

# $\lambda$ -kalkulus II.

# Egyszerű típus nélküli $\lambda$ -kalkulus

$\langle \lambda\text{-kifejezés} \rangle ::= \langle \text{változó} \rangle \mid \langle \lambda\text{-absztrakció} \rangle \mid \langle \text{applikáció} \rangle$

$\langle \lambda\text{-absztrakció} \rangle ::= (\lambda \langle \text{változó} \rangle. \langle \lambda\text{-kifejezés} \rangle)$

$\langle \text{applikáció} \rangle ::= (\langle \lambda\text{-kifejezés} \rangle \langle \lambda\text{-kifejezés} \rangle)$

$\lambda x.E$	$E(x)$	
$\lambda x.EF$	$(E \circ F)(x) = E(F(x))$	
$(\lambda x.E)F$	$E(x) \big _{x=F}$	$\beta\text{-redukció } (\rightarrow_\beta)$

“

$\lambda x.(1 + x) \quad x \mapsto 1 + x$

$\lambda x.(1 + x)F \quad x \mapsto 1 + F$

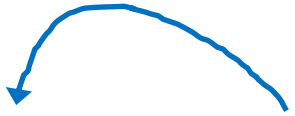
$(\lambda x.(1 + x))F \quad 1 + x \big|_{x=F}$

”

# Konstansok és függvények

- ▶ nincsenek konstansok
- ▶ nincsenek konstansokon értelmezett függvények
  - ▶ egyes kifejezéseket tekinthetjük konstansoknak
- ▶ legyen
  - ▶ **true**  $\equiv \lambda xy.x$
  - ▶ **false**  $\equiv \lambda xy.y$
- ▶ legyen
  - ▶ **pair**  $\equiv \lambda xyz.zxy$
  - ▶ **first**  $\equiv \lambda x.x\text{true} \equiv \lambda x.x(\lambda yz.y)$
  - ▶ **second**  $\equiv \lambda x.x\text{false} \equiv \lambda x.x(\lambda yz.z)$

## $\beta$ -konverzió, normál forma


$$\begin{aligned}\text{first pair } u \ v &\equiv (\lambda x.x(\lambda yz.y)) (\text{pair } u \ v) \equiv (\lambda x.x(\lambda yz.y)) (\underline{\text{pair } u \ v}) \\ &\rightarrow_{\beta} (\text{pair } u \ v)(\lambda yz.y) \\ &\equiv (\lambda xyz.zxy) \underline{u} \ v (\lambda yz.y) \\ &\rightarrow_{\beta} (\lambda yz.zuy) \underline{v} (\lambda yz.y) \\ &\rightarrow_{\beta} (\lambda z.zuv) (\underline{\lambda yz.y}) \\ &\rightarrow_{\beta} (\lambda yz.y) \underline{u} \ v \\ &\rightarrow_{\beta} (\lambda z.u) \underline{v} \\ &\rightarrow_{\beta} u\end{aligned}$$

# Láncolt lista



- ▶ `cons`  $\equiv \lambda xy. \text{pair false (pair x y)}$
- ▶ `nil`  $\equiv \text{pair true true}$
- ▶ `head`  $\equiv \lambda x. \text{first (second x)}$
- ▶ `tail`  $\equiv \lambda x. \text{second (second x)}$

$E_3 \rightarrow E_2 \rightarrow E_1 \rightarrow \text{nil}$

$E' \equiv \text{cons } E_1 \text{ nil}$

$E'' \equiv \text{cons } E_2 E'$

$E''' \equiv \text{cons } E_3 E''$

# λ-kifejezések C++-ban I.

$\lambda x.x$	<code>[](auto x) { return x; }</code>
$\lambda y.y$	<code>[](auto y) { return y; }</code>
$\lambda x.y$	<code>[](auto x) { return y; }</code>
$\lambda x.\lambda y.y$	<code>[](auto x) { return [=](auto y) { return y; } }</code>
$(xy)$	<code>x(y)</code>
$(\lambda x.x)y$	<code>[](auto x) { return x; })(y)</code>
$(\lambda z.x)y$	<code>[](auto z) { return x; })(y)</code>
$\lambda x.(xy)$	<code>[](auto x) { return x(y); }</code>
$(\lambda x.\lambda y.y)z$	<code>[](auto x) { return [=](auto y) { return y; } })(z)</code>
$((\lambda x.\lambda y.y)u)v$	<code>[](auto x) { return [=](auto y) { return y; } })(u)(v)</code>

