§ 5 Podstawy programowania funkcyjnego

model programowania
↳ λ-rachunek, ewaluacja

Przykład:

$$\begin{tikzcd}
 & \text{square}(3+4) \arrow[dl] \arrow[dr] & & \\
\text{square } 7 \arrow[ddr] & & (3+4)*(3+4) \arrow[dl] \arrow[dr] & \\
 & 7*(3+4) \arrow[d] & & (3+4)*7 \arrow[dll] \\
 & 7*7 \arrow[d] & & \\
 & 49 & &
\end{tikzcd}$$

- czy zawsze jest wynik?
- czy wynik jest jednoznaczny?
- czy wynik zawsze się znajduje?

5.1 λ-rachunek

Operacje: aplikacja $F.A$
abstrakcja $\lambda x. M(x)$

Def: λ-termy Λ
z stałymi C i zmiennymi V

(i) $c \in C \rightsquigarrow c \in \Lambda$

(ii) $x \in V \rightsquigarrow x \in \Lambda$

(iii) $M, N \in \Lambda \rightsquigarrow (MN) \in \Lambda$

(iv) $M \in \Lambda, x \in V \rightsquigarrow (\lambda x. M) \in \Lambda$

Przykład:

$x, xc, \lambda x.xc \in \Lambda$

$y(\lambda x.xc), (\lambda x.xc)y, xx \in \Lambda$

Uwaga:

$MNP \equiv (MN)P$

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

czyli $\lambda x.yx = \lambda x.(yx)$, a nie $(\lambda x.y).x$

<u>Def:</u> Wolne zmienne FV

(i) $FV(c) = \emptyset$

(ii) $FV(x) = \{x\}$

(iii) $FV(MN) = FV(M) \cup FV(N)$

(iv) $FV(\lambda x.M) = FV(M) \setminus \{x\}$

<u>Przykład:</u>

$$M = (\lambda x.xy)(\lambda y.yz)$$

$$FV(M) = \{y, z\}$$

$$\rightsquigarrow \quad M \text{ "="} \ (\lambda x.xy)(\lambda u.uz)$$

<u>α-konwersja:</u>

$$\lambda x.M =_\alpha \lambda y.M\{x/y\}, \ y \in V \text{ nowa}$$

<u>Przykład:</u>

$$(\lambda y.x+y)[x/y+5] \overset{?}{=} \lambda y.(y+5)+y$$

$$(\lambda y.x+y)[x/y+5]$$

$$=_\alpha (\lambda u.x+u)[x/y+5]$$

$$= (\lambda u.(y+5)+u)$$

$\hookrightarrow$ capture-avoiding substitution

<u>Def:</u> λ- substytucja $M[x/N]$

(i) $x[x/N] = N$

(ii) $y[x/N] = y \quad , y \neq x$

(iii) $(MP)[x/N] = (M[x/N])(P[x/N])$

(iv) $(\lambda x.M)[x/N] = \lambda x.M$

(v) $(\lambda y.M)[x/N] = \lambda y.(M[x/N])$

$x \neq y, \; y \notin FV(N)$

(vi) $(\lambda y.M)[x/N] = \lambda u.((M\{y/u\})[x/N])$

$x \neq y, \; y \in FV(N)$, u nowa

<u>Def:</u> β-redukcja

(i) $(\lambda x.M)N \longrightarrow_\beta M[x/N]$

(ii) M ewaluuje się do M', jeżeli $M \xrightarrow{*}_\beta M'$

<u>Przykład:</u>

(i) $(\lambda n.\lambda f.\lambda x.\, n\, f\,(f x))\,(\lambda f.\lambda x.x)$

$\rightarrow_\beta \lambda f.\lambda x.\,(\lambda f.\lambda x.x)\, f\,(f x)$

$\rightarrow_\beta \lambda f.\lambda x.\,(\lambda x.x)\,(f x)$

$\rightarrow_\beta \lambda f.\lambda x.\,(f x)$

(ii) $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$

$\to_\beta (\lambda x.y)((\lambda w.w)(\lambda w.w))$

$\to_\beta (\lambda x.y)(\lambda w.w)$

$\to_\beta y$

Oraz $(\lambda x.y)((\lambda z.zz)(\lambda w.w))$

$\to_\beta y$

(iii)

