λ-kalkulus I.

BME TMIT
2022

14/3

Németh Gábor

Lambda-kalkulus I.

- A. Church (1903–1995), 1932-1933
  - matematika formális leírása
  - ellentmondások is leírhatóak ☹
- függvények vizsgálata
- Kleene
  - minden kiszámítható függvény leírható λ-kalkulusban, a λ-definiálható függvények pontosan a kiszámítható függvények
- Turing-tétel
  - λ-definiálhatóság és Turing-kiszámíthatóság ekvivalens

Lambda-kalkulus II.

- minden funkcionális program egy λ-kifejezésnek tekinthető
  - végrehajtás
    - kifejezés értékének a meghatározása

Egyszerű típusnélküli λ-kalkulus

Szintaktika
Szemantika
Normál forma

λ–kifejezések I.

- Ábécé
  - változók (szimbólumok)
  - λ
  - . *(pont)*
  - (
  -)

```
<λ-kifejezés>   ::= <változó>
                   | <λ-absztrakció>
                   | <applikáció>
<λ-absztrakció> ::= (λ<változó>.<λ-kifejezés>)
<applikáció>    ::= (<λ-kifejezés><λ-kifejezés>)
```

λ-kifejezések II.

- λ-kalkulus
  - nincs típus
  - nincs konstans
  - nincs konstanson értelmezett függvény

- ≡
  - szintaktikailag azonos
    - pontosan megegyeznek
    - $E \equiv F$
    - $I \equiv \lambda x.x$
    - $K \equiv \lambda x.(\lambda y.x)$

λ-absztrakció

- E – λ-kifejezés
- x – változó

$\lambda x.E$

változó
formális paraméter

törzs

x változó hatóköre

- jobbasszociatív
  - $\lambda x.(\lambda y.E) \equiv \lambda x.\lambda y.E \equiv \lambda xy.E$

$\lambda x.E$	$E(x)$
$\lambda x.EF$	$(E \circ F)(x) = E(F(x))$

Applikáció

- E, F – λ-kifejezés

$$EF$$

- balasszociatív
  - (EF)G ≡ EFG
- precedenciája nagyobb, mint a λ-absztrakciójénál

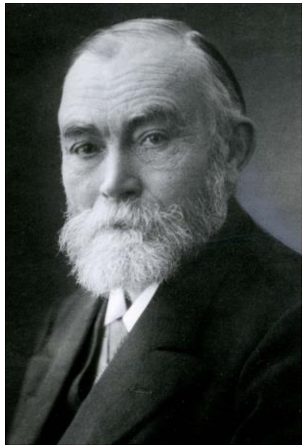
$$\lambda x.(yz) \equiv \lambda x.yz$$

$$\lambda x((\lambda y.E)F) \equiv \lambda x.(\lambda y.E)F$$

$$\lambda xyz.xyz \equiv \lambda x.(\lambda y.(\lambda z.((xy)z)))$$

Körrizés

- minden függvénynek csak egy változója lehet
- mi a helyzet a többváltozós matematikai függvényekkel?
  - magasabb rendű függvények
  - applikációk sorozata
  - ezt nevezzük **körrizés**nek
  - G. Frege, M. Schönfinkel, H. Curry



$$\mathrm{add}(x, y) \longrightarrow \mathrm{add}_x(y)$$

$$\downarrow$$

$$\lambda y.(\mathrm{add}x)y$$

$$\downarrow$$

$$\lambda x.(\lambda y.(\mathrm{add}x)y)$$

$$\downarrow$$

$$\lambda x \lambda y.\mathrm{add}xy$$

Schönfinkeling

Szabad és kötött változók I.

- a törzsben levő változó melyik absztrakcióhoz tartozik
  - melyik absztrakció köti
  - “deklaráció láthatóságához hasonlóan”

$$\lambda x.(\lambda x.x)x$$

$$\lambda xx.x \equiv \lambda x.(\lambda x.x)$$

Szabad és kötött változók II.

- szabad változók (FV)
- x változó szabad az x kifejezésben
- x szabad $\lambda y.E$, ha $x \not\equiv y$ és x szabad E-ben
- x szabad EF-ben, ha x szabad E-ben vagy F-ben

- kötött változók (BV)
- x kötött $\lambda y.E$, ha $x \equiv y$ és x szabad E-ben
- x kötött $\lambda y.E$-ben, ha x kötött E-ben
- x kötött EF-ben, ha x kötött E-ben vagy F-ben

FV(x) = {x}, ahol x változó
$FV(\lambda x.E) = FV(E) \setminus \{x\}$
$FV(EF) = FV(E) \cup FV(F)$

Szabad és kötött változók III.

