ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΙΙ

Εισαγωγή στον Λάμδα Λογισμό

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο

Ζωή Παρασκευοπούλου, 2024

ΛΑΜΔΑ ΛΟΓΙΣΜΟΣ

λ

- Αφηρημένο μοντέλο υπολογισμού (Church, 1920s)
- Ορισμοί συναρτήσεων, εφαρμογές συναρτήσεων, μεταβλητές
- Βάση για πολλά στοιχεία γλωσσών προγραμματισμού.
  - **A Correspondence between ALGOL 60 and Church's Lambda-notation** , Landin, 1965
  - **The Next 700 Programming Languages**, Landin, 1965
- Μπορεί να επεκταθεί με διάφορες συντακτικές δομές, οδηγώντας σε γλώσσες παρόμοιες με την ML.

ΣΥΝΤΑΞΗ (CONCRETE)

$$t, u ::= x \mid t\ u \mid \lambda\ x.t \mid (t)$$

ΣΥΝΤΑΞΗ (CONCRETE)

Εφαρμογή
(application)

$$t, u ::= x \mid t\ u \mid \lambda\ x.t \mid (t)$$

Μεταβλητή

Λάμδα αφαίρεση
(lambda abstraction)
Ανώνυμη συνάρτηση (`fun x -> t`)

ΠΑΡΑΔΕΙΓΜΑΤΑ

$$\lambda x.x$$

$$\lambda f.\lambda x.f\ x$$

$$\lambda x.\lambda y.y$$

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

ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΚΑΙ ΠΡΟΣΕΤΑΙΡΙΣΤΙΚΟΤΗΤΑ

- Η **εφαρμογή συνάρτησης** είναι **αριστερά** π**ροσεταιριστική**

$$t\ u\ v = (t\ u)\ v$$

- Η **εφαρμογή συνάρτησης** έχει **μεγαλύτερη** π**ροτεραιότητα** από την **λάμβδα αφαίρεση**.

$$\lambda x.xy = \lambda x.(xy)$$

- **Notation**: Μπορούμε να γράψουμε $\lambda fx.f\ x$ αντί για $\lambda f.\lambda x.f\ x$

ΔΕΣΜΕΥΜΕΝΕΣ ΜΕΤΑΒΛΗΤΕΣ

Η μεταβλητές που είναι στο scope ενός lambda λέγονται δεσμευμένες (bound)

$$\lambda x.t$$

Scope of the **binder**

Binder

Π.χ.

$$\lambda x.x$$

$$\lambda f.\lambda x.f\ x$$

$$\lambda x.\lambda x.x$$

ΕΛΕΥΘΕΡΕΣ ΜΕΤΑΒΛΗΤΕΣ

Η μεταβλητές που δεν είναι δεσμευμένες από κάποιον binder

$$\lambda x.\lambda y.x\ (y\ \boxed{z})$$

Free variable

ΣΥΝΤΑΞΗ (ABSTRACT)

$$t, u ::= x \mid t\, u \mid \lambda x.t$$

ΣΥΝΤΑΞΗ (ABSTRACT)

Εφαρμογή
(application)

$$t, u ::= x \mid t\,u \mid \lambda x.t$$

Μεταβλητή

Λάμδα αφαίρεση
(lambda abstraction)

Ανώνυμη συνάρτηση

ΣΗΜΑΣΙΟΛΟΓΙΑ

- Θα πρέπει να ορίσουμε τι σημαίνει η εφαρμογή μιας συνάρτησης
- Π.χ. διαισθητικά ο όρος

$$(\lambda f x.f\ x)\ (\lambda\ y.y)\ (\lambda\ y.y)$$

  ανάγεται στον όρο

$$\lambda y.y$$

- Πως;

