# Típuselmélet a homotópia-típuselméletben

Kaposi Ambrus[1]
együttműködve Thorsten Altenkirch-el[2]

[1]ELTE, Programozási Nyelvek és Fordítóprogramok Tanszék
[2]University of Nottingham, Egyesült Királyság

2021. június 10.

NATIONAL RESEARCH, DEVELOPMENT AND INNOVATION OFFICE
HUNGARY

PROGRAM FINANCED FROM THE NRDI FUND

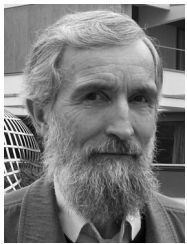
# Tartalom

- Típuselmélet
- Homotópia típuselmélet
- Folyamatban levő munka: típuselmélet a homotópia-típuselméletben
- Utóbbi év eredményeinek összefoglalása

# Típuselmélet

Programozási nyelv és a matematika nyelve egyszerre.

- 1972, Per Martin-Löf svéd matematikus (1942–)



# Típuselmélet mint programozási nyelv

Nagyon kifejező típusrendszer:

```
lnko0 : Int × Int → Int
lnko1 : (a : Int) × (b : Int) → (c : Int) × (c | a) × (c | b)
lnko2 : (a : Int) × (b : Int) → (c : Int) × (c | a) × (c | b) ×
                                  ((x : Int) × (x | a) × (x | b) → x | c)
```

# Típuselmélet mint programozási nyelv

Nagyon kifejező típusrendszer:

```
lnko0 : Int × Int → Int
lnko1 : (a : Int) × (b : Int) → (c : Int) × (c | a) × (c | b)
lnko2 : (a : Int) × (b : Int) → (c : Int) × (c | a) × (c | b) ×
                                  ((x : Int) × (x | a) × (x | b) → x | c)
```

"Index out of bounds" futási idejű hibák megszüntetése:

```
List : Type × Int → Type
head : List (1 + n , A) → A
_++_ : List (n , A) → List (m , A) → List (n + m , A)
```

# Típuselmélet mint programozási nyelv

Nagyon kifejező típusrendszer:

```
lnko0 : Int × Int → Int
lnko1 : (a : Int) × (b : Int) → (c : Int) × (c | a) × (c | b)
lnko2 : (a : Int) × (b : Int) → (c : Int) × (c | a) × (c | b) ×
                                  ((x : Int) × (x | a) × (x | b) → x | c)
```

"Index out of bounds" futási idejű hibák megszüntetése:

```
List : Type × Int → Type
head : List (1 + n , A) → A
_++_ : List (n , A) → List (m , A) → List (n + m , A)
```

Interaktív programok:

```
avail : (b : Bool) → if b then (Date → Bool) else (Data × Date → Bool)
```

# Típuselmélet mint a matematika nyelve

| | | |
|---|---|---|
| Program | $\supset$ | Bizonyítás |
| Típus | $\supset$ | Állítás |

# Típuselmélet mint a matematika nyelve

| | | |
|---|---|---|
| Program | $\supset$ | Bizonyítás |
| Típus | $\supset$ | Állítás |

- A bizonyítások végrehajthatók: ha belátjuk, hogy létezik legnagyobb közös osztó, kapunk egy programot, ami kiszámítja

# Típuselmélet mint a matematika nyelve

| | | |
|---|---|---|
| Program | $\supset$ | Bizonyítás |
| Típus | $\supset$ | Állítás |

- A bizonyítások végrehajthatók: ha belátjuk, hogy létezik legnagyobb közös osztó, kapunk egy programot, ami kiszámítja
- A bizonyítások számítógépes ellenőrzése reális, mert valódi függvényekkel, nem pedig azok gráfjaival dolgozunk, pl. $1 + 1 = 2$ bizonyítása triviális.
  Implementációk:
  - Automath: legelső, De Bruijn
  - Coq: négyszíntétel, Google Chrome, ELTE
  - Agda: kísérletezésre, ELTE
  - Idris: programozásra
  - Lean: Kevin Buzzard (Imperial College)

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

```
id  : Bool = Bool
not : Bool = Bool
```

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

      ```
      id  : Bool = Bool
      not : Bool = Bool
      ```

      A típusok homotópia-szintekbe sorolhatók:
    - állítás: tetszőleges két eleme egyenlő, pl. $3 > 1$, $\forall n.(n + 1 = n + 2)$