- Hol kötött?

$$\lambda x.(\underline{\underline{\lambda x.x}})x$$

$$\underline{\lambda x.(\lambda x.x)x}$$

Szabad és kötött változók IV.

- Melyek a szabad változók?

$$(\lambda y.(xz)\ \lambda y.y)\ \lambda y.(zy)$$

$$(\lambda x.((xx)\ \lambda z.z)(\lambda z.z\ \lambda y.y))$$

$$((\lambda v.v\ \lambda z.x)\ \lambda xy.y)$$

Kifejezések lezárása

- ha a λ-kifejezésben nincs szabad változó, akkor a λ-kifejezést zártnak nevezzük
  - a zárt λ-kifejezéseket kombinátoroknak is nevezik
  - $I \equiv \lambda x.x$
  - $K \equiv \lambda xy.x$
  - $S \equiv \lambda xyz.xz(yz)$
- ha $\{x_1, x_2, \ldots, x_n\} = FV(E)$, akkor a $\lambda x_1 x_2 \ldots x_n.E$ kifejezést az E egy lezárásának nevezzük

Helyettesítés

- ha az E λ-kifejezésben a szabad x változót mindenütt az F λ-kifejezéssel helyettesítjük, akkor az így kapott λ-kifejezést E[x:=F]-fel jelöljük
  - szabad változó nem válhat kötötté

$$x[y:=G] \equiv \begin{cases} G & \text{ha } x \equiv y \\ x & \text{egyébként} \end{cases}$$

$$(EF)[y:=G] \equiv (E[y:=G])(F[y:=G])$$

$$(\lambda x.E)[y:=G] \equiv \begin{cases} \lambda x.E & \text{ha } x \equiv y \\ \lambda x.E[y:=G] & \text{ha } x \not\equiv y \text{ és } x \notin FV(G) \\ \lambda x.E & \text{egyébként} \end{cases}$$

Egyszerű típusnélküli λ-kalkulus

Szintaktika
Szemantika
Normál forma

β-konverzió I.

- funkcionális program → λ-kifejezés
- funkcionális program futtatása → λ-kifejezés egyszerűbb alakra hozása
  - konverziós szabályok
    - reflexív, szimmetrikus, tranzitív

- β-redukció ($\rightarrow_\beta$)
- ha az E[x:=F]-ben F szabad változói nem válnak az E kötött változóivá, akkor $(\lambda x.E)F \rightarrow_\beta E[x:=F]$

- β-absztrakció ($\leftarrow_\beta$)
- ha egy λ-kifejezésből olyan applikációt írunk fel, amelynek az első tagja λ-absztrakció

- β-konverzió ($\leftrightarrow_\beta$)
- ha $E \leftrightarrow_\beta F$, akkor tetszőleges G λ-kifejezésre:
  - $GE \leftrightarrow_\beta GF$
  - $EG \leftrightarrow_\beta FG$
  - $\lambda x.E \leftrightarrow_\beta \lambda x.F$

true ≡ λxy.x
false ≡ λxy.y

if ≡ λpqr.pqr

β-konverzió II.

$$
\begin{aligned}
\textbf{if}\ \textbf{true}EF &\equiv (\lambda pqr.pqr)\ \textbf{true}EF \\
&\to_\beta (\lambda qr.\textbf{true}qr)EF \\
&\to_\beta (\lambda r.\textbf{true}Er)F \\
&\to_\beta \textbf{true}EF \\
&\equiv (\lambda xy.x)EF \\
&\to_\beta (\lambda y.E)F \\
&\to_\beta E
\end{aligned}
$$

β-konverzió III.

not E ≡ if E false true

≡ (λpqr.pqr) E false true

$\to_\beta$ …

$\to_\beta$ E false true

not ≡ λx.x false true

not false ≡ (λx.x false true) false

$\to_\beta$ false false true

≡ (λxy.y) false true

$\to_\beta$ (λy.y) true

$\to_\beta$ true

α-konverzió

- a β-redukciót nem szabad végrehajtani, ha a redukálás után a paraméter szabad változója kötötté válik