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

$$\begin{array}{ll}
[x \to u]x & = \\
[x \to u]y & = \\
[x \to u]t_1\ t_2 & = \\
[x \to u]\lambda x.t & = \\
[x \to u]\lambda y.t & =
\end{array}$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

$$\begin{array}{lcl}
[x \to u]x & = & u \\
[x \to u]y & = & \\
[x \to u]t_1\ t_2 & = & \\
[x \to u]\lambda x.t & = & \\
[x \to u]\lambda y.t & = &
\end{array}$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

$$\begin{array}{lcll}
[x \to u]x & = & u & \\
[x \to u]y & = & y & \text{if } x \neq y \\
[x \to u]t_1\ t_2 & = & & \\
[x \to u]\lambda x.t & = & & \\
[x \to u]\lambda y.t & = & &
\end{array}$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

$$\begin{array}{lcll}
[x \to u]x & = & u & \\
[x \to u]y & = & y & \text{if } x \neq y \\
[x \to u]t_1\ t_2 & = & [x \to u]t_1\ [x \to u]t_2 & \\
[x \to u]\lambda x.t & = & & \\
[x \to u]\lambda y.t & = & &
\end{array}$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

$$
\begin{array}{lcll}
[x \to u]x & = & u & \\
[x \to u]y & = & y & \text{if } x \neq y \\
[x \to u]t_1\ t_2 & = & [x \to u]t_1\ [x \to u]t_2 & \\
[x \to u]\lambda x.t & = & \lambda x.t & \\
[x \to u]\lambda y.t & = & &
\end{array}
$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

$$\begin{array}{lcll}
[x \to u]x & = & u & \\
[x \to u]y & = & y & \text{if } x \neq y \\
[x \to u]t_1\ t_2 & = & [x \to u]t_1\ [x \to u]t_2 & \\
[x \to u]\lambda x.t & = & \lambda x.t & \\
[x \to u]\lambda y.t & = & \lambda y.[x \to u]t & \text{if } x \neq y \text{ and } y \notin \mathbf{FV}(u)
\end{array}$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ

Είναι χρήσιμο να ορίσουμε την **αντικατάσταση μιας μεταβλητής x** που εμφανίζεται **ελεύθερη** σε έναν όρο **t**, με έναν άλλο όρο **u.**

$$[x \to u]t$$

Την ορίζουμε με αναδρομή στους όρους (structural recursion)

Capture avoiding

$$\begin{array}{lcll}
[x \to u]x & = & u & \\
[x \to u]y & = & y & \text{if } x \neq y \\
[x \to u]t_1\ t_2 & = & [x \to u]t_1\ [x \to u]t_2 & \\
[x \to u]\lambda x.t & = & \lambda x.t & \\
[x \to u]\lambda y.t & = & \lambda y.[x \to u]t & \text{if } x \neq y \text{ and } y \notin \mathbf{FV}(u)
\end{array}$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ: ΠΑΡΑΔΕΙΓΜΑΤΑ

$$[z \to \lambda y.y](\lambda fx.f\ x\ z) =$$

$$[y \to \lambda z.z](\lambda f.f\ (\lambda y.y)\ y) =$$

ΑΝΤΙΚΑΤΑΣΤΑΣΗ: ΠΑΡΑΔΕΙΓΜΑΤΑ

$$[z \to \lambda y.y](\lambda fx.f\ x\ z) = \lambda fx.f\ x\ (\lambda y.y)$$

$$[y \to \lambda z.z](\lambda f.f\ (\lambda y.y)\ y) = \lambda f.f\ (\lambda y.y)\ (\lambda z.z)$$

CAPTURE AVOIDING SUBSTITUTION

Όταν αντικαθιστούμε ένα όρο κάτω από μια συνάρτηση, δεν θα πρέπει να οι ελεύθερες μεταβλητές του να δεσμεύονται από την μεταβλητή της συνάρτησης. Αυτό θα άλλαζε το νόημα.

$$[x \to z](\lambda f z . f x z) \neq \lambda f z . f z z$$

Αντί αυτού:

Fresh name!
(alpha conversion)

$$[x \to z](\lambda f z . f x z) = \lambda f \boxed{z'} . f z \boxed{z'}$$

ΑΛΦΑ ΙΣΟΔΥΝΑΜΙΑ

Δύο όροι που διαφέρουν μόνο στα ονόματα των δεσμευμένων μεταβλητών τους (equal up to a **consistent renaming** of their bound variables).