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

      ```
      id  : Bool = Bool
      not : Bool = Bool
      ```

      A típusok homotópia-szintekbe sorolhatók:
    - állítás: tetszőleges két eleme egyenlő, pl. $3 > 1$, $\forall n.(n + 1 = n + 2)$
    - halmaz: egyenlősége állítás, pl. $\mathbb{N}$, `Bool`, az összes eldönthető egyenlőséggel rendelkező típus, $A \to B$ ahol $B$ halmaz

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

      ```
      id  : Bool = Bool
      not : Bool = Bool
      ```

      A típusok homotópia-szintekbe sorolhatók:
    - állítás: tetszőleges két eleme egyenlő, pl. $3 > 1$, $\forall n.(n + 1 = n + 2)$
    - halmaz: egyenlősége állítás, pl. $\mathbb{N}$, `Bool`, az összes eldönthető egyenlőséggel rendelkező típus, $A \to B$ ahol $B$ halmaz
    - groupoid: egyenlősége halmaz, pl. a halmazok gyűjteménye, kör típus

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

      ```
      id  : Bool = Bool
      not : Bool = Bool
      ```

      A típusok homotópia-szintekbe sorolhatók:
    - állítás: tetszőleges két eleme egyenlő, pl. $3 > 1$, $\forall n.(n + 1 = n + 2)$
    - halmaz: egyenlősége állítás, pl. $\mathbb{N}$, `Bool`, az összes eldönthető egyenlőséggel rendelkező típus, $A \to B$ ahol $B$ halmaz
    - groupoid: egyenlősége halmaz, pl. a halmazok gyűjteménye, kör típus
    - ...

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

      ```
      id  : Bool = Bool
      not : Bool = Bool
      ```

      A típusok homotópia-szintekbe sorolhatók:
    - állítás: tetszőleges két eleme egyenlő, pl. $3 > 1$, $\forall n.(n + 1 = n + 2)$
    - halmaz: egyenlősége állítás, pl. $\mathbb{N}$, `Bool`, az összes eldönthető egyenlőséggel rendelkező típus, $A \to B$ ahol $B$ halmaz
    - groupoid: egyenlősége halmaz, pl. a halmazok gyűjteménye, kör típus
    - ...
    - $\infty$-groupoidok: egyik szinten sem triviális az egyenlőség, pl. Type

# Absztrakció

- Reprezentáció-függetlenség. Például egy `(A:Type) → A → A` típusú program csak az identitás lehet. Parametricitás, reláció-megőrzés, John Reynolds, 1983.
- A bijekcióban levő típusok megkülönböztethetetlenek.
  - Elsőrendű logika függő típusokkal. Makkai Mihály, 1995
  - Univalence: a típusok egyenlősége az ekvivalenca. Vladimir Voevodsky, 2006
    - Homotópia-típuselmélet, a matematika univalens megalapozása
    - Az egyenlőség nem állítás, hanem struktúra!

      ```
      id  : Bool = Bool
      not : Bool = Bool
      ```

      A típusok homotópia-szintekbe sorolhatók:
    - állítás: tetszőleges két eleme egyenlő, pl. $3 > 1$, $\forall n.(n + 1 = n + 2)$
    - halmaz: egyenlősége állítás, pl. $\mathbb{N}$, `Bool`, az összes eldönthető egyenlőséggel rendelkező típus, $A \to B$ ahol $B$ halmaz
    - groupoid: egyenlősége halmaz, pl. a halmazok gyűjteménye, kör típus
    - ...
    - $\infty$-groupoidok: egyik szinten sem triviális az egyenlőség, pl. Type

"Sejtés": a homotópia-típuselmélet az ideális nyelv programozásra/matematikára.

# Egy ismerős algebrai struktúra

$$
\begin{aligned}
&\mathsf{C} && : \mathsf{Type} \\
&- \otimes - && : \mathsf{C} \to \mathsf{C} \to \mathsf{C} \\
&\mathsf{u} && : \mathsf{C} \\
&-^{-1} && : \mathsf{C} \to \mathsf{C} \\
&\mathsf{ass} && : (a \otimes b) \otimes c = a \otimes (b \otimes c) \\
&\mathsf{idl} && : \mathsf{u} \otimes a = a \\
&\mathsf{idr} && : a \otimes \mathsf{u} = a \\
&\mathsf{invl} && : a^{-1} \otimes a = \mathsf{u} \\
&\mathsf{invl} && : a \otimes a^{-1} = \mathsf{u}
\end{aligned}
$$

