Yoneda

1 Recap

1.1 Natural transformation

Mějme funktory $F, G : \mathcal{C} \to \mathcal{D}$. Přirozená transformace $\alpha : F \Rightarrow G$ je soubor morfismů:

$$\alpha = \{\alpha_a : F\ a \to G\ a \mid a \in \mathcal{C}\},$$

kde pro všechny morfismy $f : a \to b, f \in \mathcal{C}$ následující diagram komutuje:

$$\begin{tikzcd} Fa \arrow[r, "\alpha_a"] \arrow[d, "Ff"] & Ga \arrow[d, "Gf"] \\ Fb \arrow[r, "\alpha_b"] & Gb \end{tikzcd}$$

Pokud všechny α_a jsou izo, pak F a G jsou přirozeně izomorfní. Pokud C je malá kategorie, pak $Nat(F, G)$ bude značit množinu všech přirozených transformací z F do G.

1.2 Hom-functor

Mějme malou kategorii $\mathcal{C}$. Pak pro každé dva objekty $a, b \in \mathcal{C}$, $\mathcal{C}(a, b)$ je množina morfismů z a do b. Zafixujme si objekt $c \in \mathcal{C}$. Pak $\mathcal{C}(c, -) : \mathcal{C} \to \mathbf{Set}$ budeme nazývat hom-funktor.

$$\begin{tikzcd} a \arrow[r, mapsto, "\mathcal{C}(c,-)"] \arrow[d, "f"] & \mathcal{C}(c, a) \arrow[d, "\mathcal{C}(c,f)"] \\ b \arrow[r, mapsto, "\mathcal{C}(c,-)"] & \mathcal{C}(c, b) \end{tikzcd}$$

Jak ale vypadá $\mathcal{C}(c, f)$, aby nám diagram výše komutoval?
$\mathcal{C}(c, f) : (g : c \to a) \mapsto (f \circ g : c \to b)$

Homfunktory jsou fajn, proto máme následující definici: reprezentovatelné funktory jsou funktory přirozeně izomorfní nějakému hom-funktoru.

!!Motivace – full+faithfull $\Rightarrow$ izo můžeš vracet, yoneda emb jsou ff!! ex F (iso), F-1(iso)

Ekvivalentní definice koproduktu: $a + b$ existuje $\iff$ $C(a + b, -) \cong C(a, -) \times C(b, -)$.

Ekvivalentní definice iniciálního objektu: a je iniciální $\iff$ $C(a, -) \cong 1$.

Příklad – F set-valued funktory $(\mathbb{N}, \leq)$, N_i homfunktory. Obraz F budou F(0), F(1) ... Obraz N_i jsou $\emptyset$ nebo $*$. No a jak budou vypadat ty přirozené transformace?

2 Yoneda

Mějme $\mathcal{C}$ malou kategorii, $c \in \mathcal{C}$ a funktor $F : \mathcal{C} \to \mathbf{Set}$. Yonedovo lemma tvrdí:

$$Nat(\mathcal{C}(c, -), F) \cong Fc$$

Co přesně to ale znamená? Vezměme si nějakou přirozenou transformaci α a nakresleme si diagram:

$$\begin{tikzcd}
\mathcal{C}(c,x) \arrow[r, "\mathcal{C}(c,f)"] \arrow[d, "\alpha_x"] & \mathcal{C}(c,y) \arrow[d, "\alpha_y"] \\
Fx \arrow[r, "Ff"] & Fy
\end{tikzcd}$$

Pokud si za x dosadíme c, získáme ještě zajímavější diagram

$$\begin{tikzcd}
\mathcal{C}(c,c) \arrow[r, "\mathcal{C}(c,f)"] \arrow[d, "\alpha_c"] & \mathcal{C}(c,y) \arrow[d, "\alpha_y"] \\
Fc \arrow[r, "Ff"] & Fy
\end{tikzcd}$$

Jelikož $id_c \in \mathcal{C}(c, c)$, dostáváme z předchozího diagramu:

$$\alpha_y \ (f \circ id_c) = (F \ f) \ (\alpha_c \ id_c)$$

Což se dá dále zkrátit na

$$\alpha_y(f) = (F \ f) \ (\alpha_c \ id_c)$$

Pokud "máme v ruce" funktor F a hodnotu morfismu α_c v bodě id_c, pak můžeme zadefinovat přirozenou transformaci α následujícím způsobem (poznámka: α_x je zobrazení mezi množinou funkcí $\mathcal{C}(c, x)$ a množinou Fx!):

$$\alpha_x(f) = (F \ f) \ (\alpha_c \ id_c)$$

!!Navíc tedy pokud by dvě přirozené transformace měly stejnou hodnotu morfismu α_c v bodě id_c, pak by tyto přirozené transformace byly stejné (je zřejmé, že pokud pošle na různé, dostaneme různé transformace)!! K tomu α_c posílá prvky do Fc. Zbývá nám se tedy zamyslet, jestli pro každý prvek Fc existuje přirozená transformace, která na něj posílá id_c. Jednotlivé morfismy takové přirozené transformace jsou ale v **Set**! Jako domácí cvičení si můžete dokázat, že přirozené transformace zadefinované výše opravdu splňují definici přirozené transformace (dostaneme přesně obrázek výše).