$$\begin{tikzcd}
 & (\lambda x.xx)((\lambda x.x)y) \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\
(\lambda x.xx)y \arrow[ddr, "\beta"'] & & ((\lambda x.x)y)((\lambda x.x)y) \arrow[d, "\beta"] \\
 & & y((\lambda x.x)y) \arrow[dl, "\beta"] \\
 & yy &
\end{tikzcd}$$

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

$\to_\beta (\lambda x.xx)(\lambda x.xx)$

$\to_\beta \cdots$

i

$$\begin{tikzcd}
 & (\lambda x.y)((\lambda x.xx)(\lambda x.xx)) \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\
y & & \cdot \arrow[r, "\beta"] & \cdots
\end{tikzcd}$$

Def: β-równoważność $M =_\beta N$

$M =_\beta N$ wtw. $M \overset{*}{\longleftrightarrow}_\beta N$

5.2 Programowanie w λ-rachunku

(i) Boolean

$T \equiv \lambda xy.x$, $F \equiv \lambda xy.y$

$\text{and} \equiv \lambda ab.\, a\, b\, F$

$\text{if\_then\_else} \equiv \lambda x.x$

$$
\begin{aligned}
&\text{and } F\ T \\
&\equiv (\lambda ab.\, a\, b\, F)(\lambda xy.y)(\lambda xy.x) \\
&\xrightarrow{*}_\beta (\lambda xy.y)(\lambda xy.x)\, F \\
&\to_\beta (\lambda y.y)\, F \\
&\to_\beta F
\end{aligned}
$$

$$
\begin{aligned}
&\text{if } T \text{ then } M \text{ else } N \\
&\equiv (\lambda x.x)\, T\, M\, N \\
&\to_\beta T\, M\, N \\
&\equiv (\lambda xy.x)\, M\, N \\
&\to_\beta (\lambda y.M)\, N \\
&\to_\beta M
\end{aligned}
$$

(ii) Pary i Listy

$$\text{pair} \equiv \lambda xyz.\, zxy$$

$$\text{first} \equiv \lambda p.\, p(\lambda xy.x)$$

$$\text{second} \equiv \lambda p.\, p(\lambda xy.y)$$

$$\begin{aligned}
&\text{first}(\text{pair } M N) \\
\equiv\ &\text{first}((\lambda xyz.\, zxy) M N) \\
\xrightarrow{*}\ &\text{first}\ (\lambda z.\, z M N) \\
\equiv\ &(\lambda p.\, p(\lambda xy.x))(\lambda z.\, z M N) \\
\to\ &(\lambda z.\, z M N)(\lambda xy.\, x) \\
\to\ &(\lambda xy.\, x) M N \\
\xrightarrow{*}\ &M
\end{aligned}$$

$$\text{nil} \equiv F$$

$$\text{cons} \equiv \text{pair}$$

$$\text{head} \equiv \text{first}$$

$$\text{tail} \equiv \text{second}$$

$$\text{is\_nil} \equiv \lambda l.\, l(\lambda htd.\, F)\, T$$

$$\begin{aligned}
&\text{is_nil}\ (\text{pair } M N) \\
\to\ &((\lambda xyz.\, zxy) M N)(\lambda htd.\, F)\, T \\
\xrightarrow{*}\ &(\lambda z.\, z M N)(\lambda htd.\, F)\, T \\
\xrightarrow{*}\ &(\lambda htd.\, F) M N T \\
\xrightarrow{*}\ &F
\end{aligned}$$

(iii) Liczby naturalne

$\overline{n} \equiv \lambda f. \lambda x. f^n x$ (Church)

$\overline{n}$ bierze funkcję a zwraca n-tą iterację tej funkcji ("repeat n times")

