

Bevezetés a homotópia-típuselméletbe

Kaposi Ambrus

University of Nottingham
Functional Programming Lab

ELTE Eötvös Kollégium
2014. február 21.

Kifejezőbb típusrendszert!

- ▶ A típusok a biztonságos programozást segítik elő
- ▶ Ha nem elég kifejező a típusrendszer, az megakadályozza az absztrakciót
- ▶ pl. `length` program típusa:
 - ▶ `List Int → Int`
 - ▶ `List Char → Int`
 - ▶ System F (Haskell): $\forall a : \text{Type} . \text{List } a \rightarrow \text{Int}$
- ▶ egy adott $n : \mathbb{N}$ -nél kisebb számok típusa:
 - ▶ `Fin1 : Type`
 - ▶ `Fin2 : Type`
 - ▶ Függő típusok (Agda): `Fin : $\mathbb{N} \rightarrow \text{Type}$`

Curry-Howard izomorfizmus

- ▶ Logika és típuselmélet közti kapcsolat
- ▶ állítás = típus, bizonyítás = program
- ▶ példák:

$$A \wedge B \Rightarrow C \vee D$$

$$a = b$$

$$\neg(x = 3)$$

$$\forall x \in \mathbb{N}. x + \text{zero} = x$$

$$\text{List Char} \Rightarrow \text{Int}$$

$$A \times B \rightarrow C + D$$

$$\text{ld}(a, b)$$

$$\text{ld}_{\mathbb{N}}(x, 3) \rightarrow 0$$

$$\prod_{x:\mathbb{N}} \text{ld}_{\mathbb{N}}(x + \text{zero}, x)$$

$$(x : \mathbb{N}) \rightarrow \text{ld}_{\mathbb{N}}(x + y, y + x)$$

$$\text{List Char} \rightarrow \text{Int}$$

- ▶ Matematikát elsőrendű logikában végeznek: ha ezt képes kifejezni a típusrendszerünk, akkor minden matematikai állítás leírható vele

Példa

- ▶ \mathbb{N} típust a konstruktorokkal adjuk meg:

$$\text{zero} : \mathbb{N}$$

$$\text{suc} : \mathbb{N} \rightarrow \mathbb{N}$$

- ▶ az összeadás ($_ + _ : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$) függvényt rekurzívan definiáljuk:

$$\text{zero} + n : \equiv n$$

$$\text{suc}(m) + n : \equiv \text{suc}(m + n)$$

- ▶ rekurzívan definiáljuk az alábbi függvényt:

$$\text{assoc} : \prod_{x,y,z:\mathbb{N}} \text{Id}_{\mathbb{N}}((x + y) + z, x + (y + z))$$

$$\text{assoc}(\text{zero}, y, z) : \equiv \text{refl}(y + z)$$

$$\text{assoc}(\text{suc}(m), y, z) : \equiv \text{cong}(\text{suc}, \text{assoc}(m, y, z))$$

Problémák

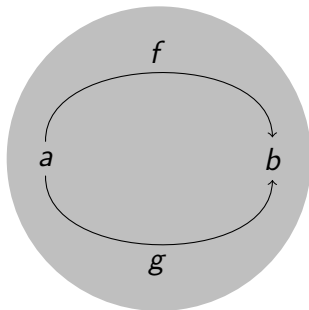
- ▶ Pontonként egyenlő függvények nem egyenlők: pl. nincs eleme az $\text{Id}_{\mathbb{N} \rightarrow \mathbb{N}}(\lambda x.x, \lambda x.x + \text{zero})$ típusnak
- ▶ Izomorf típusok egyenlősége: ha tudjuk, hogy $A \cong B$, és
 - ▶ van egy $A \rightarrow A \rightarrow A$ műveletünk, akkor automatikusan megkapjuk a megfelelő $B \rightarrow B \rightarrow B$ műveletet
 - ▶ ha tudjuk, hogy $\forall a : A.P(a)$, akkor automatikusan megkapjuk, hogy a P tulajdonság igaz B minden elemére is. Pl. $A \equiv (\text{List}, _ :: _, \dots)$, $B \equiv (\text{RBT}, \text{insert}, \dots)$, P azt mondja, hogy kétszer ugyanazt beillesztve ugyanazt a struktúrát kapjuk
- ▶ Ekvivalencia-osztályok definiálása
- ▶ Emiatt kényelmetlen a típuselmélet használata, programokról is nehezebb bizonyításokat végezni

Egyenlőség típus

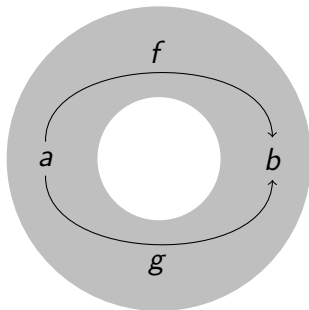
- ▶ az egyenlőség típus egyetlen konstruktora $\text{refl}(a) : \text{Id}_A(a, a)$
- ▶ Ha $p, q : \text{Id}_A(x, y)$, akkor igaz -e, $p = q$, vagyis $\text{Id}(p, q)$?
 - ▶ K axióma, mintaillesztés
 - ▶ groupoid modell (Hofmann-Streicher, 1996)
- ▶ Típusok, mint topologikus terek (Vladimir Voevodsky, 2006)