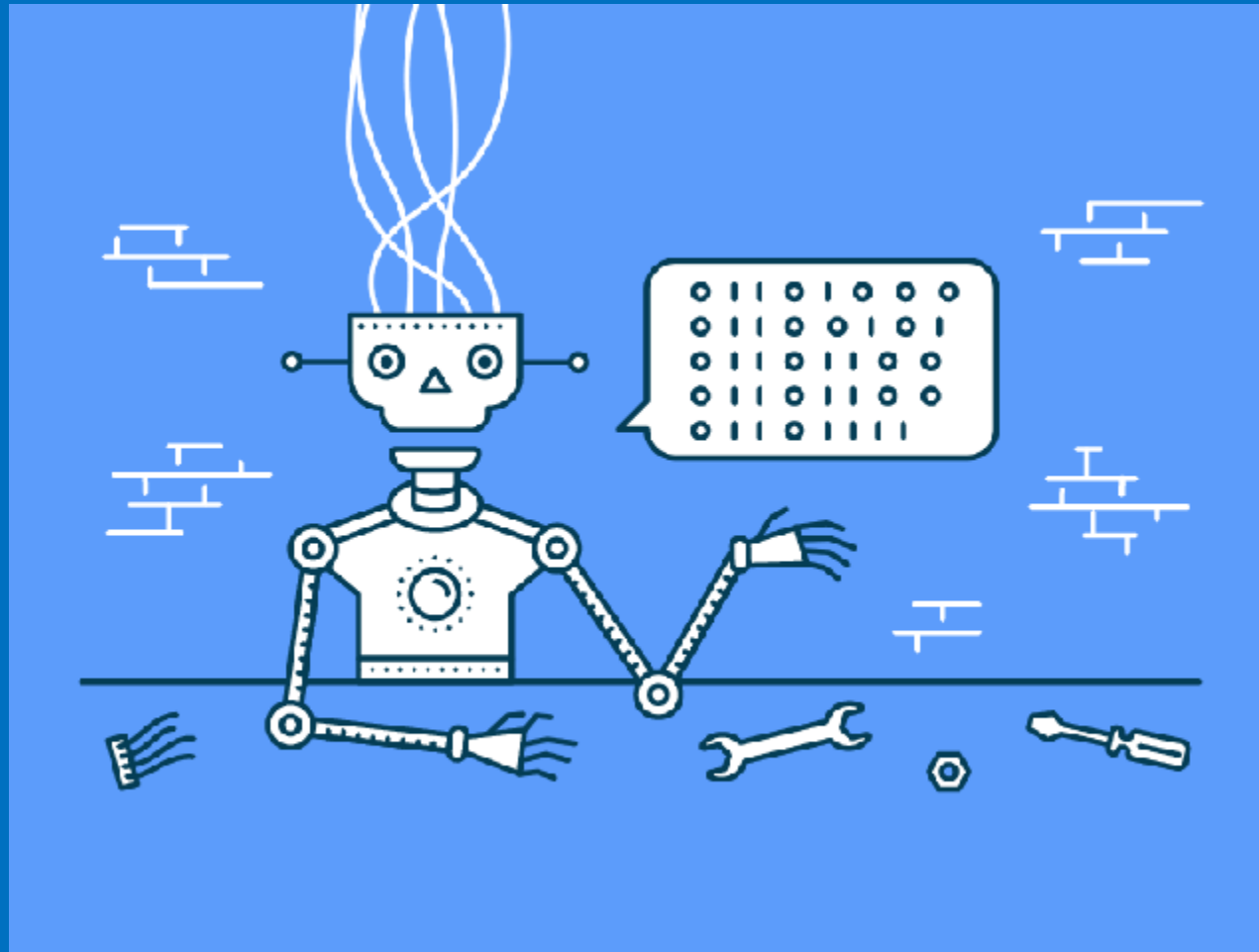
# λ-kifejezések C++-ban II.

```
[](auto x) {  
    return [=](auto y) {  
        return [=](auto x) {  
            return [=](auto z) {  
                return x(y)(z);  
            }  
        }  
    }  
}
```