<u>Def</u>: $f : \mathbb{N}^m \longrightarrow \mathbb{N}$

$\overline{f} \in \Lambda$ realizuje f, jeżeli

$\forall x_1, \ldots x_m \in \mathbb{N} : \overline{f}\, \overline{x}_1 \ldots \overline{x}_m =_\beta \overline{f(x_1, \ldots x_m)}$

<u>Przykład</u>:

$$s \equiv \lambda n. \lambda f. \lambda x.\ n\, f\, (f\, x)$$

$$\begin{aligned} s\, \overline{n} &\longrightarrow \lambda f. \lambda x. (\lambda f. \lambda x. f^n x)\, f\, (f\, x) \\ &\longrightarrow \lambda f. \lambda x. (\lambda x. f^n x)(f\, x) \\ &\longrightarrow \lambda f. \lambda x. f^n (f\, x) \\ &\equiv \overline{n+1} \end{aligned}$$

czyli s realizuje funkcję $n+1$

$plus \equiv \lambda m. \lambda n. \lambda f. \lambda x.\ m\ f\ (n\ f\ x)$

$(\text{lub } plus \equiv \lambda m. \lambda n.\ m\ s\ n)$.

$mult \equiv \lambda m. \lambda n. \lambda f.\ m\ (plus\ n)\ 0$

$is\_zero \equiv \lambda n.\ n\ (\lambda x. F)\ T$

$p(n) = n \dot{-} 1 = \begin{cases} x - 1 ; & x \geq 1 \\ 0 ; & x = 0 \end{cases} \longrightarrow$ ZD6

(iv) Rekursja

<u>Przykład</u>: silnia

$sil \equiv \lambda n.$ if $n = 0$ then 1 else $n * (\underline{sil}\ (n \dot{-} 1))$

$\overline{sil} \equiv \lambda f. \lambda n.$ if $n = 0$ then 1 else $n * (f\ (n \dot{-} 1))$

<u>Punkt stały</u>

(i) $\forall F \exists X : FX =_\beta X$

(ii) Dla $Y \equiv \lambda f. (\lambda x. f(x\ x))(\lambda x. f(x\ x))$

mamy $\forall F : F(YF) =_\beta YF$

$W \equiv \lambda x. F(x\ x)$, $X \equiv WW$

$\hookrightarrow X = (\lambda x. F(x\ x))W$

$= F(W\ W)$

$= FX$

$(YF = WW = X)$

D6

$p(x) = x \dot{-} 1$

Pomysł: pary $(x+1, x)$

$$\phi \equiv \lambda p.\, \lambda z.\, z\,(s\,(p\,T))\,(p\,T)$$

$$\overline{p} \equiv \lambda n.\, n\, \phi\, (\lambda z.\, z\,0\,0)\, F$$

$$
\begin{aligned}
&\overline{p}\ 2 \\
&= (\lambda n.\, n\,\phi\,(\lambda z.\, z\,0\,0)\,F)\,(\lambda f.\, \lambda x.\, f^2 x) \\
&= (\lambda f.\, \lambda x.\, f^2 x)\,\phi\,(\lambda z.\, z\,0\,0)\,F \\
&= \phi^2\,(\lambda z.\, z\,0\,0)\,F \\
&= \phi\,(\lambda z.\, z\,(s\,(\lambda z.\, z\,0\,0)\,T)\,((\lambda z.\, z\,0\,0)\,T))\,F \\
&= \phi\,(\lambda z.\, z\,(s\,0)\,0)\,F \\
&= (\lambda z.\, z\,(s\,(\lambda z.\, z\,1\,0)\,T)\,((\lambda z.\, z\,1\,0)\,T))\,F \\
&= (\lambda z.\, z\,(s\,1)\,1)\,F \\
&= F\,(s\,1)\,1 \\
&= 1
\end{aligned}
$$