- β-redukció nem alkalmazható:
  - $(\lambda xy.xy)y \not\twoheadrightarrow_\beta \lambda y.yy$
- $(\lambda xy.xy)y \leftrightarrow_\alpha$
- $(\lambda xz.xz)y \rightarrow_\beta$
- $\lambda z.yz$

- α-konverzió ($\leftrightarrow_\alpha$)
  - α-redukció, α-kontrakció
- ha az E-ben y nem szabad változó, akkor $\lambda x.E \leftrightarrow_\alpha \lambda y.E[x:=y]$
- nem könnyű implementálni
  - változók számmal való helyettesítése
  - de Brujin-számok

Egyenlőség I.

- az E és F λ-kifejezésekre E = F (egymásba konvertálhatóak), ha
  - $E \equiv F$
  - $E \twoheadleftarrow\!\!\twoheadrightarrow F$

- reflexív (E=E)
- szimmetrikus (ha E=F, akkor F=E)
- tranzitív (ha E=F és F=G, akkor E=G)

Egyenlőség II.

- Leibnitz-szabály
  - ha $E_1=F_1$, E_1 az E λ-kifejezés egy részkifejezése, és F csak abban különbözik E-től, hogy benne az E_1 részkifejezése helyén F_1 szerepel, akkor E=F

- E=F, G tetszőleges λ-kifejezés
  - EG = FG
  - GE = GF
  - λx.E = λx.F

Egyszerű típusnélküli λ-kalkulus axiómái

- Egyszerű típusnélküli λ-kifejezések között olyan E=F egyenlőségeket tartalmaz, amelyek a következő axiómák felhasználásával bizonyíthatóak:

- I. $(\lambda x.E)F = E[x:=F]$
- II. i. $E=E$
- II. ii. $E=F \Rightarrow F=E$
- II. iii. $E=F,\ F=G \Rightarrow E=G$
- II. iv. $E=F \Rightarrow EG = FG$
- II. v. $E=F \Rightarrow GE=GF$
- II. vi. $E=F \Rightarrow \lambda x.E = \lambda x.F$

- β-konverzió
- reflexivitás
- szimmetria
- tranzitivitás
- Leibnitz-szabály következménye
- Leibnitz-szabály következménye
- ξ-szabály

Egyszerű típusnélküli λ-kalkulus

Szintaktika
Szemantika
Normál forma

Normál forma, jelentéssel bíró kifejezés

- funkcionális program → λ-kifejezés
- funkcionális program futtatása → λ-kifejezés egyszerűbb alakra hozása
- ha egy λ-kifejezés nincs redukálható kifejezés (redex), akkor a λ-kifejezés normál formában van

- jelentéssel nem bíró kifejezések
  - $\Omega \equiv (\lambda x.xx)\ (\lambda x.xx)$
  - $Y \equiv (\lambda x.(\lambda y.x(yy))(\lambda y.x(yy))$

- van normál formája
  - jelentéssel bíró (jelentős) λ-kifejezés
- nincs normál formája
  - nincs értelmezve függvényfogalom

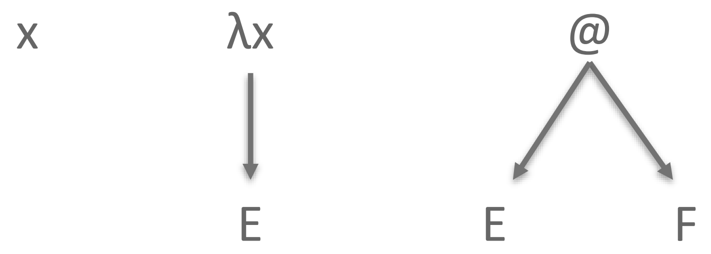
A Church-Rosser-tulajdonság

- funkcionális program → λ-kifejezés
- funkcionális program futtatása → λ-kifejezés egyszerűbb alakra hozása
- ha egy λ-kifejezés nincs redukálható kifejezés (redex), akkor a λ-kifejezés normál formában van

- ha $E_1=E_2$, akkor létezik olyan F, amelyre $E_1 \twoheadrightarrow F$ és $E_2 \twoheadrightarrow F$ (I. Church-Rosser-tétel, rombusz tulajdonság)
- ha $E_1=E_2$ és E_1 normálformában van akkor $E_1 \twoheadrightarrow E_2$
- minden λ-kifejezésnek legfeljebb egy normál formája van
- ha E és F mindegyike normál forma, és $E \not\equiv F$, akkor $E \neq F$

