

# Preghiere esami

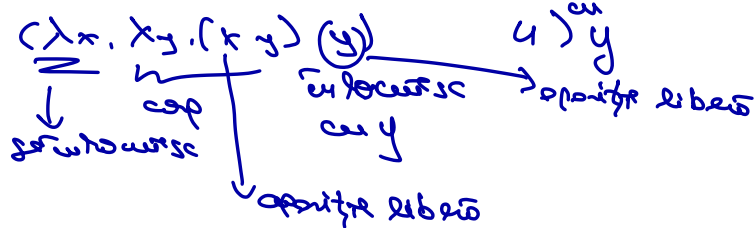
## I (ACQUI LAMBA)

### 1 Calcol simboli

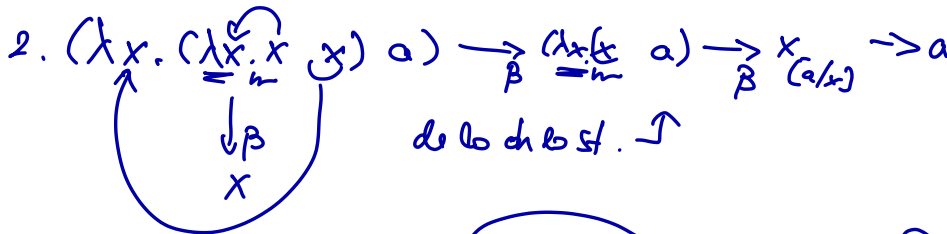
a)  $\lambda x. \lambda y. (x y)$  lib. lib.

2)  $\lambda x. x$  lib.

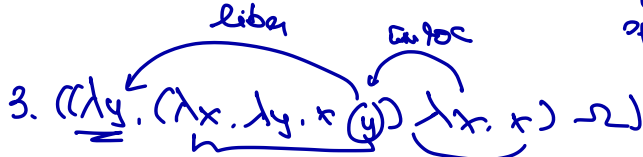
3)  $\lambda x. \lambda y. (x y)$  lib.



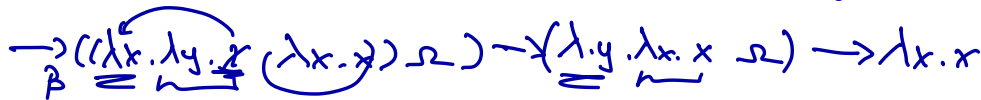
b)  $\beta$ -reduction:  $\lambda y. (y y)$   $\rightarrow$   $y y$  dove  $y$  diventa  $y$  e  $y$  diventa  $y$ .  
 $\lambda x. \lambda y. (x y) y \rightarrow \lambda x. \lambda z. (x z) y \rightarrow \lambda z. (y z) = \lambda z. (y z)$   
 dove  $y$  diventa  $z$ .



dove  $\lambda x. x$   $\rightarrow$   $x$  dove  $x$  diventa  $x$ .  
 $(\lambda x. (\lambda x. x) x) a \rightarrow (\lambda x. x) a \rightarrow a$



di cui se per  $\lambda y$ ,  $\lambda x$   $\rightarrow$   $\lambda x$  dove  $\lambda y$  diventa  $\lambda x$  e  $\lambda x$  diventa  $\lambda x$ .



4.  $(\lambda y. ((\lambda x. \lambda y. x) (y)) (\lambda x. \lambda y. y)) \rightarrow ((\lambda x. \lambda y. x) (\lambda x. \lambda y. y)) \rightarrow (\lambda y. \lambda x. \lambda y. y) \rightarrow \lambda x. \lambda y. y$

5.  $((\lambda x. \lambda y. \lambda z. y) (\lambda x. x)) (\lambda z. \lambda t. z) \rightarrow (\lambda y. \lambda z. y) (\lambda z. \lambda t. z) \rightarrow \lambda z. \lambda t. z$   
 dove  $\lambda x. x$   $\rightarrow$   $x$  dove  $\lambda x$  diventa  $x$  e  $\lambda y$  diventa  $y$ .  
 dove  $\lambda z. \lambda t. z$   $\rightarrow$   $\lambda t. z$  dove  $\lambda z$  diventa  $t$  e  $\lambda t$  diventa  $z$ .  
 $((\lambda x. \lambda y. \lambda z. y) (\lambda x. x)) (\lambda z. \lambda t. z) \rightarrow (\lambda y. \lambda z. y) (\lambda z. \lambda t. z) \rightarrow \lambda z. \lambda t. z$

$$6. \quad 0 := \lambda f. \lambda x. x \quad \text{succ} := \lambda n. \lambda f. \lambda x. (f ((n f) x))$$

(succ 0)!

