



Category Theory for Programmers

Homework (Session 5)

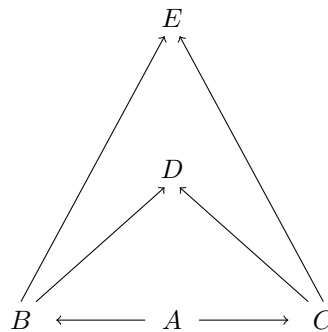
Bruno Vandekerkhove

Contents

Pushout in C++	1
Limit of the identity functor	1
Pullback, pushout, ... in Set	1
Co-equalizer	2
Pullback towards terminal object, pushout from initial object	2

Pushout in C++

We talked about products and co-products in posets before, where the term supremum and infimum applied. In this more general case applied to C++ types a diagram of a pushout D is as follows :



Any superclass E of B and C is a superclass of D . Which means that D corresponds to a superclass that encompasses all the functionality that the subclasses have in common¹.

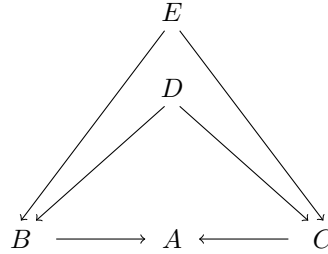
Limit of the identity functor

The identity functor Id maps a category to itself. The initial object is the object for which there's exactly one morphism to every object. When constructing the limit for Id any apex will correspond to an initial object (or there wouldn't be a natural transformation from Δ_c to Id). If there are many apexes (or initial objects) there will be invertible transformations between them. Indeed, initial objects are unique up to isomorphism, which we knew already.

¹One has to consider all possible superclasses of B and C . Then D inherits from all of those. Some superclasses may only declare a part of the functionality that's present in both the classes B and C , yet together they declare all of it and nothing more or some wouldn't be superclasses.

Pullback, pushout, ... in **Set**

Starting from the diagram depicted on the previous page and applying analogous reasoning we can see that a pushout D is the smallest superset or the union of B and C . In the opposite category we get :

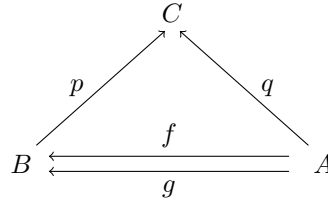


Here, the pullback D is the largest subset or the intersection (every other set that's a subset of B and C ought to be a subset of D). A lattice visualises this and the terms supremum and infimum are in order.

As for the initial - and terminal object, they (trivially) correspond to the empty set and the set on which the category is based.

Co-equalizer

Visually :



Once again q is fully defined by p and one of the morphisms from $A \rightarrow B$. We have

$$q = p \cdot f \quad \text{and} \quad q = p \cdot g \quad \Rightarrow \quad p \cdot f = p \cdot g$$

where for every other co-equalizer C' there should be a unique factoriser. Aside from this universal construction, there's a more concrete explanation of what it means in **Set**. There we end up with a partitioning of B that corresponds to an equivalence relationship $f(x) \sim g(x)$. It should be the smallest of such relationships which corresponds to the finest partitioning².

Pullback towards terminal object, pushout from initial object

A pullback is a special kind of product because A imposes some additional structure. When you're dealing with A as a terminal object and you're looking for a product, then that structure (morphisms to A from B , C and D) will necessarily be there. In other words, every such product is a pullback towards the terminal object (and vice versa, every such pullback is a product).

In the same vein, when you're dealing with A as an initial object and are looking for a coproduct then there will be unique morphisms from A to B , C and D which necessarily make it a pushout from the initial object.

²An example ; let $A = B = \{1, 2, 3\}$, $f = id$, $g(1) = 2$, $g(2) = 1$ and $g(3) = 3$. Then we find a co-equalizer p where $p(1) = p(2) = [1]$ and $p(3) = [3]$. This is a 'better' co-equalizer than p' with $p'(x) = [1]$ because we find the factoriser h with $h(x) = [1]$ for which $p' = h \cdot p$.