Π.χ.

$$\lambda fz.f\ z\ y =_\alpha \lambda gx.g\ x\ y$$

Αλλά

$$\lambda fz.f\ z\ y \neq_\alpha \lambda xx.x\ x\ y \qquad \lambda fz.f\ z\ y \neq_\alpha \lambda fy.f\ y\ y$$

CAPTURE AVOIDING SUBSTITUTION

- Για να αποφύγουμε το πρόβλημα, όταν δουλεύουμε στο χαρτί απλά το αγνοούμε…
  - Working implicitly up to alpha conversion
- Σε έναν proof assistant δεν είναι τόσο εύκολο….
  - Αρκετές τεχνικές (named binders, nameless representations (DeBruijn indices) , higher-order abstract syntax)
  - POPLmark challenge (https://www.seas.upenn.edu/~plclub/poplmark/)

ΣΗΜΑΣΙΟΛΟΓΙΑ: SMALL-STEP SEMANTICS

$$\frac{}{(\lambda x.t_1)\ t_2 \to [x \to t_2]t_1}\ \textsc{App}_1$$

ΣΗΜΑΣΙΟΛΟΓΙΑ: SMALL-STEP SEMANTICS

$$\frac{}{(\lambda x.t_1)\ t_2 \to [x \to t_2]t_1}\ \textsc{App}_1$$

β-reduction

β-redex

ΣΗΜΑΣΙΟΛΟΓΙΑ: SMALL-STEP SEMANTICS

Full beta-reduction

$$\frac{}{(\lambda x.t_1)\ t_2 \to [x \to t_2]t_1}\ \textsc{App}_1 \qquad \frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2}\ \textsc{App}_2$$

$$\frac{t_2 \to t_2'}{t_1\ t_2 \to t_1\ t_2'}\ \textsc{App}_3 \qquad \frac{t \to t'}{\lambda x.t \to \lambda x.t'}\ \textsc{Lam}$$

ΠΑΡΑΔΕΙΓΜΑ

$$
\begin{aligned}
(\lambda x.y)(\underline{(\lambda z.zz)(\lambda t.t)}) &\longrightarrow_\beta (\lambda x.y)(\underline{(\lambda t.t)(\lambda t.t)}) \\
&\longrightarrow_\beta \underline{(\lambda x.y)(\lambda t.t)} \\
&\longrightarrow_\beta y
\end{aligned}
$$

«Κανονική μορφή»

ΠΑΡΑΔΕΙΓΜΑ

$$\Omega = (\lambda x.xx)(\lambda x.xx)$$

$$(\lambda x.xx)(\lambda x.xx) \longrightarrow_\beta (\lambda x.xx)(\lambda x.xx) \longrightarrow_\beta (\lambda x.xx)(\lambda x.xx) \longrightarrow_\beta \ldots$$

CONFLUENCE (CHURCH-ROSSER THEOREM)

$$\lambda y.y \;{}_{\beta}\!\longleftarrow (\lambda xy.y)((\lambda x.x)(\lambda x.x)) \longrightarrow_{\beta} (\lambda xy.y)(\lambda x.x)$$

$$\begin{tikzcd} & t \arrow[dl, "*"'] \arrow[dr, "*"] & \\ u_1 \arrow[dr, dotted, "*"'] & & u_2 \arrow[dl, dotted, "*"] \\ & v & \end{tikzcd}$$

«Κανονική μορφή»

EVALUATION STRATEGIES

- **Normal order:** leftmost, outermost redex is reduced first
- **Applicative order:** leftmost, innnermost redex is reduced first
- **CBV:** like applicative, but **no reduction under lambdas**
- **CBN:** like normal, but **no reduction under lambdas**

Call-by-value call-by-name and the lambda calculus. G. B. Plotkin

VALUES

Πότε μια έκφραση θεωρείται αποτιμημένη;

$$\mathsf{value}(\lambda x.t) = \mathsf{true}$$

$$\mathsf{value}(\_) = \mathsf{false}$$

SMALL-STEP SEMANTICS, CALL BY VALUE

$$\frac{\mathsf{value}(v_2) = \mathsf{true}}{(\lambda x.t_1)\ v_2 \to [x \to v_2]t_1}\ \textsc{App}_1$$

$$\frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2}\ \textsc{App}_2$$

$$\frac{\mathsf{value}(v_1) = \mathsf{true} \qquad t_2 \to t_2'}{v_1\ t_2 \to v_1\ t_2'}\ \textsc{App}_3$$