więc $\mathrm{sil} \equiv Y\, \overline{\mathrm{sil}}$

$$
\begin{aligned}
\mathrm{sil}\ 2 &\equiv (Y\, \overline{\mathrm{sil}})\ 2 \\
&= \overline{\mathrm{sil}}\ (Y\, \overline{\mathrm{sil}})\ 2 \\
&= (\lambda f.\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n * f(n \dot{-} 1)) \\
&\qquad\qquad\qquad\qquad (Y\, \overline{\mathrm{sil}})\ 2 \\
&= \text{if } 2 = 0 \text{ then } 1 \text{ else } 2 * (Y\, \overline{\mathrm{sil}})\ 1 \\
&= 2 * (Y\, \overline{\mathrm{sil}})\ 1 \\
&= 2 * (\overline{\mathrm{sil}}\ (Y\, \overline{\mathrm{sil}})\ 1) \ldots
\end{aligned}
$$

(v) Funkcje rozstrzegalne $\longrightarrow$ D7

$$S \equiv \lambda n.\lambda f.\lambda x.\ n\ f\ (f\ x)$$

$$pr^n_m \equiv \lambda x_1. \ldots \lambda x_n.\ x_m$$

$$c^n_m \equiv \lambda x_1. \ldots \lambda x_n.\ \overline{m}$$

$$O \equiv \lambda x_1. \ldots \lambda x_n.\ \overline{f}\ (\overline{g}_1\ x_1 \ldots x_n) \ldots (\overline{g}_k\ x_1 \ldots x_n)$$

$$
\begin{aligned}
H \equiv \lambda h.\lambda x_1 \ldots \lambda x_n.\lambda y.\ & \text{if } y = 0 \\
& \text{then } \overline{g}\ x_1 \ldots x_n \\
& \text{else } \overline{f}\ x_1 \ldots x_n\ (y \dot{-} 1)\ (h\ x_1 \ldots x_n\ (y \dot{-} 1))
\end{aligned}
$$

$$PR \equiv Y H$$

Funkcje μ-rekursywne

(i) funkcje bazowe

$$s(n) = n+1$$

$$pr^i_m(x_1, \dots x_m) = x_i$$

$$c^i_m(x_1, \dots x_m) = i$$

(ii) schematy

$$(f \circ \langle g_1, \dots g_k \rangle)(x_1, \dots x_m) =$$

$$f(g_1(x_1, \dots x_m), \dots g_k(x_1, \dots x_m))$$

$$R_{g,f}(x_1, \dots x_m, y) =$$

$$\begin{cases} g(x_1, \dots x_m) & ; y = 0 \\ f(x_1, \dots x_m, y \dot- 1, R_{g,f}(x_1, \dots x_m, y \dot- 1)) & ; y > 0 \end{cases}$$

$$\mu f(x_1, \dots x_m) =$$

$$\begin{cases} \min M_{f, x_1, \dots x_m} & ; M_{f, x_1, \dots x_m} \neq \emptyset \\ \uparrow & ; M_{f, x_1, \dots x_m} = \emptyset \end{cases}$$

dla

$$M_{f, x_1, \dots x_m} = \{ y \geq 0 \mid f(x_1, \dots x_m, y) = 0 \wedge \forall k < y \; f(x_1, \dots x_m, k) \downarrow \}$$

<u>Def</u>: Funkcje μ-rekursywne $\mathcal{R}_p$

(i) $\{ s, pr^i_m, c^i_m \} \subseteq \mathcal{R}_p$

(ii) $f, g_1, \ldots g_k \in \mathcal{R}_p \rightsquigarrow f \circ \langle g_1, \ldots g_k \rangle \in \mathcal{R}_p$

(iii) $f, g \in \mathcal{R}_p \rightsquigarrow R_{g,f} \in \mathcal{R}_p$

(iv) $f \in \mathcal{R}_p \rightsquigarrow \mu f \in \mathcal{R}_p$

(v) $\mathcal{R}_p$ jest najmniejszą klasą funkcji, która spełnia (i) - (iv).

