

Using Postulated Colimits in Coq

Work in Progress

March 5, 2018

Abstract

In this talk we define and construct finite colimits in the Coq proof assistant in a context that is similar to the category of sets. First we review without proof the key mathematical ideas involved in the theory of postulated colimits as described in a note of Anders Kock. This theory gives us a way to prove results about colimits in an arbitrary sheaf topos. Then we give an inductive definition in Coq of the fundamental notion of zigzag in this theory. We finish by proving the result analogous to the (mathematically easy) result that in the category of sets pushouts of monomorphisms are monomorphisms.

Contents

1	General References	1
2	Postulated Coequalisers	1
3	Objoids and Mapoids	2
4	An Inductive Definition of Zigzag	3
5	Disjoint Unions	4
6	Pushouts from Coequalisers and Coproducts	4
7	Pushouts of Monomorphisms in a Topos	5

1 General References

We make general reference to a paper [?] of Anders Kock and a master's thesis of [?].

2 Postulated Coequalisers

We will use the traditional definition to define a coequaliser.

Definition 2.1. The *coequaliser* q of a parallel pair $a, b : R \rightrightarrows X$ is an arrow

$$\begin{array}{ccccc} R & \xrightarrow[s]{t} & X & \xrightarrow{q} & Q \\ & & \searrow \scriptstyle \forall z & & \downarrow \scriptstyle \exists! \psi \\ & & & & Z \end{array}$$

such that

- the equality $sq = tq$ holds
- for all $z : X \rightarrow Z$ such that $sz = tz$ there exists a unique $\psi : Q \rightarrow Z$ such that $q\psi = z$.

We will use the idea of a postulated coequaliser to give an explicit presentation of coequalisers in the category of sets. We refer to the preprint [?].

Definition 2.2. An *s-t-zigzag* from $x \in X$ to $y \in X$ is a sequence $r_1, r_2, \dots, r_n \in R$ such that

$$\begin{array}{ccccccc} & & r_1 & & r_2 & & r_3 & & \dots & & r_n & & \\ & \swarrow \phi_1 & & \searrow \phi_2 & \swarrow \phi_3 & & \searrow \phi_4 & \swarrow \phi_5 & \searrow \phi_6 & & \swarrow \phi_7 & & \searrow \phi_8 \\ x & & & x_1 & & x_2 & & & & \dots & & y \end{array}$$

where the $\phi_i \in \{s, t\}$.

Definition 2.3. A cofork

$$R \xrightarrow[t]{s} X \xrightarrow{q} Q$$

is a *postulated coequaliser* iff

- the arrow q is an epimorphism
- for all $x, y \in X$ $(x)q = (y)q$ iff there exists an *s-t-zigzag* from x to y .

We can use postulated coequalisers to give a concrete presentation of coequalisers due to the following result of Kock.

Proposition 2.4. *In a sheaf topos every postulated coequaliser is a coequaliser.*

3 Objoids and Mapoids

In our formulation we replace a sets with objoids. An objoid is a type equipped with an equivalence relation on the terms of the type. This is similiar to [?]

Record objoid: Type :=

```
{carrier:>Type;
 eq:carrier->carrier->Prop;
 refl:reflexive carrier eq;
 sym:symmetric carrier eq;
 trans:transitive carrier eq}.
```

Infix "~" := eq (at level 95).

We replace functions with *mapoids* which are the appropriate structure preserving maps for objoids.

```
Structure mapoid (A B:objoid) :=
  {map:>carrier A -> carrier B;
   pres (a1 a2:carrier A) (H:a1~a2):map a1~map a2}.
```

Sometimes it is easier to use an infix for function application.

```
Definition application (a1:carrier A) (f1:mapoid A B) := map f1 a1.
Infix "|>" := application (at level 11, left associativity).
```

We use double pipe for composition.

```
Definition comp:=
  {|map:=fun a:carrier A=>g(f(a));
   pres:=comp_pres|}.
Infix "||>" := comp (at level 10, right associativity).
```

Finally we assert extensionality for functions.

```
Axiom mapoid_ext:
  (forall a:carrier A, f(a)~f2(a)) -> f=f2.
Axiom mapoid_app:
  f=f2 -> (forall a:carrier A, f(a)~f2(a)).
```

4 An Inductive Definition of Zigzag

It turns out that we can give an inductive definition of zigzag. Note that in order to formulate this in Coq we need to add a base case and a way to cons equivalent elements onto a zigzag.

```
Inductive zigzag(x1 x2:X):Prop:=
  |xid (H:x1~x2):zigzag x1 x2
  |xcons {x3:X}: (x1~x3)-> (zigzag x3 x2) -> zigzag x1 x2
  |stcons{r:R}{x3:X} (H1:x1~(s(r))) (H2:x3~(t(r)))
    (z1:zigzag x3 x2): zigzag x1 x2
  |tscons{r:R}{x3:X} (H1:x1~(t(r))) (H2:x3~(s(r)))
    (z1:zigzag x3 x2): zigzag x1 x2.
```

Lemma 4.1. *The zigzag relation is reflexive, symmetric and transitive.*

This means that we can construct a presentation of a coequaliser as a postulated coequaliser.

```
Definition Q:objoid:=
  {|carrier:=X;
   eq:=zigzag;
```

```

    refl:=zigzag_refl;
    sym:=zigzag_sym;
    trans:=zigzag_trans|}.
Definition q:mapoid X Q:=
  {|map:=fun x:X => x:Q;
   pres:=id_pres|}.