Innermost redex first

Strict strategy

leftmost redexes first

Left-to-right evaluation order

SMALL-STEP SEMANTICS, CALL BY NAME

$$\frac{}{(\lambda x.t_1)\ t_2 \to [x \to t_2]t_1}\ \textsc{App}_1$$

Outermost redexes first

Strict strategy

$$\frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2}\ \textsc{App}_2$$

leftmost redexes first

Left-to-right evaluation order

$$\cancel{\frac{\mathsf{value}(v_1) = \mathsf{true} \qquad t_2 \to t_2'}{v_1\ t_2 \to v_1\ t_2'}}\ \textsc{App}_3$$

Τα ορίσματα δεν αποτιμώνται

Αποτίμηση μόνο όταν χρειάζεται (εδώ οι μόνες τιμές είναι συναρτήσεις).

TURING COMPLETENESS

- Μπορούμε να κωδικοποιήσουμε ακριβώς τις αναδρομικές συναρτήσεις
- Μπορούμε να κωδικοποιήσουμε ένα Turing Machine!
- **Μη αποφάνσιμα** προβλήματα:
    - Είναι δύο λάμβδα όροι **ισοδύναμοι**;
    - Είναι ένας όρος **κανονικοποιήσιμος**;

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda fx.x \qquad \underline{1} = \lambda fx.fx \qquad \underline{2} = \lambda fx.f(fx) \qquad \underline{3} = \lambda fx.f(f(fx)) \qquad \ldots$$

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda f x.x \qquad \underline{1} = \lambda f x.f x \qquad \underline{2} = \lambda f x.f(f x) \qquad \underline{3} = \lambda f x.f(f(f x)) \qquad \ldots$$

Successor?

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda f x.x \qquad \underline{1} = \lambda f x.fx \qquad \underline{2} = \lambda f x.f(fx) \qquad \underline{3} = \lambda f x.f(f(fx)) \qquad \ldots$$

Successor

$$\mathsf{succ} = \lambda n f x.f(n f x)$$

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda f x . x \qquad \underline{1} = \lambda f x . f x \qquad \underline{2} = \lambda f x . f(f x) \qquad \underline{3} = \lambda f x . f(f(f x)) \qquad \ldots$$

Successor

$$\mathsf{succ} = \lambda n f x . f(n f x)$$

Addition?

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda f x . x \qquad \underline{1} = \lambda f x . f x \qquad \underline{2} = \lambda f x . f(f x) \qquad \underline{3} = \lambda f x . f(f(f x)) \qquad \ldots$$

Successor

$\mathsf{succ} = \lambda n f x . f(n f x)$

Addition

$\mathsf{add} = \lambda m n . m \ \mathsf{succ} \ n$

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda f x . x \qquad \underline{1} = \lambda f x . f x \qquad \underline{2} = \lambda f x . f (f x) \qquad \underline{3} = \lambda f x . f (f (f x)) \qquad \ldots$$

Successor

$$\mathsf{succ} = \lambda n f x . f (n f x)$$

Addition

$$\mathsf{add} = \lambda m n . m \; \mathsf{succ} \; n$$

Multiplication

$$\mathsf{mul} = \lambda m n f x . m \, (\mathsf{add} \, n) \, 0$$

Exponentiation

$$\mathsf{exp} = \lambda m n . n \, (\mathsf{mul} \, m) \, 1$$

ΑΡΙΘΜΟΙ (CHURCH NUMERALS)

$$\underline{0} = \lambda fx.x \qquad \underline{1} = \lambda fx.fx \qquad \underline{2} = \lambda fx.f(fx) \qquad \underline{3} = \lambda fx.f(f(fx)) \qquad \ldots$$

