}$ une génération.
- Quotient : l'ensemble des classes d'équivalence
  = l'ensemble des générations

Enjeu

- On utilise parfois les quotients de manière implicite.

Enjeu

- On utilise parfois les quotients de manière implicite.
- Les quotients jouent un rôle central dans les mathématiques,

Enjeu

- On utilise parfois les quotients de manière implicite.
- Les quotients jouent un rôle central dans les mathématiques,
- Comprendre comment formaliser la notion en théorie des types.

Les quotients

→ Facile en théorie des ensembles (théorie extensionnelle, qui parle de relations)

Les quotients

→ Facile en théorie des ensembles (théorie extensionnelle, qui parle de relations)
→ Difficile en théorie des types (théorie intensionnelle, qui parle de fonctions, de calculs)

Les types quotients : illustration

```
Definition Personne :=
   (nat*(nat*(nat*(nat*(nat*nat))))).
Definition meme_anniv (p q : Personne) :=
   fst(snd p) = fst(snd q).

Lemma ma_equiv: equivalence meme_anniv.
Proof. (* ... *) Qed.

Definition Generation := quotient ma_equiv.
```

Les types quotients

- L'utilisation de `quotient` est "gratuite"

Les types quotients

- L'utilisation de `quotient` est "gratuite"
- `quotient` est un axiome
  $\Rightarrow$ on perd du contenu calculatoire

Les *Sétoïdes* : illustration

```
Lemma ma_refl: reflexive meme_anniv. (* ... *)
Lemma ma_sym: symmetruc meme_anniv. (* ... *)
Lemma ma_trans : transitive meme_anniv. (* ... *)

Add Parametric Relation : Personne meme_anniv
  reflexivity proved by ma_refl
  symmetry proved by ma_sym
  transitivity proved by ma_trans
as Generation.
```

Les *Sétoïdes*

On encode

Les *Sétoïdes*

On encode

- Utilisation calquée sur la théorie des ensemble : Un type muni d'une égalité *Sétoïde*.

Les *Sétoïdes*

On encode

- Utilisation calquée sur la théorie des ensemble : Un type muni d'une égalité *Sétoïde*.
- Beaucoup de bureaucratie, pour gérer l'encodage
  $\Rightarrow$ réalisé par de la méta-programmation

Les *Sétoïdes*

On encode

- Utilisation calquée sur la théorie des ensemble : Un type muni d'une égalité *Sétoïde*.
- Beaucoup de bureaucratie, pour gérer l'encodage
  $\Rightarrow$ réalisé par de la méta-programmation

Cette solution est implémentée dans la distribution standard de *Coq*.

Types Quotients

on perd le calcul

Sétoïde

plus de bureaucratie
$\Rightarrow$ gérée par de la méta-programmation

Types Quotients

on perd le calcul

Sétoïde

plus de bureaucratie
$\Rightarrow$ gérée par de la
méta-programmation

Les types normalisés

On modifie la théorie

Les types normalisés

On modifie la théorie

- On sort du *calcul des constructions inductives* : il faut étendre le calcul (équivalent à la *Proof-Irrelevance*)

Les types normalisés

On modifie la théorie

- On sort du *calcul des constructions inductives* : il faut étendre le calcul (équivalent à la *Proof-Irrelevance*)
- On se restreint aux quotients calculatoires

Les types normalisés

On modifie la théorie

- On sort du *calcul des constructions inductives* : il faut étendre le calcul (équivalent à la *Proof-Irrelevance*)
- On se restreint aux quotients calculatoires

L'implémentation est coûteuse et n'a pas encore été faite.

