Bizonyítás és programozás

Kaposi Ambrus

University of Nottingham
Functional Programming Lab

Hackerspace Budapest
2015. január 6.

Bizonyítás, érvelés

Példa:

$$\dfrac{\text{ha vizes a föld, esett az eső} \qquad \dfrac{\text{sáros a csizmám}}{\text{vizes a föld}}}{\text{ha sáros a csizmám, esett az eső}}$$

Melyek a megengedett lépések?

Rossz példa:

$$\dfrac{\dfrac{\text{a halak tudnak úszni} \quad \text{a delfinek halak}}{\text{a delfinek tudnak úszni}} \qquad \text{a delfinek nem halak}}{\text{a delfinek nem tudnak úszni}}$$

Egy egyszerű logika

Állítások:

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

Logikai szabályok:

$$\frac{}{\top} \qquad \frac{\bot}{A} \qquad \frac{A \quad B}{A \wedge B} \qquad \frac{A}{A \wedge B} \qquad \frac{B}{A \wedge B} \qquad \frac{A \to B \quad A}{B} \qquad \frac{\begin{array}{c} A \\ \vdots \\ B \end{array}}{A \to B}$$

Axiómák:

$$\frac{}{\text{vizes a föld} \to \text{esett az eső}} \qquad \frac{\text{sáros a csizmám}}{\text{vizes a föld}}$$

Halmazelmélet

Állítások:

- $\top$, $\bot$, $A \wedge B$, $A \vee B$, $A \to B$
- $\forall x.A$, $\exists x.A$ (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:

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

Axiómák:

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

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

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:

$$
\begin{aligned}
&0 \equiv \emptyset \\
&\mathsf{suc}(x) \equiv x \cup \{x\} && \text{(unió axióma)} \\
&\mathbb{N} = \{\emptyset\} \cup \{x \cup \{x\} \mid x \in \mathbb{N}\} && \text{(végtelenségi axióma)}
\end{aligned}
$$

- A $+$ függvény az alábbi halmaz:

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

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

Tarski (lengyel matematikus, 1901-1983) ötlete

Logikai összekötők jelentése:

- $\top$ jelentése t
- $\bot$ jelentése f
- A logikai összekötők jelentése igazságtáblázattal adható meg:

A	B	$A \wedge B$	$A \vee B$	$A \to B$
t	t	t	t	t
t	f	f	t	t
f	t	f	t	f
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
$A \to B$	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
$\bot$	(valami, aminek nincs bizonyítása)

Programozás

Programok típusai:

$$\mathsf{sort} : \mathsf{List}\,\mathsf{Int} \to \mathsf{List}\,\mathsf{Int} \qquad + : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

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
$A \to B$	$\lambda x.t$ 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 $a : A$, $b : B[x \mapsto a]$
$\bot$	(nincs ilyen típusú program)

Programok végrehajtása:

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

Észrevétel:

$$\text{bizonyítás} = \text{programozás}$$

Típuselmélet

Észrevétel:

$$\text{bizonyítás} = \text{programozás}$$

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 progamot 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:

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

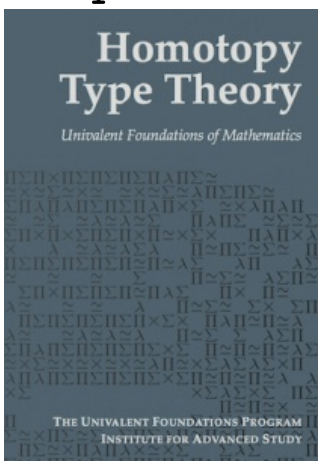
$$\mathsf{zero} + n :\equiv n$$
$$\mathsf{suc}(m) + n :\equiv \mathsf{suc}(m + n)$$

- rekurzívan definiáljuk az alábbi függvényt:

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

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

Köszönöm szépen a figyelmet!