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

## ex\_0: $\lambda$ -kifejezések C++-ban





# $\delta$ -redukció

- ▶ a  $\lambda$ -kalkulus kifejezései jobban olvashatóak, ha abban beépített konstansok, előre definiált függvények szerepelnek
- ▶ a konstansokon értelmezett függvények:  $\delta$ -függvények
- ▶  $\delta$ -redukció
  - ▶ ha egy függvényapplikációban szereplő függvény egy  $\delta$ -függvény, és az aktuális paraméter konstans, akkor a függvényapplikáció helyettesíthető azzal az értékkel, amelyet a függvény a paraméterrel megadott pontban felvesz

$$+ 1 3 \rightarrow_{\delta} 4$$

$$+ (* 2 3) 4 \rightarrow_{\delta} + 6 4 \rightarrow_{\delta} 10$$

# Rekurzív függvények I.

- ▶  $\text{fac}(n) = n!$ 
  - ▶  $\text{fac}(n) = \text{if}(n = 0) \text{ then } 1 \text{ else } (n * \text{fac}(n-1))$
  - ▶  $\text{fac} \equiv \lambda n. \text{if}(= n 0) 1 (* n (\text{fac}(- n 1)))$ 
    - ▶ de a jobb oldalon nem hivatkozhatunk **fac**-ra!

$H \equiv \lambda f. (\lambda n. \text{if}(= n 0) 1 (* n (f(- n 1))))$

$\text{fac} = H \text{ fac}$

# Fixpont

- ▶ ha az E és F  $\lambda$ -kifejezésekre  $F = EF$ , akkor az F  $\lambda$ -kifejezést E fixpontjának nevezzük

$(\lambda x. -12x)6 = 6$

$H \equiv \lambda f. (\lambda n. \text{if}(= n 0) 1 (* n (f(- n 1))))$

**fac** = H **fac**

**fac** a H fixpontja

## Y fixpontkombinátor

- ▶  $YE = E(YE)$
- ▶  $Y \equiv \lambda x. (\lambda y. x(yy)) (\lambda y. x(yy))$ 
  - ▶ Curry
  - ▶ paradoxkombinátor

# Egyéb fixpontkombinátorok

- ▶  $\Theta$ -fixpontkombinátor
  - ▶  $\Theta = (\lambda xy. y(xxy)) (\lambda xy. y(xxy))$
  - ▶ Turing, 1937, The  $\mu$  function in  $\lambda$ -K conversion (1 oldal)
    - ▶ Kleene, 1936,  $\lambda$  definability and recursiveness
- ▶ Böhm és van der Mey
  - ▶  $G = \lambda xy. y(xy)$ 
    - ▶  $G$  fixpontja fixpontkombinátor
    - ▶  $F = GF$ , akkor  $FE = E(FE)$
- ▶ ...



## Fixpontkombinátor és redukció

- ▶  $Y(\lambda uv.E)F = E[u:=Y(\lambda uv.E)][v:=F]$
- ▶  $\Theta(\lambda uv.E)F \rightarrow E[u:=\Theta(\lambda uv.E)][v:=F]$

## Rekurzív függvények II.

- ▶  $\text{fac} \equiv \lambda n. \text{if}(= n 0) 1 (* n (\text{fac}(- n 1)))$
- ▶  $H \equiv \lambda f. (\lambda n. \text{if}(= n 0) 1 (* n (f(- n 1))))$
- ▶  $\text{fac} = H \text{ fac}$
- ▶  $\text{fac} \equiv YH$

## Rekurzív függvények III.

- ▶  $\text{fac} \equiv YH$
- ▶  $Y(\lambda u v. E) F = E[u := Y(\lambda u v. E)][v := F]$

$$\begin{aligned}
 \text{fac } 1 &\equiv YH1 \equiv Y(\lambda f n. \text{if}(= n 0) 1 (* n (f(- n 1)))) 1 \\
 &\quad \begin{array}{l} \text{H} \\ \text{f} := YH \\ \text{n} := 1 \end{array} \\
 &= \text{if}(= 1 0) 1 (* 1 ((YH)(- 1 1))) \\
 &\rightarrow_{\delta} * 1 ((YH)(- 1 1)) \\
 &\rightarrow_{\delta} * 1 (YH 0)
 \end{aligned}$$



# Elsőrendű típusos $\lambda$ - kalkulus



# Típusos és nem típusos nyelvek

- ▶ nem típusos nyelvek
  - ▶ assembly
  - ▶ nincs megkötés egy változónak adható értékeket
  - ▶ a műveletek nem megfelelő argumentumokra is meghívhatóak → nem meghatározott viselkedés
- ▶ típusos nyelvek
  - ▶ fordítási idejű típushibák
  - ▶ egy típusrendszer követi nyomon a típusokat