Types Quotients

on perd le calcul

Types Normalisés

modification de la théorie

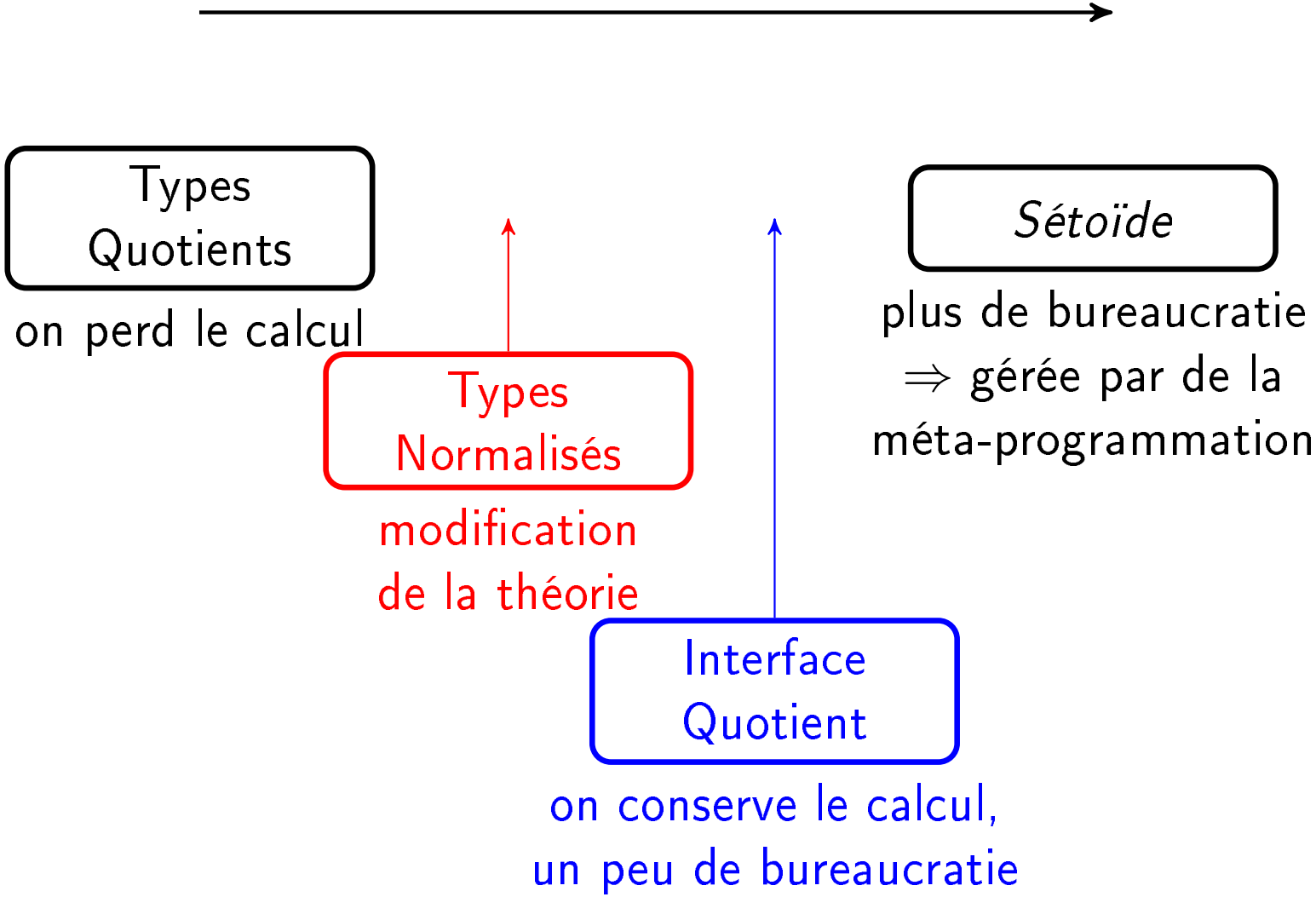
Sétoïde

plus de bureaucratie

$\Rightarrow$ gérée par de la méta-programmation

Solution proposée : *Interface Quotient*

On reprend les idées sur les types normalisés. Mais dans un cadre plus spécifique, où l'égalité est décidable.


Types
Quotients
on perd le calcul
Types
Normalisés
modification
de la théorie
Interface
Quotient
on conserve le calcul,
un peu de bureaucratie
Sétoïde
plus de bureaucratie
⇒ gérée par de la
méta-programmation

Outline

1. Définition explicite du quotient

Outline

1. Définition explicite du quotient
2. Obtention de lemmes d'elimination

Outline

1. Définition explicite du quotient
2. Obtention de lemmes d'elimination
3. Obtention de propriétés de morphisme

Principe d'utilisation

Constuction d'un quotient

Pour l'utiliser, l'utilisateur doit construire :

- un type `Q` "à la main"
- deux fonctions `repr : Q → T` et `pi : T → Q`
- une preuve `pi_repr` de $\forall$ `x : T, pi (repr x)= x`

Illustration

```
Definition Generation := nat.

Definition repr (g : Generation) : Personne :=
     (1,(g,(1,(1,(1,1))))).

Definition pi (p : Personne) : Generation :=
     fst (snd p).

Lemma pi_repr : ∀g, pi (repr g) = f.
Proof. (* ... *) Qed.
```

