

Bizonyítás és programozás

Kaposi Ambrus

University of Nottingham
Functional Programming Lab

Hackerspace Budapest
2015. január 6.

Egy egyszerű logika

Állítások:

- ▶ igaz: \top
- ▶ hamis: \perp
- ▶ ha A és B állítás, akkor $A \wedge B$ is az
- ▶ hasonlóképp $A \rightarrow B$
- ▶ alapállítások: sáros a csizmám, esett az eső, vizes a föld

Logikai szabályok:

$$\frac{}{\top} \quad \frac{}{\perp} \quad \frac{A \quad B}{A \wedge B} \quad \frac{A}{A \wedge B} \quad \frac{B}{A \wedge B} \quad \frac{A \rightarrow B \quad A}{B}$$

$$\frac{\begin{array}{c} A \\ \vdots \\ B \end{array}}{A \rightarrow B}$$

Axiómák:

$$\frac{}{\text{vizes a föld} \rightarrow \text{esett az eső}} \quad \frac{\text{sáros a csizmám}}{\text{vizes a föld}}$$

Halmazelmélet

Állítások:

- ▶ $\top, \perp, A \wedge B, A \vee B, A \rightarrow B$
- ▶ $\forall x.A, \exists x.A$ (változók), pl. $\forall x.x + 3 = 3 + x$
- ▶ alapállítások: $a = b, a \in b$ (a és b kifejezések)

Kifejezések:

- ▶ változók (x, y, \dots)
- ▶ \emptyset

Logikai szabályok mint az előbb, plusz:

$$\frac{A}{\forall x.A} \quad (x \text{ szabad } A\text{-ban}) \quad \frac{\forall x.A}{A[x \mapsto a]} \quad a \quad \frac{A[x \mapsto a]}{\exists x.A} \quad \frac{\exists x.A}{A[x \mapsto a]}$$

Axiómák:

$$\frac{a = b \quad b = c}{a = c} \quad \dots \quad \forall x.\forall y.\forall z.(z \in x \leftrightarrow z \in y) \rightarrow x = y$$

$$\forall x.\forall y.\exists z.\forall q.q \in z \leftrightarrow q = x \vee q = y \quad (\text{rövidítés: } z \equiv \{x, y\}) \quad \dots$$

ZFC

1. meghatározottság
2. pár
3. részhalmaz
4. unió
5. hatványhalmaz
6. végtelenség halmaz
7. helyettesítés
8. regularitás
9. kiválasztási

A matematika felépítése

- ▶ Ha x, y két halmaz, a belőlük álló rendezett pár:
 $(x, y) \equiv \{\{x\}, \{x, y\}\}$.
- ▶ Egy reláció rendezett párok halmaza.
- ▶ Egy f relációt függvénynek hívunk, ha
 $(x, y) \in f \wedge (x, y') \in f \rightarrow y = y'$.
- ▶ \mathbb{N} definíciója:

$$0 \equiv \emptyset$$

$$\text{succ}(x) \equiv x \cup \{x\} \quad (\text{unió axióma})$$

$$\mathbb{N} = \{\emptyset\} \cup \{x \cup \{x\} \mid x \in \mathbb{N}\} \quad (\text{végtelenségi axióma})$$

- ▶ A $+$ függvény az alábbi halmaz:

$$+ \equiv \{((0, x), x) \mid x \in \mathbb{N}\} \cup \{((\text{succ}(x), y), \text{succ}(x + y)) \mid x, y \in \mathbb{N}\}$$

- ▶ A fenti definíciókkal pl. levezethető, hogy $\forall x. x + 3 = 3 + x$.

Tarski (lengyel matematikus, 1901-1983) ötlete

Logikai összekötők jelentése:

- ▶ \top jelentése t
- ▶ \perp jelentése f
- ▶ A logikai összekötők jelentése igazságtáblázattal adható meg:

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$
t	t	t	t	t
t	f	f	t	f
f	t	f	t	t
f	f	f	f	t

- ▶ $\forall x.A$ jelentése akkor és csak akkor t, ha minden a halmazra $A[x \mapsto a]$ jelentése t.
- ▶ $\exists x.A$ jelentése akkor és csak akkor t, ha létezik olyan a halmaz, melyre $A[x \mapsto a]$ jelentése t.