<u>Przykład</u>:

$g(x) := x$, czyli $g = pr^1_1$

$f(x, y, z) := z + 1$, czyli $f = s \circ \langle \pi^3_3 \rangle$

więc $R_{g,f} \in \mathcal{R}_p$

$R_{g,f}(x, 0) = g(x) = x + 0$

dla $y > 0$:

$$R_{g,f}(x, y) = f(x, y-1, R_{g,f}(x, y-1))$$
$$= R_{g,f}(x, y-1) + 1$$
$$\overset{z.i.}{=} (x + (y-1)) + 1 = x + y$$

<u>Twierdzenie</u>

$f \in \mathcal{R}_p$ wtw. istnieje maszyna Turinga, która obliczy f.

$$
\begin{aligned}
G \equiv \lambda h.\lambda y.\lambda x_1.\dots\lambda x_m.\ & \text{if } \bar{f}\, x_1 \dots x_m\, y = 0 \\
& \text{then } y \\
& \text{else } h\,(S\,y)\, x_1 \dots x_m
\end{aligned}
$$

$$MU \equiv (Y\, G)\, 0$$

Twierdzenie

$$\forall f \in Q_p\ \exists \bar{f} \in \Lambda$$

$$\forall x_1, \dots x_m \in \mathbb{N} : \bar{f}\, \bar{x}_1 \dots \bar{x}_m =_\beta \overline{f(x_1, \dots x_m)}$$

5.3 Postacie normalne i zbieżność

czy mamy $\bar{f}\, x_1 \dots x_m \xrightarrow{*}_\beta \overline{f(x_1, \dots x_m)}$?

$$
\begin{tikzcd}
N \arrow[dr, "*"'] & =_\beta & M \arrow[dl, "*"] \\
& \exists S\ ? &
\end{tikzcd}
$$

jeżeli tak i M nie redukuje się, to implikuje $M \equiv S$, czyli $N \xrightarrow{*} M$.

Def:

N jest w postaci normalnej, jeżeli N nie redukuje się ($N \not\to_\beta$).

N jest postacią normalną dla M, jeżeli

$M \xrightarrow{*}_\beta N$ i $N \not\to_\beta$

Def:

$\to$ jest Church-Rosser, jeżeli

$\forall N, M: N =_\beta M \to \exists S: N \xrightarrow{*}_\beta S \wedge M \xrightarrow{*}_\beta S$

$\to$ jest zbieżna, jeżeli

$\forall N, M1, M2: (N \xrightarrow{*}_\beta M1 \wedge N \xrightarrow{*}_\beta M2)$

$\to \exists S: M1 \xrightarrow{*}_\beta S \wedge M2 \xrightarrow{*}_\beta S$

$\to$ jest lokalnie zbieżna, jeżeli

$\forall N, M1, M2: (N \longrightarrow_\beta M1 \wedge N \longrightarrow_\beta M2)$

$\to \exists S: M1 \xrightarrow{*}_\beta S \wedge M2 \xrightarrow{*}_\beta S$

$\to$ jest terminująca, jeżeli nie istnieje nieskończony ciąg β-redukcji

Twierdzenie

$\to$ jest Church-Rosser wtw. $\to$ jest zbieżna.

$$N \overset{i}{\longleftrightarrow} M \rightsquigarrow \exists R:\ N \xrightarrow{*} R \xleftarrow{*} M$$

$$\begin{tikzcd}
N \arrow[r, leftrightarrow, "i"] \arrow[d, "*"'] & M' \arrow[r, leftrightarrow] \arrow[r, dashed, bend right] \arrow[dl, "*"] \arrow[dr, phantom, "\to \text{zbieżna}"] & M \arrow[d, "*"] \\
\exists R' \arrow[rr, "*"] & & \exists R
\end{tikzcd}$$

