# Definable quotients in intensional type theory

#### Nuo Li

School of Computer Science University of Nottingham, UK

FP Away day, July 2014

### What is a quotient?

- In arithmatics, 4 is the quotient of  $8 \div 2$  (or 8/2)
- In set theory, given a set A and an equivalence relation  $\sim$  on it, a quotient set  $A/\sim$  is the collection of equivalence classes.
- ullet In type theory, a quotient type is a type A whose equality is redefined by an equivalence relation  $\sim$  on it.

# Quotients in type theory

- Quotient type is an extensional notion which is not available in intensional type theory.
- Setoids are sometimes used to represent quotients. A setoid consists
  of
  - $\bigcirc$  a set A,
  - $\bigcirc$  a relation  $\sim: A \to A \to \mathbf{Prop}$ , and
  - it is an equivalence, i.e. it is reflexive, symmetric and transitive.
- $\bullet$  Some quotients are definable inductively without using quotient type former, e.g. the set of rational numbers  $\mathbb Q$
- Algebraic structures for these quotients in intensional type theory:
  - Prequotient
  - Quotient (dependent eliminator)
  - Quotent (Hofmann's version)
  - Exact quotient
  - Definable quotient

### Prequotient

#### Prequotient

Given a setoid  $(A, \sim)$ , a prequotient  $(Q, [\_], sound)$  over that setoid consists of

- $\bigcirc$  a set Q,
- ② a function  $[\_]: A \rightarrow Q$ ,
- ullet a proof sound that the function  $[\_]$  is compatible with the relation  $\sim$ , that is

sound: 
$$(a, b : A) \rightarrow a \sim b \rightarrow [a] = [b]$$
,

In category theory, prequotient corresponds to a commute diagram of morphisms - (co) for k

$$R \xrightarrow{\pi_0} A \xrightarrow{[_]} Q$$

### Quotient (Hofmann's version)

### Quotient (Hofmann's)

A prequotient  $(Q, [\_], sound)$  is a quotient (Hofmann's) if we also have

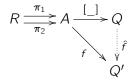
lift: 
$$(f: A \to B) \to (\forall a, b \to a \sim b \to f \ a \equiv f \ b) \to (Q \to B)$$

such that lift  $-\beta$ : lift f[a] = f[a],

together with an induction principle. Suppose B is a predicate,

qind : 
$$((a : A) \rightarrow B [a] \rightarrow ((q : Q) \rightarrow B q)$$

In category theory, a quotient is just a coequalizer (a special fork)



### Quotient (dependent eliminator)

#### Quotient(dependent eliminator)

A prequotient  $(Q, [\_], sound)$  is a quotient if we also have

 $\bigcirc$  for any  $B:Q\to\mathbf{Set}$ , an eliminator

$$\begin{aligned} \operatorname{qelim}_{B} &: (f : (a : A) \to B [a]) \\ &\to ((p : a \sim b) \to f \ a \simeq_{\mathsf{sound} \ p} f \ b) \\ &\to ((q : Q) \to B \ q) \end{aligned}$$

such that  $qelim - \beta$ :  $qelim_B f p[a] \equiv f a$ .

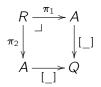
They are equivalent.

### Exact quotient

A quotient is exact (or effective) if

$$exact: (\forall a, b: A) \rightarrow [a] \equiv [b] \rightarrow a \sim b$$

i.e. a coequalizer is exact if following diagram commutes (i.e. a pullback)



ullet Propositional extensionality (univalence for propositions)  $\Longrightarrow$  every quotient is exact.

### Definable quotient

#### Definable quotient

Given a setoid  $(A, \sim)$ , a definable quotient is a prequotient  $(Q, [\_], sound)$  with

emb : 
$$Q \to A$$
  
complete :  $(a : A) \to \text{emb} [a] \sim a$ 

stable : 
$$(q:Q) \rightarrow [\text{emb } q] \equiv q$$
.

It corresponds to a split coequalizer:

$$R \xrightarrow{\pi_0} A \xrightarrow{[_]} Q$$

a fork with morphisms emb :  $Q \rightarrow A$  and  $t : A \rightarrow R$  such that

[\_] 
$$\circ$$
  $emb = 1_C$ ,  $emb \circ$ [\_] =  $\pi_2 \circ t$  and  $\pi_1 \circ t = 1_B$ . ( $t = (emb[a], a)$ )

### Relations of these structures

- Definable quotients give rise to exact quotients.
  - Quotient (dependent version) ← Quotient (Hofmann's version)
  - $\bigcirc$  Definable quotients  $\rightarrow$  quotient
  - Definable quotient is exact Proof:

$$[a] = [b]$$

 $\implies$  emb [a] = emb [b] (by congruence)

 $\implies a \sim \text{emb } [a] = emb [b] \sim b \text{ (by completeness)}$ 

 $\implies a \sim b$ 

# Example : Integers $\mathbb{Z}$ I

setoid integer :

$$\mathbb{Z}_0 := \mathbb{N} \times \mathbb{N}$$

$$(n_1, n_2) \sim (n_3, n_4) := n_1 + n_4 = n_3 + n_2$$

An example of the operation defined on setoid integers: addition

\_+\_ : 
$$\mathbb{Z}_0 \to \mathbb{Z}_0 \to \mathbb{Z}_0$$
 (x+ , x-) + (y+ , y-) = (x+ N+ y+) , (x- N+ y-)

# Example : Integers $\mathbb{Z}\ II$

set integer:

```
data \mathbb{Z}: Set where +__ : \mathbb{N} \to \mathbb{Z} -suc__ : \mathbb{N} \to \mathbb{Z}
```

Normalisation function:

# Example : Integers $\mathbb{Z}$ III

 To find a canonical choice for each equivalence class, embedding function:

sound complete stable and exact can be verified.

