Pregătire PP – Seria CC (2023)

I. Calcul lambda

1. Fie λ-expresia de mai jos:

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

   (a) (4p) Completați fraza: Pentru a evalua expresia utilizând modelul bazat pe **substituție textuală**, trebuie înlocuite aparițiile (libere/legate?) _(1)_ ale variabilei _(2)_ din expresia _(3)_ cu expresia _(4)_.

   (b) (3p) Care este rezultatul evaluării și de ce nu este corect?

   (c) (3p) Ce pas lipsește pentru a obține rezultatul corect? Scrieți rezultatul corect.

1) libere 2) variabila x 3) $\lambda y.(x\ \ y)$ 4) y

$$(\underline{\lambda x}.\underbrace{\lambda y.(x\ \ y)}\ \ y) \to_\beta \lambda y.(y\ \ y)$$

apariție liberă

apariție liberă

De ce nu este corect?

Trebuie să aplic α-conversie pt. a evita problema:

$$(\lambda x.\underline{\lambda y}.(x\ \ \underline{y})\ \ y) \to_\alpha (\underline{\lambda x}.\underbrace{\lambda z.(x\ \ z)}\ \ y) \to_\beta$$

$$\to_\beta \lambda z.(y\ \ z)$$

apariție liberă

2) $(\lambda x.(\underline{\lambda x}.x\ \ x)\ \ a) \to_\beta (\underline{\lambda x}.x\ \ a) \to_\beta a$

$(\underline{\lambda x}.\underbrace{(\lambda x.x\ \ x)}\ \ a) \to (\underline{\lambda x}.x\ \ a) \to a$

3) $((\underline{\lambda y}.\underbrace{(\lambda x.\lambda y.x\ \ y)}\ \ \lambda x.x)\ \ \Omega) \to_\beta$

$((\underline{\lambda x}.\lambda y.x\ \ \lambda x.x)\ \ \Omega) \to_\beta (\underline{\lambda y}.\lambda x.x\ \ \Omega) \to_\beta \lambda x.x$

4) $(\underline{\lambda y}.\underbrace{((\lambda x.\lambda y.x\ \ y)\ \ \Omega)}\ \ \boxed{\lambda x.\lambda y.y}) \to_\beta$

$$((\lambda x.\lambda y.x \quad \boxed{\lambda x.\lambda y.y})\ \Omega) \to_\beta$$

$$(\lambda y.\lambda x.\lambda y.y \quad \Omega) \to_\beta \lambda x.\lambda y.y$$

5) $$(((\lambda x.\lambda y.\lambda z.y \quad \boxed{\lambda x.x})\ (\lambda z.\lambda t.z \quad z))\ \Omega) \to_\beta$$

$$\to ((\lambda y.\lambda z.y \quad (\lambda z.\lambda t.z \quad z))\ \Omega) \to_\alpha$$

liber

$$\to_\beta (\lambda z.(\lambda z.\lambda t.z \quad z)\ \Omega)$$ (nu pot aplica direct)

legat

$$\to_\alpha ((\lambda y.\lambda w.y\ (\lambda z.\lambda t.z \quad z))\ \Omega) \to_\beta$$

$$\to_\beta (\lambda w.(\lambda z.\lambda t.z \quad z)\ \Omega) \to_\beta$$

$$(\lambda z.\lambda t.z \quad z) \to_\beta \lambda t.z$$

$0 := \lambda f.\lambda x.x \qquad succ := \lambda n.\lambda f.\lambda x(f \quad ((n\ f) \quad x))$

$(succ\ 0)$?

$$(\lambda n.\lambda f.\lambda x(f \quad ((n\ f) \quad x)) \quad \boxed{\lambda f.\lambda x.x}) \to$$

$$\to \lambda f.\lambda x(f \quad ((\lambda f.\lambda x.x \quad f) \quad x)) \to_\beta$$

$$\lambda f.\lambda x(f \quad (\lambda x.x \quad x)) \to_\beta \lambda f.\lambda x(f \quad x)$$

---

1. (λx y z.corp a b c) este forma prescurtată de a scrie (((λx.λy.λz.corp a) b) c).

Fie următoarele definiții în Calcul Lambda:

true = λx y.x

false = λx y.y

$true = \lambda x.\lambda y.x$

$false = \lambda x.\lambda y.y$

și fie operatorul logic:

op = λx y.(x y true)

Descrieți pas cu pas (nu efectuați mai multe β-reduceri deodată) comportamentul lui op pe valori de tip true sau/și false, astfel încât să puteți conchide la final că op se comportă ca unul dintre operatorii logici cunoscuți (și precizați care este operatorul a cărui identitate a fost ascunsă prin folosirea numelui "op").