Pokud "máme v ruce" přirozené transformace α:

$$F \ c = \{\alpha_c \ id_c \mid \alpha : \mathcal{C}(c, -) \Rightarrow F\}$$

Dokázali jsme tedy neformálně Yonedovo lemma. Hm, co když $|Fa| = 0$? Intuice – reprezentovatelné funktory jsou nejmíň lossy, další se dají získat z homfunktorů, nějaká informace se může ztratit (něco jako "nesurjektivní transformace").

Mějme kategorie $\mathcal{C}, \mathcal{D}$. Pak $\mathbf{Func}[\mathcal{C}, \mathcal{D}]$ je kategorie, kde objekty jsou funktory z $\mathcal{C}$ do $\mathcal{D}$ a morfismy jsou přirozené transformace mezi nimi. Pak Yonedovo lemma nám tvrdí:

$$\mathbf{Func}[\mathcal{C}, \mathbf{Set}](\mathcal{C}(c, -), F) \cong Fc$$

2.1 Cayley from Yoneda

Asi přeskočím.

Cayley – G se dá vložit do S(G).

Cayley z yonedy – $\mathbf{G}$ jednoprvková kategorie, kde morfismy odpovídají prvkům grupy G. $Nat(\mathbf{G}(\star, -), F) \cong F\star$, jediná možnost protože je tam jen jeden objekt. Nyní můžeme za F zvolit zase hom-funktor:
$Nat(\mathbf{G}(\star, -), \mathbf{G}(\star, -)) \cong \mathbf{G}(\star, \star) \sim G$.

Navíc každá přirozená transformace $\alpha = \{\alpha_a : \mathbf{G}(\star, a) \to \mathbf{G}(\star, a) \mid a \in \mathbf{G}\} = \{\alpha_\star : \mathbf{G}(\star, \star) \to \mathbf{G}(\star, \star)\}$. Pro tyto přirozené transformace a každý morfismus f následující diagram musí komutovat: (označím si $\mathfrak{G} := \mathbf{G}(\star, \star)$, $F := \mathbf{G}(\star, -)$)

$$\begin{array}{ccc} \mathfrak{G} & \xrightarrow{\alpha_\star} & \mathfrak{G} \\ \downarrow{\scriptstyle Ff} & & \downarrow{\scriptstyle Ff} \\ \mathfrak{G} & \xrightarrow{\alpha_\star} & \mathfrak{G} \end{array}$$

Tedy že $\alpha_\star \circ \mathbf{G}(\star, f) = \mathbf{G}(\star, f) \circ \alpha_\star$. Ale $\mathbf{G}(\star, f) : (g : \star \to \star) \mapsto (f \circ g : \star \to \star)$. Tedy pro libovolný morfismus g (tedy prvek grupy G) musí platit rovnost:

$$\alpha_\star(f \circ g) = f \circ \alpha_\star g \iff \alpha_\star(f \cdot g) = f \cdot \alpha_\star(g)$$

Pokud za g dosazíme identický prvek, dostáváme následující rovnost:

$$\alpha_\star(f) = \alpha_\star(f \cdot 1) = f \cdot \alpha_\star(1) = f \cdot h$$

Přirozené transformace tedy odpovídají násobení nějakým prvkem $h \in G$, z algebry 1 víme, že tyto zobrazení jsou bijekce. Ukázali jsme tedy, že grupa G je izomorfní nějaké množině bijekcí, cože přesně tvrdí Cayley.

3 Yoneda in Haskell

Jak jsme si řekli v předchozích přednáškách, endofunktory v **Hask** jsou právě unární typové konstruktory s 'fmap' a přirozené transformace jsou polymorfní funkce. Abychom ale využili yonedu, museli bychom mít set-valued funktory. Co s tím?

Forgetfull functor U, ex U: Mon to Set. Přivřeme ale oči a budeme se nyní dívat na všechny funktory F jako UF.

"homfunktor" `Reader a x = a → x`

"trans" `alpha :: forall x. (a → x) → F x`

"yoneda" `forall x. (a → x) → F x ≅ F a`

"coyoneda" `forall x. (x → a) → F x ≅ F a`

Co nám říká yoneda? Přirozené transformace přesně odpovídají prvkům `F a`, tedy každá polymorfní funkce typu "trans" uchovává přesně stejnou informaci jako prvek `F a`.

Tedy pokud máme nějakou polymorfní funkci typu "trans", tak můžu vyprodukovat nějaký prvek `F a` (`alpha id`). Naopak z prvku `fa :: F a` můžeme vytvořit polymorfní funkci (`alpha h = fmap h fa`).