# Example : Rational numbers $\mathbb Q$ I

Setoid encoding : unreduced fractions

$$\mathbb{Q}_0=\mathbb{Z}\times\mathbb{N}$$

Note: 1/2 represents  $\frac{1}{3}$  Equivalence relation :

$$\frac{a}{b} = \frac{c}{d} := a \times d = c \times b$$

Set encoding: reduced fractions

$$\mathbb{Q} = \Sigma((n, d) : \mathbb{Z} \times \mathbb{N})).\mathsf{coprime}\ n\ (d+1)$$

# Example : Rational numbers $\mathbb{Q}\ II$

Normalisation is essentially just fraction reducing

```
 \begin{split} & [\_]: \mathbb{Q}_0 \to \mathbb{Q} \\ & [ \ (+\ 0)\ / \text{suc d}\ ] = \mathbb{Z}. + \_\ 0 \div 1 \\ & [ \ (+\ (\text{suc n}))\ / \text{suc d}\ ] \ \text{with gcd (suc n) (suc d)} \\ & [ \ (+\ \text{suc n})\ / \text{suc d}\ ] \ | \ \text{di}\ , \ g = GCD' \to \mathbb{Q} \ (\text{suc n) (suc d)} \\ & \ \text{di}\ (\lambda\ ())\ (C.gcd-gcd'\ g) \\ & [ \ (-\text{suc n})\ / \text{suc d}\ ] \ \text{with gcd (suc n) (suc d)} \\ & \dots \ | \ \text{di}\ , \ g = -\ GCD' \to \mathbb{Q} \ (\text{suc n) (suc d)} \ \text{di}\ (\lambda\ ()) \\ & \ (C.gcd-gcd'\ g) \end{split}
```

Embedding is trivial – forgetting proof of coprimarity

# The application of definable quotients I

 As long as we have defined operations or have proved theorems for the setoid representations, we can lift them by just mixing normalisation and embedding functions. e.g.
 In the case of Integers:

$$\begin{array}{l} \mathsf{lift}_1: \ (\mathsf{op}: \, \mathbb{Z}_0 \to \mathbb{Z}_0) \to \mathbb{Z} \to \mathbb{Z} \\ \mathsf{lift}_1 \ \mathsf{op} = [\,\,] \circ \mathsf{op} \circ \ulcorner\,\,\,\,] \end{array}$$

Advantages of definable quotients:

- Setoid has simpler encoding, better manipulation
- More auxiliary functions
- Flexible in conversions of two representations
- Benefit from both

### Case: The distributivity of integers I

For set encoding:

```
\operatorname{dist}^{I} : \mathbb{Z}^{*} DistributesOver^{I} \mathbb{Z}^{+}
dist^{I} \times y \times z = sign y \times s = sign z
dist' \times y z \mid yes p
   rewrite p
       lem1 y z p
       | lem2 yzp =
   trans (cong (\lambda n \rightarrow sign x S* sign z \triangleleft n)
   (Ndist^{I} | x | | y | (| z |)))
   (lem3 (|x|N^*|y|) (|x|N^*|z|)
dist' \times y z \mid no \neg p = ...
```

Three lemmas for first case, second case has to be proved from scratch.

# Case: The distributivity of integers II

For setoid encoding:It can be done automatically

```
dist': _*_ DistributesOver' _+_
dist' (a , b) (c , d) (e , f) = solve 6
 (\lambda a b c d e f \rightarrow a :* (c :+ e) :+ b :* (d :+ f) :+
 (a :* d :+ b :* c :+ (a :* f :+ b :* e))
 :=
 a :* c :+ b :* d :+ (a :* e :+ b :* f) :+
 (a :* (d :+ f) :+ b :* (c :+ e))) refl a b c d e f
```

- Combined with "Reflection" technique, it can be done completely automatic, like the **ring** tactic in Coq.
- It can be generalised to any properties which can be converted into equations of natural numbers.

# Case: The distributivity of integers III

 In practice, we manually construct the proof for efficiency, which is still simpler.

```
\operatorname{dist}^{I}: \mathbb{Z}_{0}^{*} \operatorname{DistributesOver}^{I} \mathbb{Z}_{0}^{+}
dist' (a . b) (c . d) (e . f) =
    cong_2 N+ (dist-lem' a b c d e f)
    (sym (dist-lem<sup>1</sup> a b d c f e))
dist-lem': \forall a b c d e f \rightarrow
    a \mathbb{N}^* (c \mathbb{N}+ e) \mathbb{N}+ b \mathbb{N}^* (d \mathbb{N}+ f) \equiv
    (a \mathbb{N}^* c \mathbb{N}+ b \mathbb{N}^* d) \mathbb{N}+ (a \mathbb{N}^* e \mathbb{N}+ b \mathbb{N}^* f)
dist-lem' a b c d e f = trans
    (cong_2 \mathbb{N} + (\mathbb{N}dist' \ a \ c \ e) (\mathbb{N}dist' \ b \ d \ f))
    (swap23 (a N* c) (a N* e) (b N* d) (b N* f))
```

### Summary

- Quotients types are not available in intensional type theory.
- Setoids are not good enough.
- Algebraic structures to encode quotient sets which are definable.
- The advantages of using definable quotient structures.
- Any Questions?