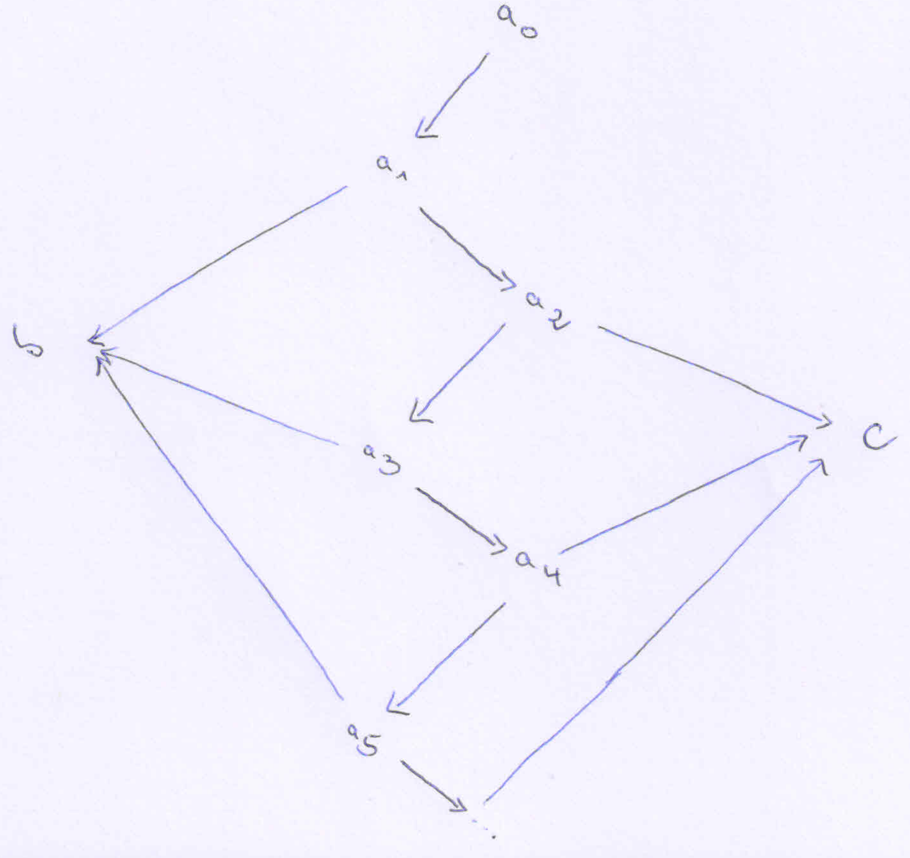
z.i.

Uwaga

- $\to$ lokalnie zbieżna nie implikuje $\to$ zbieżna

(i) $S \leftarrow N \rightleftarrows M \to R$

(ii)



Twierdzenie

Jeżeli $\to$ jest terminująca, to
$\to$ jest zbieżna wtw. $\to$ jest lokalnie zbieżna.

Przykład

$$
\begin{tikzcd}
& (\lambda x.xx)(\lambda z.zy)x \arrow[dl] \arrow[dr] & \\
(\lambda z.zy)x((\lambda z.zy)x) \arrow[d] \arrow[dr] & & (\lambda x.xx)(xy) \arrow[ddl, bend left] \\
(\lambda z.zy)x(xy) \arrow[dr] & xy((\lambda z.zy)x) \arrow[d] & \\
& xy(xy) &
\end{tikzcd}
$$

Twierdzenie

β-redukcja jest zbieżna (Church-Rosser),
czyli postać normalna jest jednoznaczna
— o ile istnieje!

Przykład

$(\lambda x.xx)(\lambda x.xx)$ nie ma postaci normalnej.

$$\begin{tikzcd}
(\lambda y.c)((\lambda x.xx)(\lambda x.xx)) \arrow[d] \arrow[dr] & \\
(\lambda y.c)((\lambda x.xx)(\lambda x.xx)) \arrow[d] \arrow[r] & c \\
\vdots \arrow[ur] &
\end{tikzcd}$$

5.4 Strategie redukcji

Przykład:

(i)

$$\begin{tikzcd}
& \text{first (square 4) (square 2)} \arrow[dl, "*"'] \arrow[ddr] & \\
\text{first 16 (square 2)} \arrow[d, "*"'] & & \\
\text{first 16 4} \arrow[dr] & & \text{square 4} \arrow[dl, "*"] \\
& 16 &
\end{tikzcd}$$

leftmost-innermost
(eager evaluation)

leftmost-outermost
(lazy evaluation)

