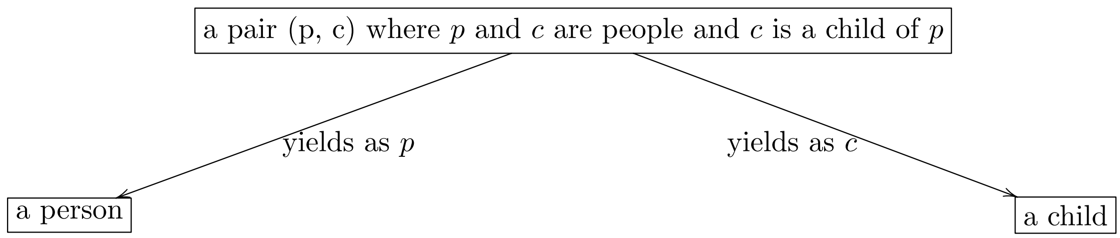
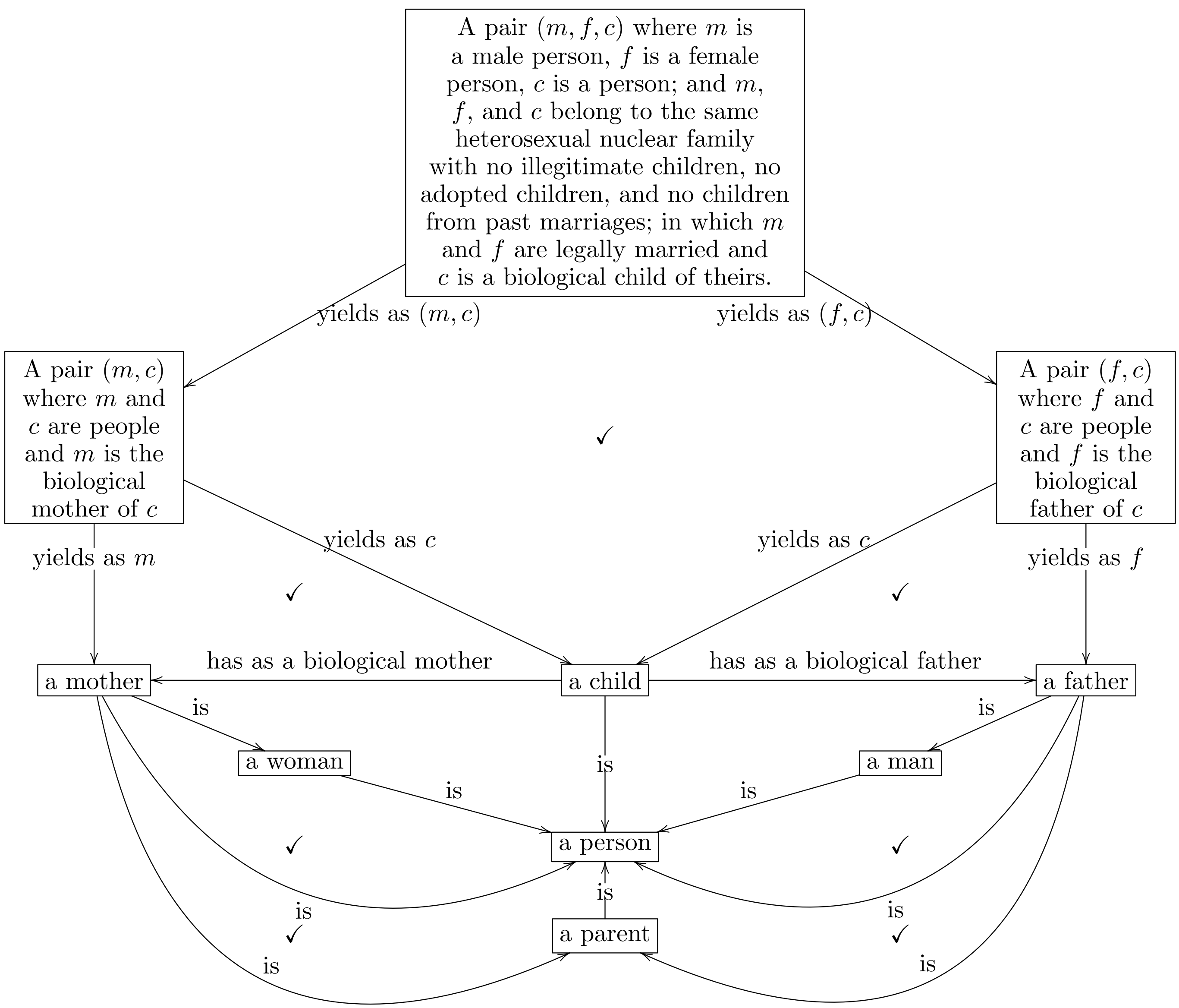
18.S996: Category Theory for Scientists (David Spivak), Problem Set 1

2.3.2.7



2.3.3.1



2.3.3.3

Omitted.