Successor

$$\mathsf{succ} = \lambda nfx.f(nfx)$$

Multiplication

$$\mathsf{mul} = \lambda mnfx.m\,(\mathsf{add}\,n)\,0$$

Predecessor

$$\mathsf{pred} = \lambda nfx.n(\lambda gh.h(gf))(\lambda y.x)(\lambda y.y)$$

Addition

$$\mathsf{add} = \lambda mn.m\ \mathsf{succ}\ n$$

Exponentiation

$$\mathsf{exp} = \lambda mn.n\,(\mathsf{mul}\,m)\,1$$

Subtraction

$$\mathsf{sub} = \lambda mn.n\,\mathsf{pred}\,m$$

BOOLEANS

$$\mathsf{T} = \lambda xy.x$$

$$\mathsf{F} = \lambda xy.y$$

If?

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad \mathsf{F} = \lambda xy.y$$

If

$$\mathsf{if} = \lambda bxy.bxy$$

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad \mathsf{F} = \lambda xy.y$$

If

$$\mathsf{if} = \lambda bxy.bxy$$

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad\qquad \mathsf{F} = \lambda xy.y$$

If

$\mathsf{if} = \lambda bxy.bxy$

Boolean connectives?

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad \mathsf{F} = \lambda xy.y$$

If

Boolean connectives

$\mathsf{if} = \lambda bxy.bxy$

$\mathsf{and} = \lambda xy.x\, y\, \mathsf{F}$

$\mathsf{or} = \lambda xy.x\, \mathsf{T}\, y$

$\mathsf{not} = \lambda x.x\, \mathsf{F}\, \mathsf{T}$

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad \mathsf{F} = \lambda xy.y$$

If

$\mathsf{if} = \lambda bxy.bxy$

Boolean connectives

$\mathsf{and} = \lambda xy.x\, y\, \mathsf{F}$

$\mathsf{or} = \lambda xy.x\, \mathsf{T}\, y$

$\mathsf{not} = \lambda x.x\, \mathsf{F}\, \mathsf{T}$

Τι συμβαίνει σε CBV?

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad \mathsf{F} = \lambda xy.y$$

If

$\mathsf{if} = \lambda bxy.bxy$

Boolean connectives

$\mathsf{and} = \lambda xy.x\, y\, \mathsf{F}$

$\mathsf{or} = \lambda xy.x\, \mathsf{T}\, y$

$\mathsf{not} = \lambda x.x\, \mathsf{F}\, \mathsf{T}$

Is zero?

BOOLEANS

$$\mathsf{T} = \lambda xy.x \qquad\qquad \mathsf{F} = \lambda xy.y$$

If

$\mathsf{if} = \lambda bxy.bxy$

Boolean connectives

$\mathsf{and} = \lambda xy.x\, y\, \mathsf{F}$

$\mathsf{or} = \lambda xy.x\, \mathsf{T}\, y$

$\mathsf{not} = \lambda x.x\, \mathsf{F}\, \mathsf{T}$

Is zero?