$op = \lambda x.\lambda y.(x\ y\ true)$

$(op\ true\ y) = y$

$(\lambda x.\lambda y.(x\ y\ true)\ true\ y) \to (true\ y\ true)$

$(\lambda x.\lambda y.x\ y\ true) \to y$

$(op\ false\ y) = true$

$(\lambda x.\lambda y.(x\ y\ true)\ false\ y) \to (false\ y\ true)$

$(\lambda x.\lambda y.y\ y\ true) \to true$

$op \equiv \text{"} \Rightarrow \text{"}$ $\qquad P \Rightarrow Q \equiv \neg P \lor Q$

II Sinteză de tip

$f\ xs = head\ xs\ xs$

$f :: a \to b$

$head\ xs \Rightarrow xs$ trebuie să fie o listă $\Rightarrow xs :: [t_1]$

$a \equiv [t_1]$

$head\ xs\ xs = (head\ xs)\ xs \Rightarrow$ elementele din xs trebuie să fie funcții $\Rightarrow$

$\Rightarrow t_1 \equiv t_2 \to t_3$

$t_2 \equiv [t_1]$ $\Big\} \Rightarrow$

$\Rightarrow t_1 \equiv [t_1] \to t_3 \equiv [[t_1] \to t_3] \to t_3$

$\Rightarrow$ Eroare de sinteză de tip!

$g\ xs = head\ xs\ 2 \to$ Corect definită!

$f\ x\ y = (head\ x, head\ y) : f\ (tail\ x)\ (tail\ y)$

$f :: a \to b \to c$

$head\ x \Rightarrow x$ este listă $\Rightarrow a \equiv [t_1]$

head $y \Rightarrow b \equiv [t_2]$

Rez. o să fie tot o listă $\Rightarrow [(t_1, t_2)]$

$f :: [t_1] \to [t_2] \to [(t_1, t_2)]$

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

$f :: a \to b \to c$

head $y \Rightarrow y$ este listă $\Rightarrow y :: [t] \Rightarrow b \equiv [t]$

$x ==$ head $y \Rightarrow x :: t$ + const. Eq t

Pe cazul de bază rez lui f este $[x] \Rightarrow c \equiv [t]$

$f :: \text{Eq}\ t \Rightarrow t \to [t] \to [t]$

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

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

map $f\ y \Rightarrow f$ este o funcție $\Rightarrow f \equiv t_1 \to t_2$
y este o listă $\Rightarrow c \equiv [t_1]$

$x\ ++ \ldots \Rightarrow x$ este listă și el. din x au același tip cu cele din lista rez după aplicarea lui map $\Rightarrow$
$\Rightarrow x :: [t_2]$

$g :: (t_1 \to t_2) \to ([t_2], [t_1]) \to [t_2]$

III. Clase în Haskell

6. Supraîncărcați în Haskell operatorul de comparație pe liste, astfel încât o listă să fie mai mică sau egală cu alta, dacă toate elementele din prima listă sunt mai mici sau egale cu toate elementele din a doua. Spre exemplu, `[2, 3, 1] <= [5, 4, 3]`, dar nu avem că `[2, 4] <= [3, 5]`.

instance Ord a => Ord [a] where
    x1 <= x2 = maximum x1 <= minimum x2

6. Evidențiați o posibilă instanță a clasei Haskell de mai jos:

```
class MyClass c where
  f :: c a -> a
```

instance MyClass [] where
    f = head

6. Supraîncărcați în Haskell operatorii (+) și (*) pentru valori booleene, pentru a surprinde operațiile de *sau*, respectiv *și* logic.

instance Num Bool where
    (+) = (||)
    (*) = (&&)

6. Generalizați funcționala `filter` din Haskell, având tipul `(a -> Bool) -> [a] -> [a]`, pentru a opera pe orice constructor de tip unar ce utilizează valori de tipul `a`, nu numai pe constructorul listă.

   (a) (3p) **Definiți** clasa `Filterable`, parametrizată cum considerați de cuviință, în care să definiți funcția `filter'`, cu tipul adecvat.

   (b) (2p) **Instanțiați** `Filterable`, definită mai sus, cu constructorul listă standard.

   (c) (5p) **Instanțiați** `Filterable`, definită mai sus, cu constructorul de tip `Maybe`, definit prin

```
1  data Maybe a = Just a | Nothing
```

a) class Filterable c where
    filter' :: (a -> Bool) -> c a -> c a

b) instance Filterable [] where
    filter' f xs = [x | x <- xs, f x]
        sau
        filter' = filter

IV Logica cu predicate

7. Demonstrați utilizând rezoluția și reducerea la absurd că propoziția

$$\forall x.(P(x) \Rightarrow Q(x))$$

are drept consecință logică propoziția

$$\exists y.P(y) \Rightarrow \exists z.Q(z).$$

(a) (2p) Aduceți prima propoziție la forma clauzală.