2.3.3.6

Given x, an operational landline phone, consider the following. We know that x is an operational landline phone, which is assigned a phone number, which has an area code, which corresponds to a region that we'll call $P(x)$.

We also know that x is an operational landline phone, which is a physical phone, which is currently located in a region that we'll call $Q(x)$.

Fact: whenever x is an operational landline phone, we will have $P(x) = Q(x)$.

2.3.3.7

No.

2.3.3.9

Every type $\ulcorner T \urcorner$ is the image of the aspect "is a T which is identical with some example of", that is, the aspect is the identity function on $\ulcorner T \urcorner$s. More interestingly,

- ⌜a book⌝ — "yields as b" (coming from, e.g., ⌜a pair (b, c) where b is a book and c is a sentence, and c describes the contents of b⌝)
- ⌜a material that has been fabricated by a process of type T⌝ — "yields as m" (coming from ⌜a pair (m, T) where m is a material and T is a type of process and m has been fabricated by a process of type T⌝)
- ⌜a bicycle owner⌝ — "is a bicycle owned by a single person, which has as an owner"
- ⌜a child⌝ — "yields as c" (coming from ⌜a pair (c, p) where c and p are people and p is a parent of c⌝)
- ⌜a used book⌝ — "yields as b" (coming from ⌜a pair (b, u) where b is a book and u is a person and u has used b⌝, or, more amusingly, coming from ⌜a pair (b, u) where b is a book and u is proof that b has been used⌝)
- ⌜an inhabited residence⌝ — "resides in" (coming from ⌜a person who resides somewhere⌝)

2.4.1.4

The set $A \times B$ has $|A| \cdot |B|$ elements. I have no idea why categorical limits are related to the operation $0 \cdot m :\equiv 0$; $Sn \cdot m :\equiv m + (n \cdot m)$. In this case, it works out to 12.

2.4.1.8

The lower left diagram and the upper center diagram commute.

2.4.1.13

$$\mathrm{Hom}_{\mathbf{Set}}(A, X) \times \mathrm{Hom}_{\mathbf{Set}}(A, Y) \cong \mathrm{Hom}_{\mathbf{Set}}(A, X \times Y)$$

2.4.1.14

The function is $s : x \mapsto (\pi_2 x) \times (\pi_1 x)$. More rigorously, here's some coq code which does this construction and proof

Library SwapExercise

```
Require Import Setoid JMeq.
Require Import Common.

Set Implicit Arguments.

Generalizable All Variables.

Definition compose X Y Z (f : Y -> Z) (g : X -> Y) := fun x => f (g x).

Arguments compose [X Y Z] f g x / .

Infix "o" := (@compose _ _ _) (at level 70).

Record is_isomorphism X Y (f : X -> Y) :=
  {
    isomorphism_inverse : Y -> X;
    isomorphism_right_inverse
    : forall x, (f o isomorphism_inverse) x = (fun x => x) x;
    isomorphism_left_inverse
    : forall x, (isomorphism_inverse o f) x = (fun x => x) x
  }.

Record isomorphic X Y :=
  {
    isomorphic_morphism :> X -> Y;
    isomorphic_is_isomorphism : is_isomorphism isomorphic_morphism
  }.

Infix "≅" := (isomorphic) (at level 70).

Section univ_prod.
  Variables X Y : Type.
```

Define products by the universal property

```
  Record product_type :=
    {
      prodXY :> Type;
      prod_fst : prodXY -> X;
      prod_snd : prodXY -> Y;
      prod_map : forall A (f : A -> X) (g : A -> Y), A -> prodXY;
      prod_fst_commutes_prop := (fun A f g prod_map =>
                                   forall x,
                                     (prod_fst o prod_map) x = f x);
      prod_snd_commutes_prop := (fun A f g prod_map =>
                                   forall x,
                                     (prod_snd o prod_map) x = g x);
      prod_fst_commutes : forall A f g,
                            prod_fst_commutes_prop A f g (@prod_map A f g);
      prod_snd_commutes : forall A f g,
```

```
                          prod_snd_commutes_prop A f g (@prod_map A f g);
      prod_map_unique : forall A f g prod_map',
                         prod_fst_commutes_prop A f g prod_map'
                         -> prod_snd_commutes_prop A f g prod_map'
                         -> forall x, prod_map' x = (prod_map f g) x
    }.

  Existing Class product_type.

  Definition product := { p : product_type & p }.
  Definition product_element (p : product) := projT2 p.
  Definition product_of `(p : product_type) (x : p) : product := existT _ p x.
  Global Coercion product_element : product >-> prodXY.
  Global Coercion product_of : prodXY >-> product.
End univ_prod.

Delimit Scope product_scope with product.
Bind Scope product_scope with prodXY.

Arguments prod_fst_commutes_prop _ _ _ _ _ _ _ / .
Arguments prod_snd_commutes_prop _ _ _ _ _ _ _ / .

Infix "×" := product : type_scope.
Infix "×" := (fun a b =>
              @prod_map _ _ _ unit (fun _ => a) (fun _ => b) tt) : product_scope.
Infix "×" := prod : old_type_scope.

Notation "π₁" := (@prod_fst _ _ _).
Notation "π₂" := (@prod_snd _ _ _).

Delimit Scope category_scope with category.
Delimit Scope old_scope with old_type.

Section swap.
```