$\mathsf{iszero} = \lambda nxy.n(\lambda z.y)x$

Επίσης: ζεύγη, λίστες…

ΑΝΑΔΡΟΜΗ

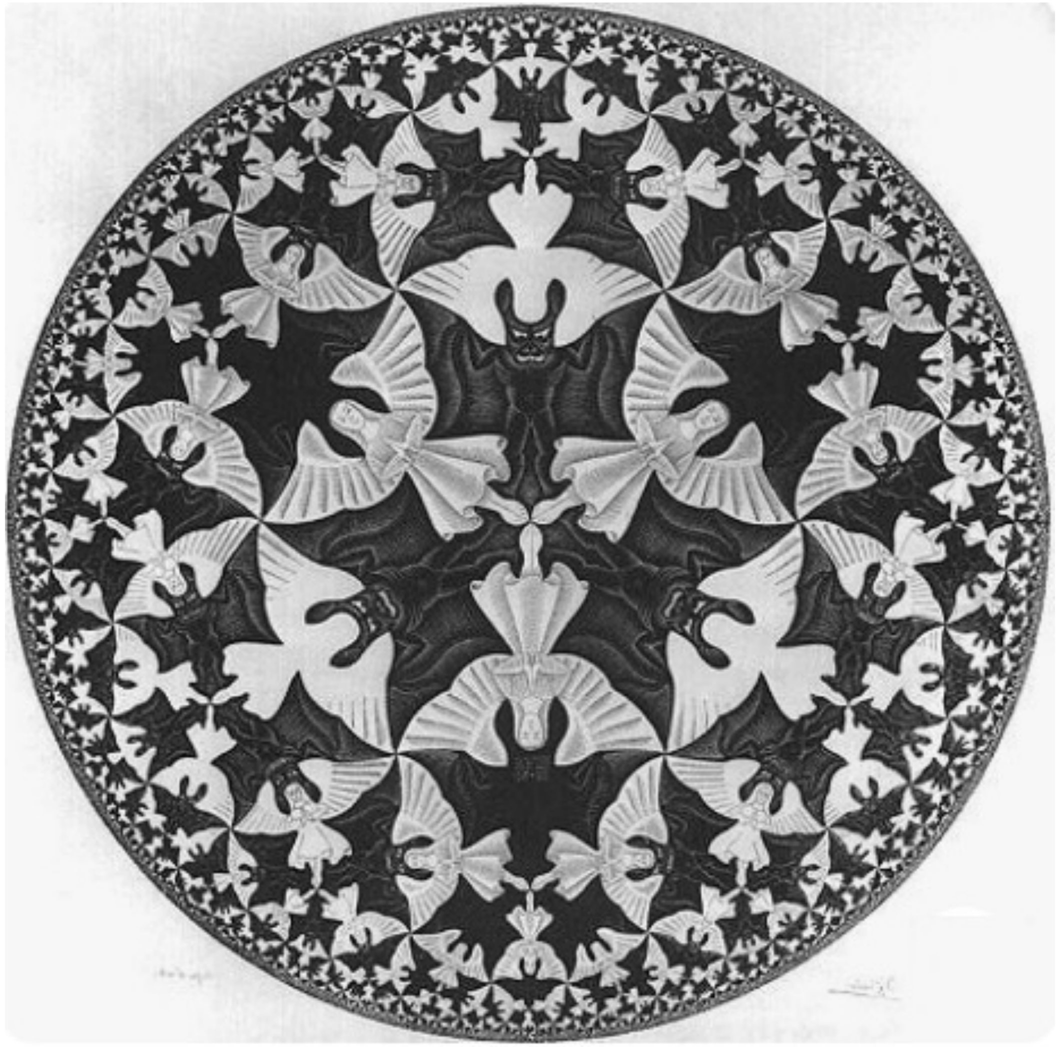
- Πως μπορώ να υπολογίσω αναδρομικές συναρτήσεις;
- Έστω

$$\mathsf{fact}' = \lambda fact.\lambda n.\ \mathsf{if\ iszero}\ n\ \mathsf{else}\ 1\ \mathsf{then}\ n * (fact(n-1))$$

- Στα μαθηματικά, το fixed point μιας συνάρτησης f, είναι μια τιμή τέτοια ώστε f(x) = x
- Γιατί; Έστω fact το fixed point της fact'. Τότε $\mathsf{fact} = \mathsf{fact}'\ \mathsf{fact}$
  Άρα,

$$\mathsf{fact} = \lambda n.\ \mathsf{if\ iszero}\ n\ \mathsf{else}\ 1\ \mathsf{then}\ n * (\mathsf{fact}(n-1))$$

- Άρα, η αναδρομική συνάρτηση που υπολογίζει το παραγοντικό, είναι το fixed point της fact'!



