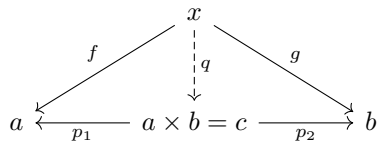


Seminář 3: Produkty, koprodukty a algebraické datové typy (ADT)

Produkty



- Objekt $c \in \mathcal{C}$ s projekcemi $p_1 : c \rightarrow a$ a $p_2 : c \rightarrow b$ je produkt objektů a a $b \iff \forall x \in \mathcal{C}. \forall f : x \rightarrow a, g : x \rightarrow b. \exists ! q : x \rightarrow c. (f = p_1 \circ q \wedge g = p_2 \circ q)$
- q faktorizuje f a $g \iff f = p_1 \circ q \wedge g = p_2 \circ q$
- q je univernální mapování

Příklady

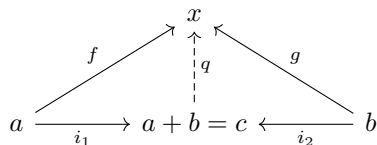
- V **Set** kartézský součin
- V **Poset** (částečně uspořádaná množina) *infimum*

Haskell

```
data (,) a b = (,) a b
fst (x, _) = x -- p_1
snd (_, y) = y -- p_2
```

```
q :: (j -> a) -> (j -> b) -> j -> (,) a b
q f g x = (f x, g x)
```

Koprodukty (sumy)



- Objekt $c \in \mathcal{C}$ se zahrnutími (*inkluzemi*) $i_1 : a \leftarrow c$ a $i_2 : b \leftarrow c$ je koprodukt objektů a a $b \iff \forall x \in \mathcal{C}. \forall f : x \leftarrow a, g : x \leftarrow b. \exists ! q : x \leftarrow c. (f = q \circ i_1 \wedge g = q \circ i_2)$

Příklady

- V **Set** *disjunktní sjednocení*
- V **Poset** (částečně uspořádaná množina) *supremum*

Haskell

```
data Either a b = Left a | Right b
-- i_1 = Left
-- i_2 = Right
```

```
q :: (a -> j) -> (b -> j) -> Either a b -> j
q f _ (Left x) = f x
q _ g (Right x) = g x
```

Bifunktory

Produkt kategorií $\mathcal{C} \times \mathcal{D}$

Skládá se z

- $O_{\mathcal{C} \times \mathcal{D}} = \{(c, d) | c \in O_{\mathcal{C}}, d \in O_{\mathcal{D}}\}$
- $A_{\mathcal{C} \times \mathcal{D}} = \{(f, g) | f : a \rightarrow c \in A_{\mathcal{C}}, g : b \rightarrow d \in A_{\mathcal{D}}\}$

Musí platit

- *identita* pro (a, b) je (id_a, id_b)
- $(f, h) \circ (g, k) \equiv (f \circ g, h \circ k)$

Bifunktor $F =$ funktor z produktu kategorií $F : \mathcal{C} \times \mathcal{D} \rightarrow \mathcal{E}$

V každé kategorii \mathcal{C} , kde pro každé dva objekty a a b můžeme najít jejich produkt, existuje bifunktor \times (zapisován infixově).

$$\begin{aligned} \times &: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C} \\ &: (a, b) \mapsto a \times b \\ &: (f : a \rightarrow a', g : b \rightarrow b') \mapsto (f \times g : a \times b \rightarrow a' \times b') \end{aligned}$$

$$\begin{array}{ccccc} a & \xleftarrow{p_1} & a \times b & \xrightarrow{p_2} & b \\ f \downarrow & & \downarrow f \times g & & \downarrow g \\ a' & \xleftarrow{p'_1} & a' \times b' & \xrightarrow{p'_2} & b' \end{array}$$

Bifunktory v Haskellu

```
class Bifunctor f where
  bimap :: (a -> b) -> (c -> d) -> f a c -> f b d
  bimap g h = first g . second h
  first :: (a -> b) -> f a c -> f b c
  first g = bimap g id
  second :: (c -> d) -> f a c -> f a d
  second h = bimap id h
  {-# MINIMAL bimap | first, second #-}
```

```
instance Bifunctor (,) where
  bimap g h (x, y) = (g x, h y)
```

```
instance Bifunctor Either where
  bimap g _ (Left x) = Left (g x)
  bimap _ h (Right y) = Right (h y)
```

ADT a funktory

- Pro každý ADT umí GHC instancovat funktor s pomocí rozšíření (od GHC 9.2.1 bez rozšíření)

```
{-# LANGUAGE DerivingFunctor #-}
data Example a = Ex a Int (Example a) (Example Int)
  deriving (Functor)
```