# Típusok célja

- ▶ futási idejű hibák megakadályozása
- ▶ tradicionálisan elkapott hibák
  - ▶ a végrehajtás azonnal megáll
  - ▶ gyakran a hardver kényszeríti ki
    - ▶ egészek nullával való osztás
    - ▶ nullptr feloldása
- ▶ nem elkapott hibák
  - ▶ olvasás egy tömb utolsó eleme után

jól meghatározott viselkedés

nem specifikált viselkedés

# Miért...

- ▶ típusos nyelv
  - ▶ teljesítmény
    - ▶ a dinamikus ellenőrzés időigényes lehet
  - ▶ hibák korai megtalálása
  - ▶ típusok használata a tervezés során
  - ▶ egyszerűbb statikusan analizálni
- ▶ nem típusos nyelv
  - ▶ a statikus ellenőrzés megkötés a programozónak
    - ▶ érvényes programok elutasításra kerülhetnek
  - ▶ dinamikus memóriamenedzsment

	típusos		nem típusos
	statikus	dinamikus	
biztonságos	Java, ML, Haskell, Ada, C#	Lisp, PHP, Ruby, Perl, Python	$\lambda$ -kalkulus
nem biztonságos	C, C++, Pascal	?	Assembly

# Elsőrendű Church-típusos $\lambda$ -kalkulus

- ▶ F1 típusrendszer
- ▶  $\lambda x:A.E$ 
  - ▶ kötött változó típust kap
  - ▶ függvénytípus:  $A \rightarrow B$ 
    - ▶ típuskonstruktor
  - ▶ alaptípushalmaz
  - ▶  $\{\text{Bool}, \text{Nat}\}$

$$(EF):A \quad \equiv \quad EF:A$$

$$(\lambda x:A.E):B \quad \equiv \quad \lambda x:A.E:B$$

$\langle \text{típus} \rangle \quad ::= \langle \text{alaptípus} \rangle \mid \langle \text{típus} \rangle \rightarrow \langle \text{típus} \rangle$

$\langle \lambda\text{-kifejezés} \rangle \quad ::= \langle \text{változó} \rangle \mid \lambda \langle \text{változó} \rangle : \langle \text{típus} \rangle. \langle \lambda\text{-kifejezés} \rangle \mid \langle \lambda\text{-kifejezés} \rangle \langle \lambda\text{-kifejezés} \rangle$

# Típus szempontjából hibás kifejezés

- ▶  $\lambda x:A.xy$ 
  - ▶ szintaktikusan helyes
  - ▶ DE!
    - ▶ a kifejezés törzséből:  $x:A \rightarrow B$



- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>▶ <b>jól formált <math>\lambda</math>-kifejezés</b><ul style="list-style-type: none"><li>▶ szintaktikusan helyes</li><li>▶ megfelel a <math>\lambda</math>-kifejezések szintaktikájának leírásában megadott szintaktikai szabályoknak</li></ul></li></ul> | <ul style="list-style-type: none"><li>▶ <b>jól formált típus</b><ul style="list-style-type: none"><li>▶ szintaktikusan helyes</li><li>▶ megfelel a típus szintaktikájában leírásában megadott szintaktikai szabályoknak</li></ul></li></ul> |
|---|---|

Nem garantálja, hogy a kifejezés típus szempontjából hibátlan!





# F1 következtetések

- ▶  $\Gamma$  típuskörnyezet
  - ▶ változók típusa
  - ▶  $\emptyset \mid \langle \text{típuskörnyezet} \rangle, \langle \text{változó} \rangle : \langle \text{típus} \rangle$
  - ▶ következtetés
    - ▶  $\Gamma \vdash I$ 
      - $I$  az  $\Gamma$ -ből adódó állítás
    - ▶ jól formált
      - $\Gamma \vdash \text{wf}$
      - $\emptyset \vdash \text{wf}$
  - ▶ **érvényes** következtetés
    - ▶ ha létezik hozzá típuslevezetés
  - ▶ **jól típusozott** kifejezés
    - ▶ ha  $\Gamma \vdash E:A$  érvényes, akkor  $E$  jól típusozott
- ▶ következtetések
  - ▶  $\Gamma \vdash \text{wf}$ 
    - ▶  $\Gamma$  jól formált környezet
  - ▶  $\Gamma \vdash A$ 
    - ▶  $\Gamma$ -ben az  $A$  jól formált típus
  - ▶  $\Gamma \vdash E:A$ 
    - ▶  $\Gamma$ -ben az  $E$  kifejezés típusa  $A$