Déclaration

Interface quotient

```
QuotType : T → ∀ (Q : Type),
   ∀ (repr : Q → T) (pi : T → Q),
   (∀ x : Q, pi (repr x) = x) → quotType T
```

On a une coercion `quot_sort : quotType T ↦ Q`

Illustration

```
Definition Generation_qT := QuotType pi_repr.
```

Principe d'utilisation

Schéma d'élimination

Si `qT : quotType T`

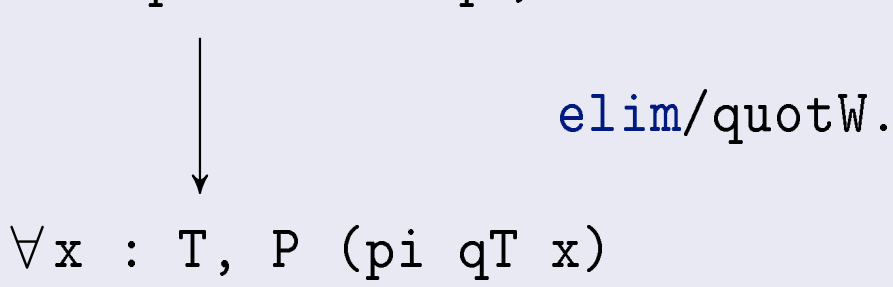


Illustration de l'élimination

```
∀g : Generation, P g
```

elim/quotW.

```
∀x : Personne, P (pi Generation_qT x)
```

Fonctions stables

Fonctions stables $\leftrightarrow$ Morphismes *Sétoïdes*

Fonctions stables : illustration

- Si on a `scol : Personne → Scol`
- et si on a

```
Lemma scol_compat : ∀ (p q : Personne),
  p ≡ q mod Generation → scol p = scol q.
```

- Alors on obtient `scol_Q : Generation → Scol`

Fonction stable

Définition d'une fonction stable

Il faut

- une fonction `f : T → S`
- une preuve `f_compat`
- On obtient le relèvement `f_Q : Q → S` de `f` de la manière suivante.

```
Definition f_Q := qT_op1 f_compat.
```

Réécriture

Réécriture

```
f_Q (pi x) = ...
```

```
                rewrite qTE.
pi (f x) = ...
```

Réécriture : Illustration

```
scol_Q (pi p) = ...
```

```
rewrite qTE.
```

```
scol p = ...
```

Applications

J'ai pu appliquer cette construction sur trois exemples

Applications

J'ai pu appliquer cette construction sur trois exemples

- L'exemple "jouet" des entiers relatifs

Applications

J'ai pu appliquer cette construction sur trois exemples

- L'exemple "jouet" des entiers relatifs
- La construction du corps des fractions sur un anneau intègre

Applications

J'ai pu appliquer cette construction sur trois exemples

- L'exemple "jouet" des entiers relatifs
- La construction du corps des fractions sur un anneau intègre
- La construction de l'anneau des polynômes multivariés (avec un nombre dénombrable d'indéterminées).

Puissance

- On a un cadre interne à la théorie

Puissance

- On a un cadre interne à la théorie
- On fait de la “vraie” réécriture

Puissance

- On a un cadre interne à la théorie
- On fait de la “vraie” réécriture
- On dispose d’un lemme d’élimination des quotients vers leur type de base

Particularités

- On se limite à une catégorie un peu plus spécifique de quotients constructifs

Particularités

- On se limite à une catégorie un peu plus spécifique de quotients constructifs
- On est obligé de construire le type quotient à la main, puis de lui faire correspondre la spécification

Particularités

- On se limite à une catégorie un peu plus spécifique de quotients constructifs
- On est obligé de construire le type quotient à la main, puis de lui faire correspondre la spécification
- On va vers une technique d'encodage plutôt que de modification du calcul,

Particularités

- On se limite à une catégorie un peu plus spécifique de quotients constructifs
- On est obligé de construire le type quotient à la main, puis de lui faire correspondre la spécification
- On va vers une technique d'encodage plutôt que de modification du calcul,
  Donc il reste de la bureaucratie, gérée par les *Structures Canoniques*.

Conclusion

- On trouve un équilibre entre simplicité et calcul (on a toutes les conditions demandées par les types normalisés, tout en restant dans CCI)

Conclusion

- On trouve un équilibre entre simplicité et calcul (on a toutes les conditions demandées par les types normalisés, tout en restant dans CCI)
- On obtient une structure générique, réutilisable et qui s'intègre bien dans les développements du projet *Composants Mathématiques*

Merci de votre attention

Des questions ?