# Egyszerű programozási nyelv, mint algebrai struktúra

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Type} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} \\
\mathsf{Bool} & : \mathsf{Ty} \\
\mathsf{Nat} & : \mathsf{Ty} \\
\mathsf{true} & : \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{false} & : \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{if\_then\_else\_} & : \mathsf{Tm}\,\mathsf{Bool} \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \\
\mathsf{num} & : \mathbb{N} \to \mathsf{Tm}\,\mathsf{Nat} \\
\mathsf{isZero} & : \mathsf{Tm}\,\mathsf{Nat} \to \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{if}\beta_1 & : \mathsf{if}\,\mathsf{true}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t \\
\mathsf{if}\beta_2 & : \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t' \\
\mathsf{isZero}\beta_1 & : \mathsf{isZero}\,(\mathsf{num}\,0) = \mathsf{true} \\
\mathsf{isZero}\beta_2 & : \mathsf{isZero}\,(\mathsf{num}\,(1+n)) = \mathsf{false}
\end{array}
$$

# Egyszerű programozási nyelv, mint algebrai struktúra

$$
\begin{array}{ll}
\mathsf{Ty} & : \mathsf{Type} \\
\mathsf{Tm} & : \mathsf{Ty} \to \mathsf{Type} \\
\mathsf{Bool} & : \mathsf{Ty} \\
\mathsf{Nat} & : \mathsf{Ty} \\
\mathsf{true} & : \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{false} & : \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{if\text{-}then\text{-}else\text{-}} & : \mathsf{Tm}\,\mathsf{Bool} \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \\
\mathsf{num} & : \mathbb{N} \to \mathsf{Tm}\,\mathsf{Nat} \\
\mathsf{isZero} & : \mathsf{Tm}\,\mathsf{Nat} \to \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{if}\beta_1 & : \mathsf{if}\,\mathsf{true}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t \\
\mathsf{if}\beta_2 & : \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t' \\
\mathsf{isZero}\beta_1 & : \mathsf{isZero}\,(\mathsf{num}\,0) = \mathsf{true} \\
\mathsf{isZero}\beta_2 & : \mathsf{isZero}\,(\mathsf{num}\,(1+n)) = \mathsf{false}
\end{array}
$$

A szabad algebra a szintaxis. Benne az egyenlőség eldönthető (tehát halmaz).

# Típuselmélet, mint algebrai struktúra

**Type theory**

$\mathsf{Con}_{\_} \quad : \mathbb{N} \to \mathsf{Set}$

$\mathsf{Ty}_{\_} \quad : \mathbb{N} \to \mathsf{Con}_i \to \mathsf{Set}$

$\mathsf{Sub} \quad : \mathsf{Con}_i \to \mathsf{Con}_j \to \mathsf{Set}$

$\mathsf{Tm} \quad : (\Gamma : \mathsf{Con}_i) \to \mathsf{Ty}_j\ \Gamma \to \mathsf{Set}$

Substitution calculus

$\cdot \quad : \mathsf{Con}_0$

$- \triangleright - \quad : (\Gamma : \mathsf{Con}_i) \to \mathsf{Ty}_j\ \Gamma \to \mathsf{Con}_{i \sqcup j}$

$-[-] \quad : \mathsf{Ty}_j\ \Delta \to \mathsf{Sub}\ \Gamma\ \Delta \to \mathsf{Ty}_j\ \Gamma$

$\mathsf{id} \quad : \mathsf{Sub}\ \Gamma\ \Gamma$

$- \circ - \quad : \mathsf{Sub}\ \Theta\ \Delta \to \mathsf{Sub}\ \Gamma\ \Theta \to \mathsf{Sub}\ \Gamma\ \Delta$

$\epsilon \quad : \mathsf{Sub}\ \Gamma\ \cdot$

$-, - \quad : (\sigma : \mathsf{Sub}\ \Gamma\ \Delta) \to \mathsf{Tm}\ \Gamma\ (A[\sigma]) \to$
$\mathsf{Sub}\ \Gamma\ (\Delta \triangleright A)$

$\pi_1 \quad : \mathsf{Sub}\ \Gamma\ (\Delta \triangleright A) \to \mathsf{Sub}\ \Gamma\ \Delta$

$\pi_2 \quad : (\sigma : \mathsf{Sub}\ \Gamma\ (\Delta \triangleright A)) \to \mathsf{Tm}\ \Gamma\ (A[\pi_1\ \sigma])$