Heyting (holland matematikus, 1898 – 1980) ötlete

Logikai összekötők jelentése:

$A \wedge B$ A egy bizonyítása és B egy bizonyítása

$A \vee B$ egy (i, p) pár, ahol ha $i = 0$, akkor p A egy bizonyítása,
ha $i = 1$, akkor p B egy bizonyítása

$A \rightarrow B$ egy függvény, mely A egy bizonyítását átalakítja B egy bizonyításává

$\forall x.A$ egy függvény, mely egy a halmazt $A[x \mapsto a]$ bizonyításává alakítja

$\exists x.A$ egy (a, p) pár, ahol a halmaz, p pedig $A[x \mapsto a]$ egy bizonyítása

\perp (valami, aminek nincs bizonyítása)

Programozás

Programok típusai:

$\text{sort} : \text{List Int} \rightarrow \text{List Int}$ $+$: $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

Hogyan írunk adott típusú programot?

$A \times B$ két program, egy A típusú és egy B típusú

$A \vee B$ vagy egy A típusú, vagy egy B típusú program

$A \rightarrow B$ $\lambda x.t$ ahol $x : A$, $t : B$, és t felhasználhatja x -et

$\Pi x : A. B$ $\lambda x.t$ ahol $x : A$, $t : B$, ahol t és B felhasználhatja x -et

$\Sigma x : A. B$ egy (a, b) pár, ahol $a : A$, $b : B[x \mapsto a]$

\perp (nincs ilyen típusú program)

Programok végrehajtása:

$$\frac{\lambda x.t : A \rightarrow B \quad u : A}{(\lambda x.t)(u) \equiv t[x \mapsto u] : B}$$

Észrevétel:

bizonyítás = programozás

Típuselmélet

Észrevétel:

bizonyítás = programozás

Egy programozási nyelv, mely erre az észrevételre épül.

Használható halmazelmélet helyett a matematika felépítésére. Előnyei:

- ▶ A bizonyítások számítógéppel ellenőrizhetők (típusellenőrzés).
- ▶ A bizonyítások végrehajthatók,
 - ▶ pl. ha bizonyítottam, hogy bármely két számnak van legnagyobb közös osztója, akkor ingyen kapok egy programot is, mely ezt kiszámolja.
- ▶ A matematikusoknak van intuíciója a típusokról, pl. nem érdekes nekik, hogy $3 \in 5$ igaz vagy sem. A típuselmélet explicitté teszi ezt az intuíciót.
- ▶ Egyszerűség: a típuselméletben \rightarrow , a függvény típus, és \forall megegyeznek. Nincs külön logikai és nem-logikai rész.

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:

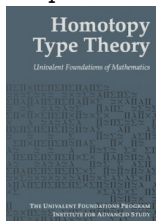
$$\text{zero} + n \equiv n$$
$$\text{suc}(m) + n \equiv \text{suc}(m + n)$$

- ▶ rekurzívan definiáljuk az alábbi függvényt:

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

Következő lépések

- ▶ Magyarul: honlapomon egy cikk a homotópia-típuselméletről
- ▶ Könyvek:
 - ▶ logika és programozás kapcsolata: Girard, Lafont: Proofs and types (fent van a neten)
 - ▶ típuselmélet: Homotopy Type Theory könyv (letölthető)
<http://homotopytypetheory.org>



- ▶ logika: Girard: The blind spot
- ▶ Martin-Löf típuselméletre épülő programozási nyelvek:
 - ▶ Coq: legrégebbi, inkább tételbizonyító rendszerként használják
 - ▶ Agda: legfejlettebb
 - ▶ Idris: gyakorlati programozásra