Korong

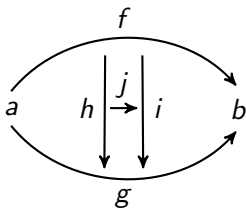


Gyűrű



Magasabb egyenlőségek

- ▶ $f, g : \text{Id}(a, b)$
- ▶ $h, i : \text{Id}_{\text{Id}(a,b)}(f, g)$
- ▶ $j : \text{Id}_{\text{Id}(f,g)}(h, i)$



Homotópia-szintek

- ▶ A típusok szintekbe sorolhatók:

-2. szint A kontraktibilis

-1. szint A egy állítás (propozíció)

0. szint A egy halmaz

1. szint A egy groupoid

...

- ▶ Bool , \mathbb{N} , List Bool , $\text{Tree } \mathbb{N}$ stb. típusok mind halmazok
- ▶ Mi a magasabb egyenlőségek forrása?
 - ▶ Típusok egyenlősége
 - ▶ Magasabb induktív típusok

$a : A$ és $\prod_{x:A} \text{Id}_A(a, x)$

$\prod_{x,y:A} \text{Id}_A(x, y)$

$\prod_{x,y:A,p,q:-\text{Id}_A(x,y)} \text{Id}(p, q)$

(két elem csak -
egyféleképpen lehet
egyenlő)

(egyenlőségek

egyenlőségei egyenlők)

Univalence

- ▶ Topologikus terek egyenlősége azizomorfizmus.
- ▶ Voevodsky univalence axiómája azt mondja*, hogy ha két típus izomorf, akkor egyenlő: $\text{univ} : (A \cong B) \rightarrow \text{Id}_{\text{Type}}(A, B)$.
- ▶ Ebből következik az izomorf típusok egyenlősége és a pont szerint egyenlő függvények egyenlősége is
- ▶ Pl. az $\text{Id}(\text{Bool}, \text{Bool})$ típus elemeit a $\text{Bool} \rightarrow \text{Bool}$ izomorfizmusok generálják, és ebből kettő van:

$\text{iso}_1 : \text{Bool} \rightarrow \text{Bool}$

$\text{iso}_1(b) :\equiv b$

$\text{iso}_2 : \text{Bool} \rightarrow \text{Bool}$

$\text{iso}_2(\text{true}) :\equiv \text{false}$

$\text{iso}_2(\text{false}) :\equiv \text{true}$

- ▶ Belátható, hogy $\neg(\text{Id}(\text{univ}(\text{iso}_1), \text{univ}(\text{iso}_1)))$

Magasabb induktív típusok

- ▶ Az egész számok \mathbb{Z} típusát az alábbi konstruktorok generálják:

$$\text{minus} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{Z}$$

$$\text{quot} : \prod_{a,b,c,d:\mathbb{N}} \text{Id}_{\mathbb{N}}(a + d, c + b) \rightarrow \text{Id}_{\mathbb{Z}}(\text{minus}(a, b), \text{minus}(c, d))$$

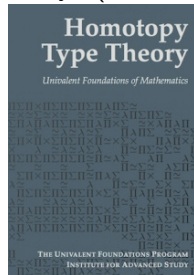
$$\text{set} : \prod_{(x,y:\mathbb{Z})} \prod_{p,q:\text{Id}_{\mathbb{Z}}(x,y)} \text{Id}_{\text{Id}_{\mathbb{Z}}(x,y)}(p, q)$$

Tennivalók

- ▶ Az Id típus egyedüli konstruktora a $\text{refl}(a) : \text{Id}(a, a)$ volt
- ▶ Az univalence axióma egy új konstruktort ad, izomorfizmussal is lehet egyenlőséget generálni
- ▶ Nem igaz, hogy bármely $n : \mathbb{N}$, ahol n -ben nincs szabad változó, egyenlő $\text{suc}^k(\text{zero})$ -val valamely k természetes számra
- ▶ Emiatt nem használható programozási nyelvként (bizonyos programok futása elakad)

Kipróbálnám

- ▶ Magyarul: honlapomon cikk + ez az előadás
- ▶ Könyv (Princeton, 2013), blog: <http://homotopytypetheory.org>



- ▶ Martin-Löf típuselméletre épülő programozási nyelvek:
 - ▶ Coq: legrégebbi, inkább tételbizonyító rendszerként használják
 - ▶ Agda: homotópia-típuselméletre legalkalmasabb
 - ▶ Idris: gyakorlati programozásra, homotópia-típuselméletre nem használható

Köszönöm szépen a figyelmet!