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

↳ unnecessary for co-recursion

$$\rightarrow \lambda f. \lambda x. (f (\lambda x. x) x) \rightarrow \lambda f. \lambda x. (f x) \text{ we avoid case - I need for}$$

$$7. \quad \text{true} = \lambda x. \lambda y. x$$

$$\text{false} = \lambda x. \lambda y. y$$

$$\lambda x. \lambda y. (x y \text{ true})$$

$$\text{true} = \lambda x. \lambda y. x$$

$$\text{false} = \lambda x. \lambda y. y$$

$$\text{op} = \lambda x. \lambda y. (x y \lambda x. \lambda y. x)$$

NOT -> not, not both

on op's re test and pos, on re cu not pos.

$$(\text{op true } y) = (\lambda x. \lambda y. (x y \text{ true})) \text{ true } y \rightarrow (\text{true } y \text{ true}) \rightarrow$$

problem in deni of  
true in def den, den  
of false fixed to  
no instructions in den.

of def every in comp ref.  
is in fi op's. but x is pos, y & true

$$\rightarrow (\lambda x. \lambda y. x) y \text{ true} \rightarrow y$$

def true

$$(\text{op false } y) \rightarrow (\lambda x. \lambda y. (x y \text{ true})) \text{ false } y \rightarrow (\text{false } y \text{ true}) \rightarrow$$

$$\rightarrow (\lambda x. \lambda y. y) y \text{ true} \rightarrow \text{true}$$

$$\text{op } \vdash \text{true} \rightarrow p \Rightarrow q \equiv p \vee q$$

$$\Downarrow$$

$$p \vdash \text{false} \vee y \Rightarrow y$$

$$p \vdash \text{true} \vee y \Rightarrow \text{true}$$

# REȘINTEZĂ DE TIP:

1)  $f :: xs \rightarrow head\ xs\ xs.$

Prin înțelegerea puterilor de la pașii cu  $f$  nuia const., de la cu. gen.

Apoi nuă uit la const., la tipuri:

init:  $f :: a \rightarrow b$

cu  $head\ xs \Rightarrow x \times tb$  să fie o listă  $\Rightarrow xs :: [t_1]; \underline{a = [t_1]}$

mai mult decât e val de nu am mai head  $xs \Rightarrow$

$= head\ xs\ xs = (head\ xs)\ xs$

primul el din listă să-l aplic pe  $xs \Rightarrow$

$\Rightarrow$  el din  $xs$  tb să fie  $f$ .

el din  $xs$  genit de tipul  $t_1 \Rightarrow t_1 \equiv t_2 \rightarrow t_3.$

problema  $xs \Rightarrow t_2 \equiv [t_1]$

$\Rightarrow t_1 \equiv [t_1] \rightarrow t_3 \equiv [[t_1] \rightarrow t_3] \rightarrow t_3 \rightarrow$  eror de sintaxă de tip.

definiție ciclică:  $t_1$  e mai, cu cu. ce dep de  $t_1$

$g\ xs = head\ xs\ 2$  a uit  $xs$  prin 6 pro m cu de tip Number

2)  $f\ x\ y = (head\ x, head\ y) \rightarrow f\ (tail\ x)\ (tail\ y)$

putere de la cu. gen: cons adăugăm în plus lista  $f \dots$

$f :: a \rightarrow b \rightarrow c$   $\rightarrow$  putere de la cu. gen în care să obț. const.

$\downarrow$   
 $x\ y$

$head\ x \Rightarrow x :: [t_1] \Rightarrow a \equiv [t_1]$  nu' lasă cu oclor tip.

ibidem pt  $b: head\ y = b \equiv [t_2]$

Rez e listă de pașii:  $\Rightarrow [t_1, t_2] \rightarrow$  merge sau

$\downarrow$

$f :: [t_1] \rightarrow [t_2] \rightarrow [t_1, t_2].$

nu poate fi dată  
 $[t_1, t_2] \times = t_1 \equiv t_2$

eror de sint de tip pt a se of la const de listă si pua de  $[ ] \Rightarrow head$  er  
din p. de val. do pb.

eror de tip nu val. p. de  
eror de tip, p. de tip din tip

3)  $f \ x \ y = \text{if } x == \text{head } y \text{ then } [x] \text{ else } f \ x (\text{tail } y)$

$\downarrow$   $x$  to so imp. param.  $E_2$  at as am ==

$f :: a \rightarrow b \rightarrow c$

$\text{head } y \Rightarrow y = \text{lists} \Rightarrow y :: [t_1] \Rightarrow b = [t_1]$

$x == \text{head } y \Rightarrow x :: t_1 + \text{cons } E_2 \ t_1$

post deduce cons despr c?

pl b2ul de b2020, vz lui  $f$  este o listă cu  $x \Rightarrow c \equiv [t_1] \equiv b$ .

