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` $\to$ `Int`
  - `List Char` $\to$ `Int`
  - System F (Haskell): $\forall$ `a : Type . List a` $\to$ `Int`
- egy adott $n : \mathbb{N}$-nél kisebb számok típusa:
  - `Fin1 : Type`
  - `Fin2 : Type`
  - Függő típusok (Agda): `Fin :` $\mathbb{N} \to$ `Type`

Curry-Howard izomorfizmus

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

$A \wedge B \Rightarrow C \vee D$	$A \times B \to C + D$
$a = b$	$\mathsf{Id}(a, b)$
$\neg(x = 3)$	$\mathsf{Id}_{\mathbb{N}}(x, 3) \to 0$
$\forall x \in \mathbb{N}. x + \mathsf{zero} = x$	$\prod_{x:\mathbb{N}} \mathsf{Id}_{\mathbb{N}}(x + \mathsf{zero}, x)$
	$(x : \mathbb{N}) \to \mathsf{Id}_{\mathbb{N}}(x + y, y + x)$
`List Char` $\Rightarrow$ `Int`	`List Char` $\to$ `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:

$$\mathsf{zero} : \mathbb{N}$$
$$\mathsf{suc} : \mathbb{N} \to \mathbb{N}$$

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

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

- rekurzívan definiáljuk az alábbi függvényt:

$$\mathsf{assoc} : \prod_{x,y,z:\mathbb{N}} \mathsf{Id}_{\mathbb{N}}((x + y) + z, x + (y + z))$$
$$\mathsf{assoc}(\mathsf{zero}, y, z) :\equiv \mathsf{refl}(y + z)$$
$$\mathsf{assoc}(\mathsf{suc}(\mathsf{m}), y, z) :\equiv \mathsf{cong}(\mathsf{suc}, \mathsf{assoc}(\mathsf{m}, \mathsf{y}, \mathsf{z}))$$

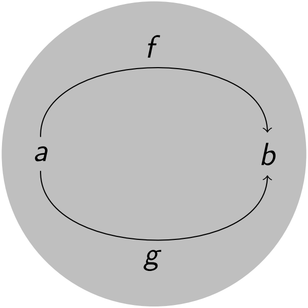
Problémák

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

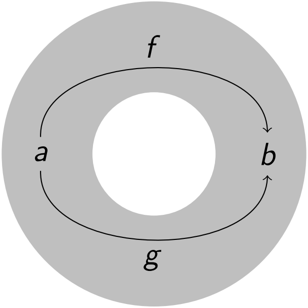
Egyenlőség típus

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

Korong

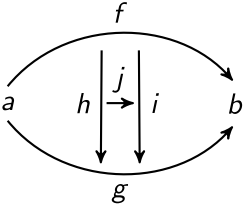


Gyűrű



Magasabb egyenlőségek

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



Homotópia-szintek

- A típusok szintekbe sorolhatók:

-2. szint	A kontraktibilis	$a : A$ és $\prod_{x:A} \mathsf{Id}_A(a, x)$
-1. szint	A egy állítás (propozíció)	$\prod_{x,y:A} \mathsf{Id}_A(x, y)$
0. szint	A egy halmaz	$\prod_{x,y:A,p,q:-\mathsf{Id}_\mathsf{A}(\mathsf{x},\mathsf{y})} \mathsf{Id}(p, q)$ (két elem csak - egyféleképpen lehet egyenlő)
1. szint	A egy groupoid	(egyenlőségek egyenlőségei egyenlők)

  ...

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

Univalence

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

$$\begin{aligned}
&\mathsf{iso}_1 : \mathsf{Bool} \to \mathsf{Bool}\\
&\mathsf{iso}_1(b) :\equiv b\\
&\mathsf{iso}_2 : \mathsf{Bool} \to \mathsf{Bool}\\
&\mathsf{iso}_2(\mathsf{true}) :\equiv \mathsf{false}\\
&\mathsf{iso}_2(\mathsf{false}) :\equiv \mathsf{true}
\end{aligned}$$

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

Magasabb induktív típusok

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

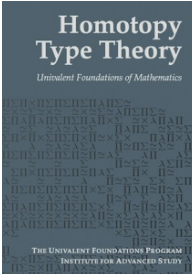
$$\begin{aligned}
\mathsf{minus} &: \mathbb{N} \to \mathbb{N} \to \mathbb{Z} \\
\mathsf{quot} &: \prod_{a,b,c,d:\mathbb{N}} \mathsf{Id}_{\mathbb{N}}(a+d, c+b) \to \mathsf{Id}_{\mathbb{Z}}(\mathsf{minus}(a,b), \mathsf{minus}(c,d)) \\
\mathsf{set} &: \prod_{(x,y:\mathbb{Z})} \prod_{p,q:\mathsf{Id}_{\mathbb{Z}}(x,y)} \mathsf{Id}_{\mathsf{Id}_{\mathbb{Z}}(x,y)}(p,q)
\end{aligned}$$

Tennivalók

- Az Id típus egyedüli konstruktora a $\mathsf{refl}(a) : \mathsf{Id}(a, a)$ 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 $\mathsf{suc}^k(\mathsf{zero})$-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!