M. C. Escher. “Circle Limit IV (Heaven and Hell).”

FIXED POINTS

- Στον λάμδα λογισμό, κάθε όρος έχει ένα fixed point
- Θέλουμε έναν όρο **Y** τέτοιος ώστε για κάθε t

$$\mathsf{Y}\, t \longrightarrow_{\beta} t\,(\mathsf{Y}\, t)$$

- Άρα για κάθε όρο t, το (**Y** t) είναι fixed point
- Το Y λέγεται **fixed point combinator**

Y COMBINATOR

- Y combinator (CBN)

$$\mathsf{Y} = \lambda f.\ (\lambda x.\ f(x\ x))\ (\lambda x.\ f(x\ x))$$

- Z combinator (CBV) (or *strict* Y combinator)

$$\mathsf{Z} = \lambda f.\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y)\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))$$

- Π.χ.

$$\mathsf{fact} = \mathsf{Z}\ (\lambda fact.\lambda n.\ \mathsf{if\ iszero}\ n\ \mathsf{else}\ 1\ \mathsf{then}\ n * (fact(n-1)))$$

Y COMBINATOR

$$
\begin{aligned}
\mathsf{Y}\,t &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))\,t \\
&\longrightarrow_\beta (\lambda x.t(xx))(\lambda x.t(xx)) \\
&\longrightarrow_\beta t((\lambda x.t(xx))(\lambda x.t(xx))) \\
{}_\beta\!\longleftarrow\ & t\,(\mathsf{Y}\,t)
\end{aligned}
$$

- Y combinator (CBN)

$$\mathsf{Y} = \lambda f.\ (\lambda x.\ f(x\ x))\ (\lambda x.\ f(x\ x))$$

- Z combinator (CBV) (or *strict* Y combinator)

$$\mathsf{Z} = \lambda f.\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))$$

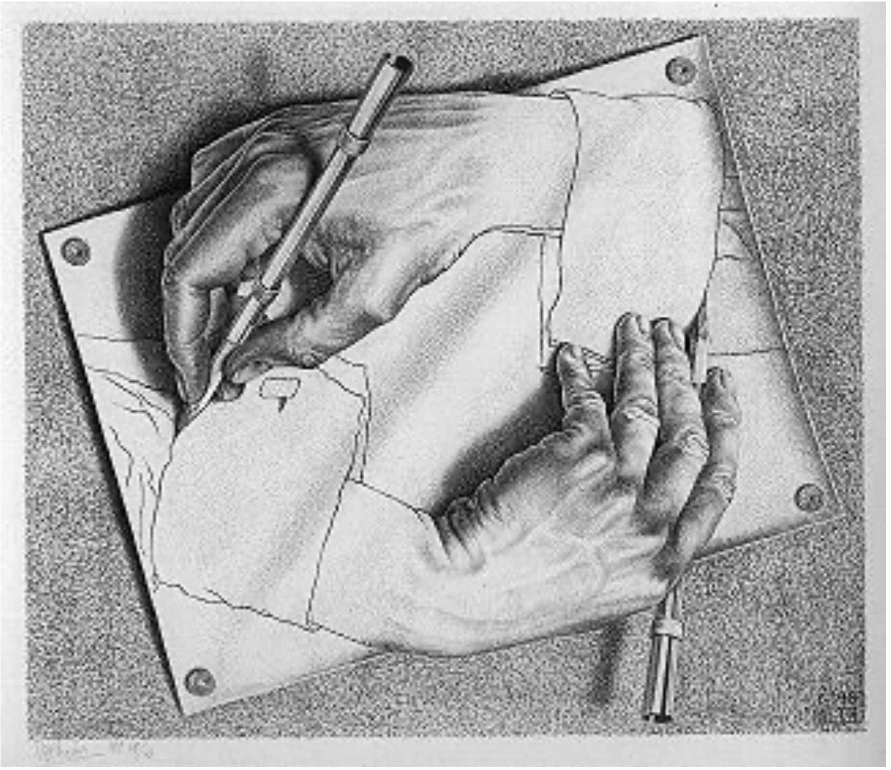
- Π.χ.

$$\mathsf{fact} = \mathsf{Z}\ (\lambda fact.\lambda n.\ \mathsf{if\ iszero}\ n\ \mathsf{else}\ 1\ \mathsf{then}\ n * (fact(n-1)))$$