$\text{tail } y = \text{tail } [\text{lists}] \Rightarrow f :: E_2 \ t_1 \Rightarrow t_1 \rightarrow [t_1] \rightarrow [t_1]$

4)  $g \ f \ (x, y) = x++ \text{map } f \ y$   
 $\text{map}$

$g :: a \rightarrow (b, c) \rightarrow d$

$\text{map } f \ y \Rightarrow f \text{ este o func } \Rightarrow f \equiv t_1 \rightarrow t_2$

$y \text{ e o listă} \Rightarrow y = [t_1] \equiv c$

$t_2$  b2020  $f$  de tipul lui  $x$  și cum apare  $t$  și o listă  $\Rightarrow$

$\Rightarrow x++ \uparrow x$  listă cu el din listă not map  $\Rightarrow x :: [t_2]$ .

$\Rightarrow g :: (t_1 \rightarrow t_2) \rightarrow ([t_2], [t_1]) \rightarrow [t_2]$ .

## III. INSTANTIERE

1) Suprînsc. în Hs de comp pe listă o.i. o listă se  $f$   $\Leftarrow$  este o listă de  $f$  și el e prima  $<$  el e o listă.

$[2, 1, 1] < [5, 1, 3]$  dar nu  $[2, 1, 1] < [3, 5]$

pp.  
 Aș putea suprînsc. op. de comp listă

Sau  $\left( \begin{array}{l} \text{filter pe } f \text{ are o listă primă } 1 \text{ care } f \text{ este } 1 \text{ și o listă opri} \\ \text{length, opri unde } \text{length} \text{ de } f \text{ filter} = \text{length listă} \end{array} \right.$

dar cum  $f$  suprînsc.  $\rightarrow$  just total b2020 b2020 op. op.

$\Rightarrow$  instance Ord a  $\Rightarrow$  Ord (a) where  $\rightarrow$  o b2020  $f$  de ord. pt e b2020  
 $x_1 < x_2 = \text{maximum } x_1 < \text{minimum } x_2$  compar func de el  
 din listă.

Ord cu listă de op  $\rightarrow$  cons.

2) Find a good inst = class

class MyClass c where

$f :: c \rightarrow a$

instance MyClass  $\xrightarrow[\text{lists for each}]{\text{constructors}}$  c e de tip  $\rightarrow$  lista

$f = \text{head}$

-- a e tipul de din ocl c

-- on the prim lista e o-in si int un a,  $\Rightarrow$  head

3) exprs.  $sp + = 11$ ,  $* = 48$  pt val bool:

closo cu +  $48 = \underline{\text{Num}}$   $\rightarrow$  cl pt cas in ss for inst.

= instance Num Bool where

$\downarrow$   
tip pt cas in ss for inst.

-- Zic ce se interp pt (+) & (\*)

-- gen cu o de fit

c de o  $\rightarrow$  fit de date pascu  
in fct de a  $\Rightarrow$  Maybe de lista

(+) = (11)  $\rightarrow$  de fit sau

(\*) = (48) -- nu te complia, ai facut ce ti s-a cerut

Sau tie  
nu  
de date

-- am pus bool pt o in ss for pt val boolean

4) generalize lui filter pt  $sp^1$  & construc un ce utilizez tipul a

filter ::  $a \rightarrow \text{Bool} \rightarrow [a] \rightarrow [a]$

de tip Maybe a = Just a | Nothing

class Filterable c where -- for each class + zic trend de date

filter' ::  $(a \rightarrow \text{Bool}) \rightarrow \mathbb{R} \rightarrow a \rightarrow \mathbb{R} \rightarrow a$  (nu zic ca e pe lista, ci un

construc un ce util. pt a)

b) instance Filterable [ ] where

filter' f xs = [x | x <- xs, f x]

-- obis acum interper: e list comprehension or cu filter' = filter (post

c) -- similar pt Maybe, gen explic de sorte mpk conol for filter pe cele 2 variante

## IV LOGICA CU PREDICATE

den. univ., rez & R.A.

1)  $\forall x. (P(x) \Rightarrow Q(x))$  este corect. Ex.  $\exists y. P(y) \Rightarrow \exists z. Q(z)$

c) aducem o primă prop la formă clauze

$\sim$  de  $\Rightarrow$  avem  $\neg P \vee Q$

privește înțelesul explicațiile din curs + ex. cu pozi  
elime implic

imping neg înainte de ot.

restul variab

deci nu sunt învinse lui  $\Rightarrow$  înlocuim cu  $\forall$  și  $\exists$  în

locul variab care depinde de alt

— vezi PDF

$$\forall x. (P(x) \Rightarrow Q(x)) \quad \left\{ \begin{array}{l} P \Rightarrow Q \equiv \neg P \vee Q \\ \Downarrow \\ \forall x. (\neg P(x) \vee Q(x)) \end{array} \right.$$

remuând la implicare

câte clauze are?