Výhody přecházení mezi těmito dvěma reprezentacemi? Jedna může být v nějakých částech efektivnější, nebo jednodušší na použití.

Použití: konstrukce kompilátoru – continuation passing style. Yoneda nám říká (s identickým funktorem: `forall x. (a → x) → x ≅ a`, že libovolný typ a může být nahrazen handlerem pro a – handla vezme něco typu a a pokračuje ve výpočtu.

4 Yoneda embedding

Mějme nějakou malou kategorii $\mathcal{C}$ a $c \in \mathcal{C}$. Před chvílí jsme viděli, že $\mathcal{C}(c, -) : \mathcal{C} \to \mathbf{Set}$. Nic nám ale nebrání si vytvořit zobrazení Y, který objektům kategorie $\mathcal{C}$ bude přiřazovat odpovídající hom-funktory – bude to tedy zobrazení $Y : \mathcal{C} \to \mathbf{Func}[\mathcal{C}, \mathbf{Set}]$, $a \mapsto \mathcal{C}(a, -)$. Aby to ale byl funktor, musíme se rozhodnout, kam ale budeme zobrazovat morfismy z $\mathcal{C}$.

Z yonedy víme, že $\mathbf{Func}[\mathcal{C}, \mathbf{Set}](\mathcal{C}(a, -), F) \cong Fa$. Při dosazení $F := \mathcal{C}(b, -)$ tedy dostáváme, že $\mathbf{Func}[\mathcal{C}, \mathbf{Set}](\mathcal{C}(a, -), \mathcal{C}(b, -)) \cong \mathcal{C}(b, a)$. Skoro se nám to líbí, ale nějak se nám "obrátily šipečky". No, budeme tedy muset posílat do $\mathbf{Func}[\mathcal{C}^{op}, \mathbf{Set}]$

Tomuto funktoru se říká yonedovo vložení. Bude to teda funktor $Y_\star : \mathcal{C} \to \mathbf{Func}[\mathcal{C}^{op}, \mathbf{Set}]$, $a \mapsto \mathcal{C}(-, a)$, a morfismus $f : a \to b$ na přirozenou transformaci α (z $\mathbf{Func}[\mathcal{C}, \mathbf{Set}](\mathcal{C}(-, a), \mathcal{C}(-, b))$):

$$\alpha_c(g) = f \circ g \ (g : c \to a)$$

Jak duálníhoho yonedu, tak máme i duální yonedovo vložení: $Y^\star : \mathcal{C}^{op} \to \mathbf{Func}[\mathcal{C}, \mathbf{Set}]$, $a \mapsto \mathcal{C}(a, -)$. Tento funktor ale bude "kontravariantní" – bude obracet šipečky. Morfismus $f : a \to b$ pošle na přirozenou transformaci α (z $\mathbf{Func}[\mathcal{C}, \mathbf{Set}](\mathcal{C}(b, -), \mathcal{C}(a, -))$):

$$\alpha_c(g) = g \circ f \ (g : b \to c)$$

... a ten kovariantní je hezčí! A proč jsme se teda o tomto furt bavili? Obě tyto vložení jsou ff! Navíc toho o Setech víme dost, takže se tam dost často lépe pracuje.

$$
\begin{tikzcd}
\mathsf{C} \arrow[rr, hook, "y"] & & \mathsf{Set}^{\mathsf{C}^{op}} \\
c \arrow[d, "f"'] & \mapsto & \mathsf{C}(-,c) \arrow[d, "f_*"] \\
d & \mapsto & \mathsf{C}(-,d)
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{C}^{op} \arrow[rr, hook, "y"] & & \mathsf{Set}^{\mathsf{C}} \\
c \arrow[d, "f"'] & \mapsto & \mathsf{C}(c,-) \\
d & \mapsto & \mathsf{C}(d,-) \arrow[u, "f^*"']
\end{tikzcd}
$$

Ve skutečnosti to jsou dvě inkarnace (částečná aplikace) bifunktoru $\mathcal{C}(-,-) : \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$. Posílá dvojici $(x, y) \mapsto C(x, y)$ a dvojici morfismů $f : w \to x$, $h : y \to z$ pošle na funkci $(g \mapsto hgf)$.

5 YEiH

Yonedovo vložení (to druhé, hezčí) se dá reprezentovat jako izomorfismus mezi reader funktory a funkcemi v opačném směru, i.e.:
`forall x. (a -> x) -> (b -> x)` $\cong$ `b -> a`.

Mějme tedy funkci `BtoA :: B->A`. Poté následujícím způsobem můžeme zadefinovat levou stranu:
`fromY f b = f (BtoA b)`.

Naopak pokud máme takovou funkci `fromY :: (a -> x) -> (b -> x)`, pak pravou stranu můžeme zadefinovat:
`BtoA b = fromY id b`