(ii) $f\,x = x+1$

$g\,x = 5$

$$\begin{tikzcd}
 & g\,(f\,10) \arrow[dl] \arrow[dr] & \\
g\,11 \arrow[d] & & 5 \\
5 & &
\end{tikzcd}$$

(iii) $f\,x = f\,(x+1)$

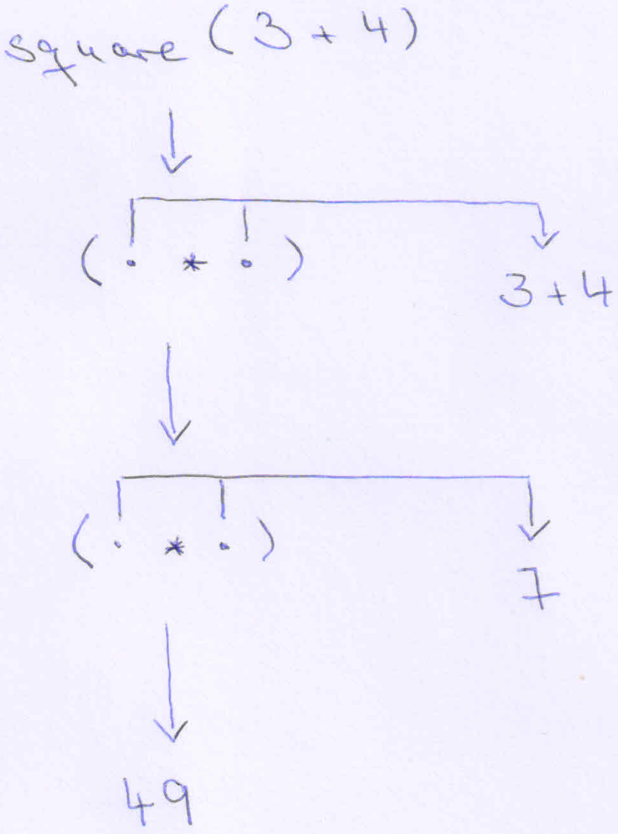
$g\,x = 5$

$$\begin{tikzcd}
 & g\,(f\,10) \arrow[dl, "*"'] \arrow[dr] & \\
g\,(f\,11) \arrow[d, "*"] & & 5 \\
g\,(f\,12) \arrow[d, "*"] & & \\
\vdots & &
\end{tikzcd}$$

- leftmost-outermost jest non strict
- leftmost-outermost znajduje wszystkie postacie normalne
- leftmost-outermost ewtl. duplikuje argumenty funkcji.

w Haskellu: lazy evaluation

na podstawie "grafów"



(tak jak w definicjach let i where)

§6 Listy nieskończone

```
ones = 1 : ones
enum n = n : enum (n+1)
nats = enum 1

fibs = 0 : 1 : (zipWith (+) fibs (tail fibs))

(!!) : [α] -> Int -> α
take :: Int -> [α] -> [α]

take 8 fibs = [0,1,1,2,3,5,8,13]
fibs !! 5 = 3

(x : xs) !! 1 = x
(x : xs) !! (n+1) = xs !! n
```

Producer → Transformer → Consumer

Przykład

(i) squares n = (n * n) : squares (n+1)

(squares 0) !! 3

→ ((0 * 0) : squares (0+1)) !! 3

→ (squares (0+1)) !! 2

→ (((0+1) * (0+1)) : (squares ((0+1)+1))) !! 2

→ (squares ((0+1)+1)) !! 1

→ ((((0+1)+1) * ((0+1)+1) : (squares (((0+1)+1)+1))) !! 1

→ ((0+1)+1) * ((0+1)+1)

$\xrightarrow{*}$ 4

(ii) Sito Eratostenesa

```
primes = sieve [2..]
sieve (x:xs)
    = x : sieve [y | y <- xs, y 'mod' x > 0]
```