$-[-] \quad : \mathsf{Tm}\ \Delta\ A \to (\sigma : \mathsf{Sub}\ \Gamma\ \Delta) \to \mathsf{Tm}\ \Gamma\ (A[\sigma])$

$[\mathsf{id}] \quad : A[\mathsf{id}] = A$

$[\circ] \quad : A[\sigma \circ \delta] = A[\sigma][\delta]$

$\mathsf{ass} \quad : (\sigma \circ \delta) \circ \nu = \sigma \circ (\delta \circ \nu)$

$\mathsf{idl} \quad : \mathsf{id} \circ \sigma = \sigma$

$\mathsf{idr} \quad : \sigma \circ \mathsf{id} = \sigma$

$\cdot\eta \quad : (\sigma : \mathsf{Sub}\ \Gamma\ \cdot) = \epsilon$

$\triangleright\beta_1 \quad : \pi_1\ (\sigma, t) = \sigma$

$\triangleright\beta_2 \quad : \pi_2\ (\sigma, t) = t$

$\triangleright\eta \quad : (\pi_1\ \sigma, \pi_2\ \sigma) = \sigma$

$, \circ \quad : (\sigma, t) \circ \delta = (\sigma \circ \delta, t[\delta])$

Function space

$\Pi \quad : (A : \mathsf{Ty}_i\ \Gamma) \to \mathsf{Ty}_j\ (\Gamma \triangleright A) \to \mathsf{Ty}_{i \sqcup j}\ \Gamma$

$\mathsf{lam} \quad : \mathsf{Tm}\ (\Gamma \triangleright A)\ B \to \mathsf{Tm}\ \Gamma\ (\Pi\ A\ B)$

$\mathsf{app} \quad : \mathsf{Tm}\ \Gamma\ (\Pi\ A\ B) \to \mathsf{Tm}\ (\Gamma \triangleright A)\ B$

$\Pi\beta \quad : \mathsf{app}\ (\mathsf{lam}\ t) = t$

$\Pi\eta \quad : \mathsf{lam}\ (\mathsf{app}\ t) = t$

$\Pi[] \quad : (\Pi\ A\ B)[\sigma] = \Pi\ (A[\sigma])\ (B[\sigma^\uparrow])$

$\mathsf{lam}[] \quad : (\mathsf{lam}\ t)[\sigma] = \mathsf{lam}\ (t[\sigma^\uparrow])$

Sigma

$\Sigma \quad : (A : \mathsf{Ty}_i\ \Gamma) \to \mathsf{Ty}_j\ (\Gamma \triangleright A) \to \mathsf{Ty}_{i \sqcup j}\ \Gamma$

$-, - \quad : (u : \mathsf{Tm}\ \Gamma\ A) \to \mathsf{Tm}\ \Gamma\ (B[\langle u \rangle]) \to$
$\mathsf{Tm}\ \Gamma\ (\Sigma\ A\ B)$

$\mathsf{proj}_1 \quad : \mathsf{Tm}\ \Gamma\ (\Sigma\ A\ B) \to \mathsf{Tm}\ \Gamma\ A$

$\mathsf{proj}_2 \quad : (t : \mathsf{Tm}\ \Gamma\ (\Sigma\ A\ B)) \to \mathsf{Tm}\ \Gamma\ (B[\langle \mathsf{proj}_1\ u \rangle])$

$\Sigma\beta_1 \quad : \mathsf{proj}_1\ (u, v) = u$

$\Sigma\beta_2 \quad : \mathsf{proj}_2\ (u, v) = v$

$\Sigma\eta \quad : (\mathsf{proj}_1\ t, \mathsf{proj}_2\ t) = t$

$\Sigma[] \quad : (\Sigma\ A\ B)[\sigma] = \Sigma\ (A[\sigma])\ (B[\sigma^\uparrow])$