Define swap via the universal property; note that the object type doesn't change, but the projection maps are switched. This is because the objects are truely opaque.

```
  Definition swap_types X Y : product_type X Y -> product_type Y X :=
    fun xy =>
      {|
        prodXY := prodXY xy;
        prod_fst := prod_snd xy;
        prod_snd := prod_fst xy;
        prod_map := (fun _ f g => prod_map xy g f);
        prod_fst_commutes := (fun _ f g => prod_snd_commutes xy g f);
        prod_snd_commutes := (fun _ f g => prod_fst_commutes xy g f);
        prod_map_unique := (fun _ _ _ _ Hfst Hsnd => prod_map_unique Hsnd Hfst)
      |}.
```

We must have `swap_types` in the environment for type class resolution to pick it up. Use `Eval simpl` so that it doesn't stick around.

```
  Definition swap X Y : X × Y -> Y × X
    := Eval simpl in
```

```
fun xy => let s := swap_types (projT1 xy) in
          (π₂ xy × π₁ xy)%product.
```

Prove that `swap_types` is an isomorphism; its inverse is `swap_types`

```
Definition swap_types_iso X Y : is_isomorphism (@swap_types X Y).
Proof.
  exists (@swap_types _ _);
  abstract (
      repeat intro;
      destruct_head_hnf @product_type;
      reflexivity
    ).
Defined.
```

Prove that `swap` is an isomorphism; its inverse is `swap`

```
Definition swap_iso X Y : is_isomorphism (@swap X Y).
Proof.
  exists (@swap _ _);
  abstract (
      repeat intro; simpl; unfold swap, swap_types; simpl;
      destruct_head_hnf @sigT;
      destruct_head_hnf @product_type;
      simpl in *;
      simpl_eq; simpl;
      trivial;
      symmetry;
      repeat match goal with
               | [ |- JMeq ?a ?b ] => (cut (a = b);
                                       [ let H := fresh in
                                         intro H;
                                           solve [ rewrite <- H; reflexivity
                                                 | rewrite H; reflexivity ]
                                       | ])
               | [ H : _ |- _ ] => rewrite H
               | [ x : _, H : _ |- _ ] => (eapply (H _ _ _ (fun _ : unit => x));
                                             hnf;
                                             trivial)
             end
    ).
Defined.
End swap.
```

Index

This page has been generated by coqdoc

2.4.1.15

$s : x \mapsto f(\pi_1(x)) \times f(\pi_2(x))$. Slightly more rigorously, here's some coq code which does this construction

Library FunctionProduct

```
Require Import Setoid.
Require Export SwapExercise.
Require Import Common.

Set Implicit Arguments.

Generalizable All Variables.

Local Open Scope type_scope.
Local Open Scope product_scope.

Section function_prod.
```

If A × B and A' × B' exist, then we can build a function from A × B to A' × B'

```
  Definition function_prod A A' B B' `(product_type A' B') (f : A -> A') (g : B ->
B') : A × B -> A' × B'
    := fun x => f (π1 x) × g (π2 x).
End function_prod.
```

Index

This page has been generated by coqdoc

2.4.2.4

If there are no non-landline phones which are not cell-phones, then, yes, ⌜a phone⌝ is the coproduct of ⌜a cellphone⌝ and ⌜a landline phone⌝. But maybe walkie-talkies are “phones”, or maybe there are non-landline phones on submarines or other boats which are not considered cellphones.

2.4.2.10

$$\mathrm{Hom}_{\mathbf{Set}}(X, A) \sqcup \mathrm{Hom}_{\mathbf{Set}}(Y, A) \cong \mathrm{Hom}_{\mathbf{Set}}(X \sqcup Y, A)$$

2.4.2.13

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{a phenomenon}\\ \text{which can be}\\ \text{viewed as a particle}\end{array}} \arrow[r, "\text{labeled “P” is}"] & \boxed{\begin{array}{c}\text{a phenomenon}\\ \text{which can be}\\ \text{viewed as a particle}\\ \text{(labeled “P”) or a}\\ \text{phenomenon which}\\ \text{can be viewed as a}\\ \text{wave (labeled “W”)}\end{array}} & \boxed{\begin{array}{c}\text{a phenomenon}\\ \text{which can be}\\ \text{viewed as a wave}\end{array}} \arrow[l, "\text{labeled “W” is}"']
\end{tikzcd}$$

Photons, can either be viewed as a wave or as a particle, and must be labeled with our viewpoint when mapped to the disjoint union in the center.