UNA

cu  $\neg P(x)$  și  $Q(x)$

$\Rightarrow$  clauze:  $\{\neg P(x), Q(x)\}$

b). neg o prop + ad. la formă clauzele

$\exists y. P(y) \Rightarrow \exists z. Q(z) \Rightarrow$  prima dată o neg, prin  $\neg \exists \equiv \forall$

$\Rightarrow \neg (\exists y. P(y) \Rightarrow \exists z. Q(z))$  (apoi de implicare)  $\Rightarrow$

$\Rightarrow \neg (\neg (\exists y. P(y)) \vee \exists z. Q(z)) \rightarrow$  dist. termenilor fals  $\Rightarrow$

$\Rightarrow \neg (\neg (\exists y. P(y)) \wedge \neg (\exists z. Q(z)))$   $\neg (P \vee Q) \equiv \neg P \wedge \neg Q$

$\hookrightarrow$  se duce  $\Rightarrow \exists y. P(y) \wedge \forall z. \neg Q(z) \Rightarrow$  pot surd. not  $\Rightarrow$

$\Downarrow$   
 $\neg \exists \equiv \forall$  / câte clauze are? 1)  $\exists y. P(y)$  & 2)  $\forall z. \neg Q(z)$

clauze:  $\{P(y), \neg Q(z)\}$ ,  $C = \emptyset$ .

c) Rezolvare pe contabile sub ord.

$$(1): \{ \neg P(x), Q(x) \}$$

$$(2): \{ P(c) \}$$

$$(3): \{ \neg Q(z) \}$$

(1) & (2) unificabile doar dacă

folosim o substituție  $\Rightarrow x \leftarrow c \Rightarrow$  înlocuim  $Q \Rightarrow$

$$\Rightarrow (4) \{ Q(c) \}$$

definit = înlocuim

Ce contabile unificabile? Ca să poată să unifice, ele trebuie să

conțin

Aplicăm una pe lângă alta:

$$\{ p_1, \dots, x_1, \dots, p_m \}$$

$$\{ z_1, \dots, \neg x_1, \dots, \neg z_m \}$$

$$\{ p_1, \dots, p_m, z_1, \dots, z_m \} \text{ fără } R \text{ și fără } \neg$$

$$(1) \& (3): \neg P(x) (u)$$

$$(4) \& (2) \neg$$

se ține predicatul fals  
← caracteristică

contabile

(3) și (4) unificabile; din nou, o nouă substituție  $\Rightarrow z \leftarrow c \Rightarrow \emptyset$

2) Cine învătă, nu merge de fapt. merge (cine, forme)

am spus R.A. ca  
ceea

Într-un învătă  $\Rightarrow \forall \text{ Cine. învătă}(\text{Cine}) \Rightarrow \exists \text{ merge}(\text{Cine}, \text{forme})$ .

Cine = variabilă, forme = ct;  
sau =

$\forall \text{ Cine. învătă}(\text{Cine}) \Rightarrow \exists \text{ merge}(\text{Cine}, \text{forme})$ .

merge (cine, forme)  $\rightarrow$  merge am a săi fac.

$\exists \text{ învătă}(\text{cine})$

prel.  $\forall x. \text{învătă}(x) \Rightarrow \exists \text{ merge}(x, \text{forme}) \Rightarrow \text{deine} \Rightarrow$

$\Rightarrow \forall x (\neg \text{învătă}(x) \vee \exists \text{ merge}(x, \text{forme})) \Rightarrow$  alina  $\Rightarrow$

$\Rightarrow$  Clauza:  $\{ \neg \text{învătă}(x), \exists \text{ merge}(x, \text{forme}) \} (1)$

$\{ \text{merge}(\text{cine}, \text{forme}) \} (2)$

neg concl.  $\Rightarrow \{ \neg \text{învătă}(\text{cine}) \} (3)$

(1) cu (2) unificabile  $\Rightarrow x \leftarrow \text{cine} \Rightarrow \{ \neg \text{învătă}(\text{cine}) \} (4)$

(3) cu (4)  $\Rightarrow \neg \text{învătă}(\text{cine})$

de primul cauzari de  
fapt să puz

Haskell: documentație  
de sintaxă de tip,  
instanțiere cu  $\varphi$ ,  
implem.

implem predicate  
la Prolog  
predicate intermed  
ph liste min,  
max,  
ultimul, primul  
etc.

uzabili ex. exm. printate  
în subiect