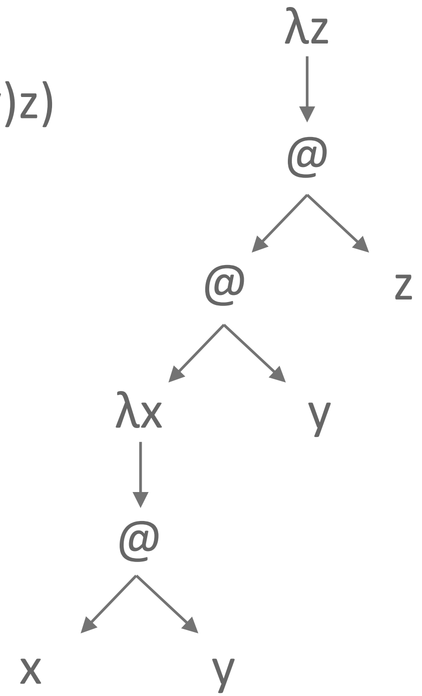
$$\begin{tikzcd}
& E \arrow[dl, two heads] \arrow[dr, two heads] & \\
E_1 \arrow[dr, two heads] & & E_2 \arrow[dl, two heads] \\
& F &
\end{tikzcd}$$

A λ-kifejezés gráfja



λz.(λx.xy)yz

λz.(((λx.xy)y)z)



Redukálási stratégiák I.