## Az F1 szabályai I.

$$\frac{}{\emptyset \vdash \text{wf}} \text{ [ENV } \emptyset]$$

$$\frac{\Gamma \vdash A, x \notin \text{dom}(\Gamma)}{\Gamma, x:A \vdash \text{wf}} \text{ [ENV } x]$$

$$\frac{\Gamma \vdash \text{wf}, A \in \{\text{alaptípus}\}}{\Gamma \vdash A} \text{ [TYPE CONST]}$$

$$\frac{\Gamma \vdash A, \Gamma \vdash B}{\Gamma \vdash A \rightarrow B} \text{ [TYPE ARROW]}$$

## Az F1 szabályai II.

$$\frac{\Gamma', x:A, F'' \vdash \text{wf}}{\Gamma', x:A, F'' \vdash x:A} \text{ [VAL X]}$$

$$\frac{\Gamma, x:A \vdash E:B}{\Gamma \vdash \lambda x:A. E:A \rightarrow B} \text{ [VAL FUN]}$$

$$\frac{\Gamma \vdash E:A \rightarrow B \quad \Gamma \vdash F:A}{\Gamma \vdash EF:B} \text{ [VAL APPL]}$$

## $\lambda x:\text{Nat}.x$ kifejezés típusa

- ▶ alaptípusok halmaza  $K = \{\text{Nat}, \text{Bool}\}$

$$\frac{\frac{\frac{\emptyset \vdash \text{wf}, \text{Nat} \in K}{\emptyset \vdash \text{Nat}} \quad [\text{TYPE CONST}] \quad x \notin \text{dom}(\emptyset)}{\emptyset, x:\text{Nat} \vdash \text{wf}} \quad [\text{ENV X}]}{\emptyset, x:\text{Nat} \vdash x:\text{Nat}} \quad [\text{VAL FUN}]$$

## Az F1 rendszer szemantikája I.

- ▶ helyettesítési lemma

- ▶ E típusa nem változik meg

$$\frac{\Gamma, x:A \vdash E:B, \Gamma \vdash F:A}{\Gamma \vdash E[x:=F]:B} \text{ [SUBST]}$$

- ▶  $\beta$ -redukció

- ▶  $(\lambda x:A.E)F \rightarrow_{\beta} E[x:=F]$

$$\frac{\Gamma \vdash (\lambda x:A.E)F:B}{\Gamma \vdash E[x:=F]:B} \text{ [CONV } \beta]$$

- ▶  $\alpha$ -konverzió

- ▶  $\lambda x:A.E \leftrightarrow_{\alpha} \lambda y:A.E[x:=y]$

$$\frac{\Gamma \vdash \lambda x:A.E:A \rightarrow B, y \notin FV(E) \ y \notin \text{dom}(\Gamma)}{\Gamma \vdash \lambda y:A.E[x:=y]:A \rightarrow B} \text{ [CONV } \alpha]$$

## Az F1 rendszer szemantikája II.

- ▶ tárgyredukció tétele
  - ▷ ha  $\Gamma \vdash E:A$  és  $E \rightarrow F$ , akkor  $\Gamma \vdash F:A$
- ▶ tárgykiterjesztés tétele
  - ▷ ha  $\Gamma \vdash F:B$ , és van olyan  $E$ , amelyre  $E \rightarrow F$  és  $\Gamma \vdash E:A$ , akkor  $A \equiv B$
- ▶ típusok egyértelműsége
  - ▷ ha  $\Gamma \vdash E:A$  és  $\Gamma \vdash E:B$ , akkor  $A \equiv B$

# Normalizálás I.

- ▶ **erős normalizálás** tétele
  - ▶ Az F1 típusrendszerben az  $E:A$  jól típusozott kifejezésnek nincs végtelen  $\beta$ -redukciós sora
- ▶ **normalizálás** tétele
  - ▶ az F1 rendszerben minden jól típusozott  $\lambda$ -kifejezésnek van normál formája
- ▶ **I. Church-Rosser** tétel
  - ▶ ha az  $E:A$  jól típusozott kifejezésre  $E:A \rightarrow E_1:A$  és  $E:A \rightarrow E_2:A$ , akkor létezik olyan  $F:A$  kifejezés, amelyre  $E_1:A \rightarrow F:A$  És  $E_2:A \rightarrow F:A$

## Normalizálás II.

- ▶ egy jól típusozott program nem fut rosszul
  - Milner
  - 1978, A Theory of Type Polymorphism in Programming





# Köszönöm a figyelmet!

Folytatjuk...