$, [] \quad : (u, v)[\sigma] = (u[\sigma], v[\sigma])$

Unit

$\top \quad : \mathsf{Ty}_0\ \Gamma$

$\mathsf{tt} \quad : \mathsf{Tm}\ \Gamma\ \top$

$\top\eta \quad : (t : \mathsf{Tm}\ \Gamma\ \top) = \mathsf{tt}$

$\top[] \quad : \top[\sigma] = \top$

$\mathsf{tt}[] \quad : \mathsf{tt}[\sigma] = \mathsf{tt}$

Empty

$\bot \quad : \mathsf{Ty}_0\ \Gamma$

$\mathsf{exfalso} \quad : \mathsf{Tm}\ \Gamma\ \bot \to \mathsf{Tm}\ \Gamma\ C$

$\bot[] \quad : \bot[\sigma] = \bot$

$\mathsf{exfalso}[] \quad : (\mathsf{exfalso}\ t)[\sigma] = \mathsf{exfalso}\ (t[\sigma])$

Sum

$- + - \quad : \mathsf{Ty}_i\ \Gamma \to \mathsf{Ty}_j\ \Gamma \to \mathsf{Ty}_{i \sqcup j}\ \Gamma$

$\mathsf{inj}_1 \quad : \mathsf{Tm}\ \Gamma\ A \to \mathsf{Tm}\ \Gamma\ (A + B)$

$\mathsf{inj}_2 \quad : \mathsf{Tm}\ \Gamma\ B \to \mathsf{Tm}\ \Gamma\ (A + B)$

$\mathsf{case} \quad : (P : \mathsf{Ty}_i\ (\Gamma \triangleright A + B)) \to$
$\mathsf{Tm}\ (\Gamma \triangleright A)\ (P[\mathsf{wk}, \mathsf{inj}_1\ \mathsf{vz}]) \to$
$\mathsf{Tm}\ (\Gamma \triangleright B)\ (P[\mathsf{wk}, \mathsf{inj}_2\ \mathsf{vz}]) \to$
$(t : \mathsf{Tm}\ \Gamma\ (A + B)) \to \mathsf{Tm}\ \Gamma\ (P[\langle t \rangle])$

$+\beta_1 \quad : \mathsf{case}\ P\ u\ v\ (\mathsf{inj}_1\ t) = u[\langle t \rangle]$

$+\beta_2 \quad : \mathsf{case}\ P\ u\ v\ (\mathsf{inj}_2\ t) = v[\langle t \rangle]$

$+[] \quad : (A + B)[\sigma] = A[\sigma] + B[\sigma]$

$\mathsf{inj}_1[] \quad : (\mathsf{inj}_1\ t)[\sigma] = \mathsf{inj}_1\ (t[\sigma])$

$\mathsf{inj}_2[] \quad : (\mathsf{inj}_2\ t)[\sigma] = \mathsf{inj}_2\ (t[\sigma])$

$\mathsf{case}[] \quad : (\mathsf{case}\ P\ u\ v\ t)[\sigma] =$
$\mathsf{case}\ (P[\sigma^\uparrow])\ (u[\sigma^\uparrow])\ (v[\sigma^\uparrow])\ (t[\sigma])$

Coquand universes

$\mathsf{U}_{\_} \quad : (i : \mathbb{N}) \to \mathsf{Ty}_{i+1}\ \Gamma$

$\mathsf{El} \quad : \mathsf{Tm}\ \Gamma\ \mathsf{U}_i \to \mathsf{Ty}_i\ \Gamma$

$\mathsf{c} \quad : \mathsf{Ty}_i\ \Gamma \to \mathsf{Tm}\ \Gamma\ \mathsf{U}_i$

$\mathsf{U}\beta \quad : \mathsf{El}\ (\mathsf{c}\ A) = A$

$\mathsf{U}\eta \quad : \mathsf{c}\ (\mathsf{El}\ a) = a$

$\mathsf{U}[] \quad : \mathsf{U}_i[\sigma] = \mathsf{U}_i$

$\mathsf{El}[] \quad : (\mathsf{El}\ a)[\sigma] = \mathsf{El}\ (a[\sigma])$

Booleans

$\mathsf{Bool} \quad : \mathsf{Ty}_0$

$\mathsf{true} \quad : \mathsf{Tm}\ \Gamma\ \mathsf{Bool}$

$\mathsf{false} \quad : \mathsf{Tm}\ \Gamma\ \mathsf{Bool}$

$\mathsf{if} \quad : (P : \mathsf{Ty}_i\ (\Gamma \triangleright \mathsf{Bool})) \to \mathsf{Tm}\ \Gamma\ (P[\langle \mathsf{true} \rangle]) \to$
$\mathsf{Tm}\ \Gamma\ (P[\langle \mathsf{false} \rangle]) \to (t : \mathsf{Tm}\ \Gamma\ \mathsf{Bool}) \to$
$\mathsf{Tm}\ \Gamma\ (P[\langle t \rangle])$