```

Proposition 4.2. *The postulated coequaliser is a coequaliser.*

Proof. The factorisation is given by the identity

```

Definition fact_arrow(Z:objoid) (z:mapoid X Z):Q->Z:=
  fun x:Q => z(x).
Definition factorisation
  (Z:objoid) (z:mapoid X Z) (H:s||>z = t||>z):mapoid Q Z:=
  {|map:=fact_arrow Z z;
   pres:=(fact_arrow_pres Z z H)|}.

```

and the main work is to prove that \sim is preserved. For this we refer to the proof in Coq. \square

5 Disjoint Unions

We have already defined and given a concrete presentation of coequalisers. In order to get all finite limits we only need to add finite coproducts. The natural translation of disjoint union into type theory is:

```

Inductive du:Type :=
  |b:B -> du
  |c:C -> du.
Inductive du_eq:relation du:=
  |beq(b1 b2:B) (H:b1~b2):du_eq (b b1) (b b2)
  |ceq(c1 c2:C) (H:c1~c2):du_eq (c c1) (c c2).

```

Proposition 5.1. *The disjoint unions defined above satisfy the universal property of coproducts.*

6 Pushouts from Coequalisers and Coproducts

In any category with finite colimits if the square

$$\begin{array}{ccc}
 A & \xrightarrow{g} & B \\
 \downarrow f & & \downarrow i_1 \\
 C & \xrightarrow{i_0} & Q
 \end{array}$$

is a pushout then we can compute Q using a disjoint union and a coequaliser

$$\begin{array}{ccccc}
 & & B & & \\
 & f \nearrow & \downarrow u_0 & & \\
 A & \xrightarrow{s} & B \amalg C & \xrightarrow{q} & Q \\
 & t \searrow & \uparrow u_1 & & \\
 & & C & &
 \end{array}$$

and where u_0 and u_1 are the coproduct inclusions $i_0 = u_0q$ and $i_1 = u_1q$.

Proposition 6.1. *The coequaliser of fu_0 and gu_1 is the pushout of f and g .*

7 Pushouts of Monomorphisms in a Topos

In a general topos we can make the following argument. For simplicity we assume that we are working in the category of sets and functions.

Lemma 7.1. *If f is a monomorphism and*

$$\begin{array}{ccc}
 A & \xrightarrow{g} & C \\
 \downarrow f & & \downarrow i_1 \\
 B & \xrightarrow{i_0} & Q
 \end{array}$$

is a pushout in Set then i_1 is a monomorphism.

Proof. We first note that $(f, g) : A \rightarrow B \times C$ is a monomorphism because f is a monomorphism. Next

$$\begin{array}{ccc}
 A & \xrightarrow{g} & C \\
 \downarrow f & & \downarrow i_1 \\
 B & \xrightarrow{i_0} & Q
 \end{array}
 \begin{array}{c}
 \searrow \text{ } \\
 \downarrow \psi \\
 \mathcal{P}C
 \end{array}
 \begin{array}{c}
 c \mapsto \{c\} \\
 \downarrow \phi_{(f,g)}
 \end{array}$$

where $\phi_{(f,g)}(b) = \{c \in C \mid \exists a \in A. f(a) = b \wedge g(a) = c\}$. Therefore i_1 is a monomorphism because $c \mapsto \{c\}$ is a monomorphism. \square

The proof that we will generalise is more straightforward. We sketch the argument below.

Lemma 7.2. *If f is a monomorphism and*

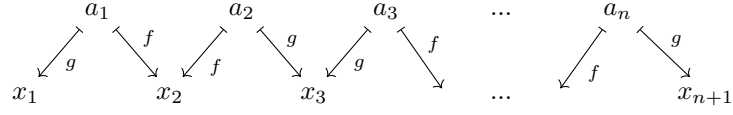
$$\begin{array}{ccc}
 A & \xrightarrow{g} & C \\
 \downarrow f & & \downarrow i_1 \\
 B & \xrightarrow{i_0} & Q
 \end{array}$$

is a pushout in Set then i_1 is a monomorphism.

Proof.

□

Suppose that $c1, c2 \in C$ such that $i_1(c1) \sim i_1(c2)$. This means that we can find a zigzag



where $x_1 \sim c1$, $x_{n+1} \sim c2$ and the order of the f and g s has been forced by the disjoint union. (As well as the fact that we are starting and ending at a g .) However since f is a monomorphism we see that $a_{2i-1} \sim a_{2i}$ and so

$$c1 \sim x_1 \sim x_3 \sim x_5 \sim \dots \sim x_{n+1} \sim c2$$

as required. **WARNING:** this proof hides a technical problem when doing the induction that Coq does not allow us to hide.