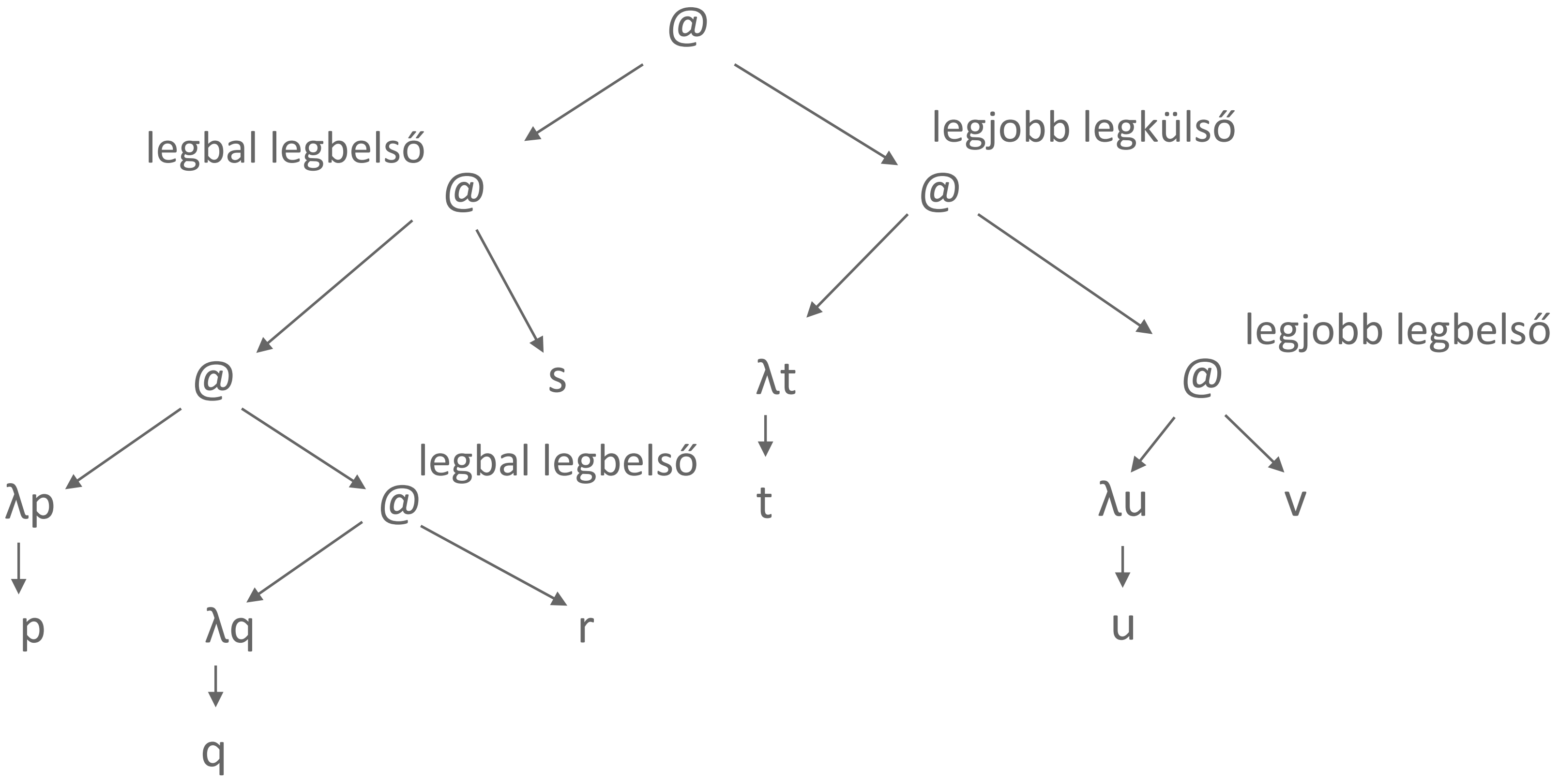
- redukálási sorrend

- legkülső redex
  - nincs más redex belsejében
- legbelső redex
  - belsejében már nincs redex

- E és F két redukálható kifejezés, és E első λ-ja az F első λ-jától balra van, akkor E baloldalibb redex, mint F
- legbaloldalibb redex
  - baloldalibb a kifejezés minden más redexénél
- legjobboldali redex
  - …

Redukálási stratégiák II.

((λp.p)((λq.q)r)) s ((λt.t) ((λu.u)v))



Normál sorrendű redukálási stratégia I.

- legbaloldalibb legkülső
- normalizáló redukálási stratégia
  - normálformát adja
  - II. Church-Rossen-tétel
- név szerinti redukálási stratégia
  - speciális esete
  - absztrakciónál megáll

$(\lambda x.x)(\lambda x.(\lambda z.u)v) \to_\beta$
$(\lambda x.(\lambda z.u)v) \to_\beta$
$\lambda x.u$

Normál sorrendű redukálási stratégia II.

- lusta redukálási stratégia
  - lusta paraméterátadás
  - az argumentum kiértékelését csak akkor végzi el, ha arra már szükség van
  - pl. imperatív nyelvek *if-then-else* struktúrája

Applikatív sorrendű redukálási stratégia

- legbaloldalibb legbelső
- nem feltétlenül találja meg a normál formát

$$(\lambda x.1)(????)$$

- érték szerint λ-kalkulus
  - G. D. Plotkin
  - Lisp , LM nyelvek alapja

Konstansok és függvények

Bevezetés

Logikai konstansok

Rendezett pár

Scott-számjegyek

Konstansok és függvények

- nincsenek konstansok
- nincsenek konstansokon értelmezett függvények

Konstansok és függvények

Bevezetés

Logikai konstansok

Rendezett pár

Scott-számjegyek

Logikai konstansok és műveletek

- Legyen
  - **true** ≡ λxy.x
  - **false** ≡ λxy.y
  - **if** ≡ λpqr.pqr

  - **and** ≡ …
  - **or** ≡ …
  - **not** ≡ **if** E **false true**
    - **not** ≡ λx.x **false true**
    - **not** ≡ λxyz.xzy

Konstansok és függvények

Rendezett pár

- **pair** $\equiv \lambda xyz.zxy$
- **first** $\equiv \lambda x.x$**true** $\equiv \lambda x.x(\lambda yz.y)$
- **second** $\equiv \lambda x.x$**false** $\equiv \lambda x.x(\lambda yz.z)$

- **pair** EF $\twoheadrightarrow \lambda z.zEF$

Konstansok és függvények

Bevezetés

Logikai konstansok

Rendezett pár

Scott-számjegyek

Scott-számjegyek

- [0] $\equiv \lambda xy.x \equiv$ **true**
- **succ** $\equiv \lambda zxy.yz$
- **zero** $\equiv \lambda x.x$**true** $(\lambda y.$**false**$)$
- **pred** $\equiv \lambda x.x[0](\lambda y.y)$

- [i] Scott-számjegyek

Köszönöm a figyelmet!

Folytatjuk…