Universal Constructions

1 Terminal objects

Objects that have a unique morphism coming into them from every other object in the category

$$\forall c \in \zeta \exists ! c \to T \tag{1}$$

If there are more than one such objects, they are isomorphic

1.1 Example

In a category of sets, the terminal object is a singleton set, because there can be only one unique function from any set to a singleton set.

1.2 Code

```
data Terminal = MkTerm
bang :: a -> Terminal
bang _ = MkTerm
```

In haskell, the object `()` is a terminal object

2 Initial object

Objects that have a unique morphism going out to every other object in the category

$$\forall c \in \zeta \exists ! I \to c \tag{2}$$

If there are more than one such objects, they are isomorphic

2.1 Example

In a category of sets, the initial object is a null set, because there can be only one unique function from a null set to any other set.

2.2 Code

```
data Void
absurd :: Void -> a
absurd x = absurd x
```

3 Products

A product of two objects a and b is an object $a \times b$ that has projections π_1 and π_2 to a and b respectively. These projections factorize any other projections that exist from other objects c (potential products) to a and b

3.1 Example

A pair/tuple is a product

3.2 Code

```
data Pair a b = MkPair a b
fst :: Pair a b -> a
fst (MkPair a _) = a
snd :: Pair a b -> b
snd (MkPair _ b) = b

-- For potential products c
tuple :: (c->a, c->b) -> c -> Pair a b
tuple f g = \c -> MkPair (f c) (g c)

untuple :: (c -> Pair a b) -> (c->a, c->b)
untuple h = (fst . h, snd . h)

-- Bifunctor
-- Code easily obtained by visualizing that Pair a' b' is
-- a product of a' and b', and thinking of Pair a b as a potential
-- product c. So what we are basically doing is finding a mapping
-- from potential product to actual product
```

```
bimap :: (a->a', b->b') -> Pair a b -> Pair a' b'
bimap f g = tuple (f . fst) (g . snd)
```

4 Coproducts

If $c, d \in Ob(\zeta)$, then the coproduct of c and d consists of:

1. An object $c \oplus d \in Ob(\zeta)$
2. A morphism $Left : c \to c \oplus d$
3. A morphism $Right : d \to c \oplus d$

satisfying the property that $\forall X \in Ob(\zeta)$, and morphisms $L : c \to X$, $R : d \to X$ there exists a unique morphism $c \oplus d \to X$, and the diagram commutes.

4.1 Code

```
data Either a b = Left a | Right b
-- Left :: a -> Either a b
-- Right :: b -> Either a b

either :: (c -> x, d -> x) -> Either c d -> x
either (f, g) (Left c) = f c
either (f, g) (Right d) = g d

-- Bifunctor
-- Code can be easily obtained by drawing a diagram
bimap :: (c->c', d->d') -> Either c d -> Either c' d'
bimap (f, g) = either (Left . f) (Right . g)
```

5 Exponentials

Given a function from a product (a, b) to c, we can always get a function from a to a function $f : b \to c$. A function $f : b \to c$ is an exponential object c^b

5.1 Code

```
curry :: ((a, b) -> c) -> a -> b -> c
curry f = \a -> (\b -> f (a,b))
```

```
-- And the counterpart
uncurry :: (a -> (b -> c) -> ((a, b) -> c)
uncurry f = \(a,b) -> (f a) b
```