$\mathsf{Bool}\beta_{\mathsf{true}} \quad : \mathsf{if}\ P\ u\ v\ \mathsf{true} = u$

$\mathsf{Bool}\beta_{\mathsf{false}} \quad : \mathsf{if}\ P\ u\ v\ \mathsf{false} = v$

$\mathsf{Bool}[] \quad : \mathsf{Bool}[\sigma] = \mathsf{Bool}$

$\mathsf{true}[] \quad : \mathsf{true}[\sigma] = \mathsf{true}$

$\mathsf{false}[] \quad : \mathsf{false}[\sigma] = \mathsf{false}$

$\mathsf{if}[] \quad : (\mathsf{if}\ P\ u\ v\ t)[\sigma] = \mathsf{if}\ (P[\sigma^\uparrow])\ (u[\sigma])\ (v[\sigma])\ (t[\sigma])$

Natural numbers

$\mathsf{Nat} \quad : \mathsf{Ty}_0$

# Típuselmélet, mint algebrai struktúra részlete

$$
\begin{aligned}
&\mathsf{Con} && : \mathsf{Type} \\
&\mathsf{Sub} && : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type} \\
&\mathsf{Ty} && : \mathsf{Type} \\
&\mathsf{Tm} && : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type} \\
&\mathsf{id} && : \mathsf{Sub}\,\Gamma\,\Gamma \\
&-[-] && : \mathsf{Tm}\,\Delta\,A \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Tm}\,\Gamma\,A \\
&[\mathsf{id}] && : t[\mathsf{id}] = t \\
&\mathsf{zero} && : \mathsf{Tm}\,\Gamma\,\mathsf{Nat} \\
&\mathsf{zero}[] && : \mathsf{zero}[\sigma] = \mathsf{zero}
\end{aligned}
$$

# Típuselmélet, mint algebrai struktúra részlete

$$
\begin{aligned}
&\mathsf{Con} && : \mathsf{Type}\\
&\mathsf{Sub} && : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type}\\
&\mathsf{Ty} && : \mathsf{Type}\\
&\mathsf{Tm} && : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type}\\
&\mathsf{id} && : \mathsf{Sub}\,\Gamma\,\Gamma\\
&-[-] && : \mathsf{Tm}\,\Delta\,A \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Tm}\,\Gamma\,A\\
&[\mathsf{id}] && : t[\mathsf{id}] = t\\
&\mathsf{zero} && : \mathsf{Tm}\,\Gamma\,\mathsf{Nat}\\
&\mathsf{zero}[] && : \mathsf{zero}[\sigma] = \mathsf{zero}
\end{aligned}
$$

A szabad algebrában két különböző $\mathsf{zero}[\mathsf{id}] = \mathsf{zero}$ egyenlőség van. Emiatt $\mathsf{Tm}\,\Gamma\,\mathsf{Nat}$ nem halmaz, ezért nem eldönthető az egyenlőség, tehát ez nem egy programozási nyelv!

# Javítási kísérlet 1.

Adjuk hozzá a hiányzó egyenlőséget!

$$
\begin{array}{ll}
\mathsf{Con} & : \mathsf{Type} \\
\mathsf{Sub} & : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type} \\
\mathsf{Ty} & : \mathsf{Type} \\
\mathsf{Tm} & : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type} \\
\mathsf{id} & : \mathsf{Sub}\,\Gamma\,\Gamma \\
-[-] & : \mathsf{Tm}\,\Delta\,A \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Tm}\,\Gamma\,A \\
[\mathsf{id}] & : t[\mathsf{id}] = t \\
\mathsf{zero} & : \mathsf{Tm}\,\Gamma\,\mathsf{Nat} \\
\mathsf{zero}[] & : \mathsf{zero}[\sigma] = \mathsf{zero} \\
\mathsf{zero}[\mathsf{id}] & : \mathsf{zero}[] = [\mathsf{id}]
\end{array}
$$

# Javítási kísérlet 1.

Adjuk hozzá a hiányzó egyenlőséget!