2.4.2.14

Following “Ologging products” very closely

Given two types c, d in an olog, there is a canonical label “$c \sqcup d$” for their product $c \sqcup d$, written in terms of the labels “c” and “d”. Namely,

$$\text{"}c \sqcup d\text{” := a “}c\text{” (labeled } x\text{) or a “}d\text{” (labeled } y\text{).}$$

The inclusions $c \to c \sqcup d \leftarrow d$ can be labeled “is, when labeled x,” and “is, when labeled y,” respectively.

Suppose that e is another object and $p : c \to e$ and $q : d \to e$ are two arrows. By the universal property of coproducts, p and q induce a unique arrow $c \sqcup d \to e$ making the evident diagrams commute. This arrow can be labeled

as “c”, insofar as it is “c” (labeled x), “p”; or as “d”, insofar as it is “d” (labeled y), “q”

$$\begin{tikzcd}
\boxed{\text{a .wav file}} \arrow[r, "\text{labeled ``W'' is}"] \arrow[dr, bend right, "\text{can be played on an element of}"'] & \boxed{\begin{array}{c}\text{a .wav file (labeled}\\ \text{``W'') or a .mp3}\\ \text{file (labeled ``M'')}\end{array}} \arrow[d, "\begin{array}{c}\text{as a .mp3 file, insofar as it is a .mp3 file (labeled ``M''), can}\\ \text{be played on an element of; or as a .wav file, insofar as it is}\\ \text{a .wav file (labeled ``W''), can be played on an element of}\end{array}"] & \boxed{\text{a .mp3 file}} \arrow[l, "\text{labeled ``M'' is}"'] \arrow[dl, bend left, "\text{can be played on an element of}"] \\
& \boxed{\begin{array}{c}\text{the set of modern}\\ \text{computers which}\\ \text{can play both .mp3}\\ \text{and .wav files}\end{array}} &
\end{tikzcd}$$

2.5.1.2

The pullback is $\{(x_1, z_1, y_1), (x_2, z_2, y_2), (x_3, z_2, y_2), (x_2, z_2, y_4), (x_3, z_2, y_4)\}$.

2.5.1.3

Omitted

2.5.1.5

We have $X \times_Z \emptyset = \emptyset$. We have $X \times_{\{*\}} Y = X \times Y$, the Cartesian product.

2.5.1.6

We have $W_1 = \{\, (\smile, f_1(\smile), y) \,|\, g_1(y) = f_1(\smile) \,\}$. When we take the third projection, we get the set of space-time coordinates whose spatial component corresponds to the center of mass of MIT at the time of it's founding.

We have $W_2 = \{\, (\smile, f_2(\smile), y) \,|\, g_2(y) = f_2(\smile) \,\}$. When we take the third projection, we get the set of space-time coordinates whose temporal component corresponds to the time of MIT's founding.

2.5.1.10

Top: Looks fine to me.

Middle: mostly-all-right, except for maybe that dog's don't necessarily have unique owners.

Bottom: "good fit"s, whatever that might mean, are generally not uniquely associated with a piece of furniture, nor with a width. (What is a "good fit", as a noun, rather than an adjective, anyway?)

2.5.1.11

Yes, the square where two pairs are projected from a triplet, and then both project to their common component, is a pullback square.

Under the simplified world-view that every child is legitimate and from a couple who are in their first marriage, the pullback of the diagram on the left is ⌜a brother⌝, and the pullback of the diagram on the right is ⌜a sister⌝

$$\begin{tikzcd} & \boxed{\text{a male person}} \arrow[d, "\text{has}"] \\ \boxed{\text{a female person with at least two children}} \arrow[r, "\text{is}"] & \boxed{\text{a mother}} \end{tikzcd} \qquad \begin{tikzcd} & \boxed{\text{a female person}} \arrow[d, "\text{has}"] \\ \boxed{\text{a female person with at least two children}} \arrow[r, "\text{is}"] & \boxed{\text{a mother}} \end{tikzcd}$$

2.5.1.13

$$\begin{tikzcd} f^{-1}(y) \arrow[r, "f"] \arrow[d, "f"] \arrow[dr, phantom, "\lrcorner"] & \{y\} \arrow[d, hook] \\ Y \arrow[r, "f"] & Y \end{tikzcd}$$