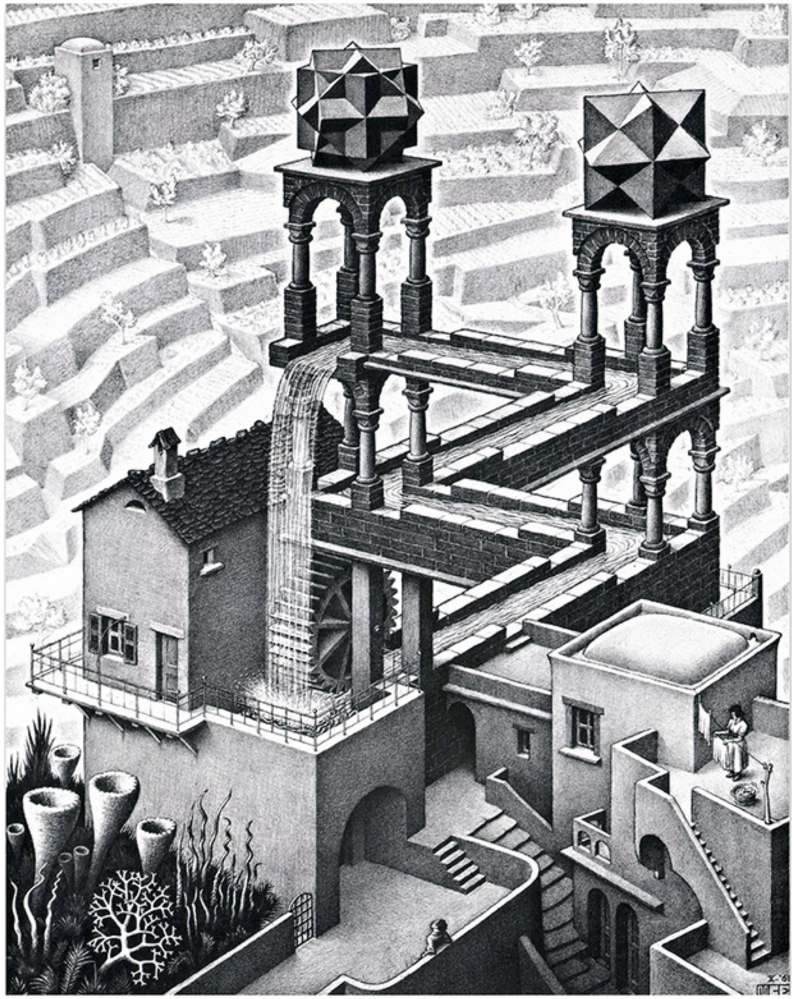
Y COMBINATOR

$$
\begin{aligned}
\mathsf{fact}\,\underline{2} &= (\mathsf{Y}F)\,\underline{2} \\
&=_\beta F\,(\mathsf{Y}F)\,\underline{2} \\
&\xrightarrow{*}_\beta \mathsf{if}\,(\mathsf{iszero}\,\underline{2})\,\underline{1}\,(\mathsf{mul}\,\underline{2}\,((\mathsf{Y}F)\,(\mathsf{pred}\,\underline{2}))) \\
&\xrightarrow{*}_\beta \mathsf{if}\,\mathsf{false}\,\underline{1}\,(\mathsf{mul}\,\underline{2}\,((\mathsf{Y}F)\,(\mathsf{pred}\,\underline{2}))) \\
&\xrightarrow{*}_\beta \mathsf{mul}\,\underline{2}\,((\mathsf{Y}F)\,(\mathsf{pred}\,\underline{2})) \\
&\xrightarrow{*}_\beta \mathsf{mul}\,\underline{2}\,((\mathsf{Y}F)\,\underline{1}) \\
&\vdots \\
&\xrightarrow{*}_\beta \mathsf{mul}\,\underline{2}\,(\mathsf{mul}\,\underline{1}\,\underline{1}) \\
&\xrightarrow{*}_\beta \underline{2}
\end{aligned}
$$

όπου $F = \lambda fn.\mathsf{if}\,(\mathsf{iszero}\,n)\,\underline{1}\,(\mathsf{mul}\,n\,(f\,(\mathsf{pred}\,n)))$



M. C. Escher. “Drawing Hands.”



M. C. Escher. “Waterfall.”