$$
\begin{array}{ll}
\mathsf{Con} & : \mathsf{Type} \\
\mathsf{Sub} & : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type} \\
\mathsf{Ty} & : \mathsf{Type} \\
\mathsf{Tm} & : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type} \\
\mathsf{id} & : \mathsf{Sub}\,\Gamma\,\Gamma \\
-[-] & : \mathsf{Tm}\,\Delta\,A \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Tm}\,\Gamma\,A \\
[\mathsf{id}] & : t[\mathsf{id}] = t \\
\mathsf{zero} & : \mathsf{Tm}\,\Gamma\,\mathsf{Nat} \\
\mathsf{zero}[] & : \mathsf{zero}[\sigma] = \mathsf{zero} \\
\mathsf{zero}[\mathsf{id}] & : \mathsf{zero}[] = [\mathsf{id}]
\end{array}
$$

Végtelen sok ilyen újabb egyenlőségre lenne szükségünk, és nem ismert, hogy ez leírható-e egyáltalán típuselméletben.

# Javítási kísérlet 2.

Tegyük halmazzá erővel!

$$\begin{aligned}\mathsf{Tm} &: \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type}\\ \mathsf{setTm} &: (u\,v : \mathsf{Tm}\,\Gamma\,A)(p\,q : u = v) \to p = q\end{aligned}$$

# Javítási kísérlet 2.

Tegyük halmazzá erővel!

$$\begin{aligned}\mathsf{Tm} &: \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type}\\ \mathsf{setTm} &: (u\,v : \mathsf{Tm}\,\Gamma\,A)(p\,q : u = v) \to p = q\end{aligned}$$

Probléma: így nem adható meg a metacirkuláris interpreter (értelmező) a szabad algebrára. Hiszen az értelmező esetén $\mathsf{Ty} = \mathsf{Type}$ és $\mathsf{Tm}\,\Gamma\,A = \Gamma \to A$, tehát a termek nem alkotnak halmazt.

# (Részleges) megoldásunk

A termeket erővel halmazzá tesszük (és a típusokat groupoiddá).

$$\begin{aligned}\mathsf{Tm} &: \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type}\\ \mathsf{setTm} &: (u\,v : \mathsf{Tm}\,\Gamma\,A)(p\,q : u = v) \to p = q\end{aligned}$$

Az interpreter megadható úgy, hogy a típusokat halmazokkal értelmezzük: $\mathsf{Ty} = \mathsf{Set}$ és $\mathsf{Tm}\,\Gamma\,A = \Gamma \to A$, és mivel $A$ halmaz, ezért $\Gamma \to A$ is halmaz.

# (Részleges) megoldásunk

A termeket erővel halmazzá tesszük (és a típusokat groupoiddá).

$$\begin{aligned}\mathsf{Tm} &: \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type}\\ \mathsf{setTm} &: (u\,v : \mathsf{Tm}\,\Gamma\,A)(p\,q : u = v) \to p = q\end{aligned}$$

Az interpreter megadható úgy, hogy a típusokat halmazokkal értelmezzük: $\mathsf{Ty} = \mathsf{Set}$ és $\mathsf{Tm}\,\Gamma\,A = \Gamma \to A$, és mivel $A$ halmaz, ezért $\Gamma \to A$ is halmaz.

Agdában formalizáltuk a megoldásunkat, és majdnem készen vagyunk annak bizonyításával, hogy az így kapott szintaxis ekvivalens a hagyományos szintaxissal. Ez a legegyszerűbb változata annak, amikor a típuselméletet beágyazzuk a homotópia-típuselméletbe.

# Utóbbi év konkrét eredményei

- Meghívott előadás:
  - Quotient inductive-inductive types and higher friends. Homotopy Type Theory Electronic Seminar Talks (HoTTEST), online, 22 October 2020
- Elfogadott cikkek:
  - Constructing a universe for the setoid model. Thorsten Altenkirch, Simon Boulier, K.A., Christian Sattler, Filippo Sestini. 24th International Conference on Foundations of Software Science and Computation Structures, 27 March-1 April 2021
  - For Finitary Induction-Induction, Induction Is Enough. K.A., Kovács András, Ambroise Lafont. 25th International Conference on Types for Proofs and Programs (TYPES 2019) post-proceedings.
- Beküldött/beküldendő cikkek:
  - A calculus of single substitutions for simple type theory. K.A., Luksa Norbert. MACS 2020
  - Induction principles for type theories, internally to presheaf categories. Rafaël Bocquet, K.A. Christian Sattler. Computer Science Logic 2021.
- Kapcsolódás: Horpácsi Dániellel konzultációk, ők is típuselméletet használnak.
- Szeptemberben kezdődő típuselmélet COST Action-ben magyar részvétel