(b) (5p) Negați a doua propoziție și aduceți rezultatul la forma clauzală.

(c) (3p) Aplicați rezoluția pe clauzele obținute anterior.

a)

Conversia propozițiilor în FNC (1)

$P \vee \overline{P}$

Eliminare implicații, împingere negații, redenumiri

1. Eliminarea implicațiilor ($\not\Rightarrow$)
2. Împingerea negațiilor până în fața atomilor ($\overset{\rightarrow}{\neg}$)
3. Redenumirea variabilelor cuantificate pentru obținerea unicității de nume (R):
$$\forall x.p(x) \wedge \forall x.q(x) \vee \exists x.r(x) \;\rightarrow\; \forall x.p(x) \wedge \forall y.q(y) \vee \exists z.r(z)$$
4. Deplasarea cuantificatorilor la începutul expresiei, conservându-le ordinea (forma normală *prenex*) (P):
$$\forall x.p(x) \wedge \forall y.q(y) \vee \exists z.r(z) \;\rightarrow\; \forall x.\forall y.\exists z.(p(x) \wedge q(y) \vee r(z))$$

Conversia propozițiilor în FNC (2)

$P \vee \overline{P}$

Skolemizare

5. Eliminarea cuantificatorilor existențiali (skolemizare) (S):
   - Dacă nu este precedat de cuantificatori universali: înlocuirea aparițiilor variabilei cuantificate printr-o constantă (bine aleasă):
   $$\exists x.p(x) \;\rightarrow\; p(c_x)$$
   - Dacă este precedat de cuantificatori universali: înlocuirea aparițiilor variabilei cuantificate prin aplicația unei funcții unice asupra variabilelor anterior cuantificate universal:
   $$\forall x.\forall y.\exists z.((p(x) \wedge q(y)) \vee r(z))$$
   $$\rightarrow \forall x.\forall y.((p(x) \wedge q(y)) \vee r(f_z(x,y)))$$

Conversia propozițiilor în FNC (3)

$P \vee \overline{P}$

Cuantificatori universali, Distribuire $\vee$, Clauze

6. Eliminarea cuantificatorilor universali, considerați, acum, impliciți ($\not\forall$):
$$\forall x.\forall y.(p(x) \wedge q(y) \vee r(f_z(x,y))) \;\rightarrow\; p(x) \wedge q(y) \vee r(f_z(x,y))$$
7. Distribuirea lui $\vee$ față de $\wedge$ ($\vee/\wedge$):
$$\alpha \vee (\beta \wedge \gamma) \;\rightarrow\; (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$
8. Transformarea expresiilor în clauze (C).

$\forall x (P(x) \Rightarrow Q(x))$ $\qquad$ $P \Rightarrow Q \equiv \neg P \vee Q$

$\forall x . (\neg P(x) \vee Q(x)) \Rightarrow \neg P(x) \vee Q(x)$

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

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

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

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

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

$\underline{\exists y . P(y)} \wedge \underline{\forall z . \neg Q(z)}$

Clauze: $\{P(c)\}$ $\quad$ $\{\neg Q(z)\}$

$c =$ constantă

- Forma generală a pasului de rezoluție:

$$\frac{\{p_1, \ldots, r, \ldots, p_m\} \quad \{q_1, \ldots, \neg r, \ldots, q_n\}}{\{p_1, \ldots, p_m, q_1, \ldots, q_n\}}$$

c) Aplic rezoluția

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

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

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

1 și 2 unifică utilizând $x \leftarrow c \Rightarrow$
$\Rightarrow \{Q(c)\}$ (4)

4 și 3 unifică utilizând $z \leftarrow c \Rightarrow$
$\square$

8. Știind că *Cine învață, nu moare de foame* și că $moareDe(Ion, foame)$, demonstrați **prin metoda rezoluției** că *Ion nu învață*.

"Cine învață, nu moare de foame"

$\forall Cine . învață(Cine) \Rightarrow \neg moareDe(Cine, foame)$ $\quad$ cst

Cine = var

$moareDe(Ion, foame)$

"Ion nu învață"

$\neg$ învață (Ion)

Dem. prin rez:

$\forall x.$ învață $(x) \Rightarrow \neg$ moare de $(x,$ foame$)$

$\forall x.$ $(\neg$ învață $(x) \lor \neg$ moare de $(x,$ foame$))$

Clauza: $\{\neg$ învață $(x), \neg$ moare de $(x,$ foame$)\}$ 1

$\{$moare de (Ion, foame)$\}$ 2

Neg concluzia: $\{$învață (Ion)$\}$ 3

1 și 2 cu subst. $X \leftarrow$ Ion $\Rightarrow \{\neg$ învață (Ion)$\}$ 4

3 și 4 $\Rightarrow \square$