Nyelvek típusrendszere vizsga (B csoport)

2024. december 30.

`Név és Neptun kód: ____________________________________________________________________________`

Ott van minden feladat mellett, hogy mennyi pontot ér. Alapértelmezetten minden feladat többszörös választás (nulla vagy több megoldást kell bekarikázni). Ha nem az, akkor oda van írva. Összesen 70 pont.

1 Programozási nyelv

A vizsgában az összes kérdés az alábbi programozási nyelvre vonatkozik. Ez a System F, szabályai az alábbiak (ugyanazok, mint előadáson).

```
Ty      : Set
Tm      : Ty → Set
_⇒_     : Ty → Ty → Ty
lam     : (Tm A → Tm B) → Tm (A ⇒ B)
_·_     : Tm (A ⇒ B) → Tm A → Tm B
⇒β      : lam t · a = t a
⇒η      : (f : Tm (A ⇒ B)) → f = lam (λx.f·x)
∀       : (Ty → Ty) → Ty
Lam     : ((A : Ty) → Tm (F A)) → Tm (∀ F)
_•_     : Tm (∀ F) → (A : Ty) → Tm (F A)
∀β      : Lam t • A = t A
∀η      : (f : Tm (∀ F)) → f = Lam (λX.f•X)
```

2 Lexing, parsing

Az alábbi lexikális elemeket használjuk: `lam`, `·`, `Lam`, `•`, `.`, `(`, `)`, `X`, `Y`, `Z`, …, `x`, `y`, `z`, … A típusozatlan AST-k az alábbi módon vannak megadva:

```
TyVar := String        Ty  : Set                  Tm  : Set
TmVar := String        var : TyVar → Ty           var : TmVar → Tm
                       _⇒_ : Ty → Ty → Ty         lam : TmVar → Tm → Tm
                       ∀   : TyVar → Ty → Ty      _·_ : Tm → Tm → Tm
                                                  Lam : TyVar → Tm → Tm
                                                  _•_ : Tm → Ty → Tm
```

1. (1 pont) Tekintsük az alábbi sztringet:

   `Lam X . Lam Y . (lam x . lam . x`

   Ebből a sztringből egy lexer melyik lexikális elem-sorozatot adja végeredményül? (egyszeres választás)

   - [`Lam`, `X`, `.`, `Lam`, `Y`, `.`, `(`, `lam`, `x`, `.`, `lam`, `.`, `x`, `)`]
   - [`Lam`, `X`, `.`, `Lam`, `Y`, `.`, `(`, `lam`, `x`, `.`, `x`, `)`]
   - [`Lam`, `X`, `.`, `Lam`, `Y`, `.`, `(`, `lam`, `x`, `.`, `lam`, `.`, `x`]
   - [`Lam`, `X`, `.`, `Lam`, `Y`, `.`, `lam`, `x`, `.`, `lam`, `.`, `x`]
   - [`Lam`, `X`, `.`, `Lam`, `Y`, `.`, `lam`, `x`, `.`, `x`]
   - Egyiket sem, mert nincs jól zárójelezve a kifejezés.

- Egyiket sem, mert a `lam` után mindig változónak kell szerepelnie, és a második `lam` után nincs változó.

2. (1 pont) Az alábbi fát hogyan írjuk le típusozatlan AST-ként? (egyszeres választás)

```
      _·_
     /   \
   Lam    lam
   / \     / \
  X  lam  x   var
     /  \      |
    x   var    x
         |
         x
```

- `(lam x . var x) · (lam x . var x)`
- `(Lam X . lam x . var x) · (lam x . var x)`
- `Lam X . ((lam x . var x) · lam x . var x)`
- `Lam X . lam x . lam x . var x · var x`
- Nem írható le Agdában, mert a `_·_` első paramétere `⇒` típusú, a `Lam` viszont `∀` típusú.

3. (2 pont) Rajzold le, hogy az alábbi sztringből melyik absztrakt szintaxisfát hozza létre a parser (az előző feladat fordítottja)!

```
(lam x . var x) · ((lam x . var x) · (lam x . var x))
```

3 Típuskikövetkeztetés

Az implicit paramétereket kapcsos zárójelekkel jelöljük, például `lam {A}{B} : (Tm A → Tm B) → Tm (A ⇒ B)`, a következő esetben csak az első implicit paramétert adjuk meg: `lam {A} (λx.x) : Tm (A ⇒ A)`.

A típustörlés `Tm A` egy eleméből egy típusozatlan AST-t hoz létre. Például `Lam λX.Lam λY.lam {X}{Y⇒X} λx.lam {Y}{X} λy.x`-ból `Lam X . Lam Y . lam x . lam y . var x` lesz. A típuskikövetkeztetés a típustörlés ellentéte: egy típusozatlan AST-ből egy `A` típust és `Tm A` egy elemét hozza létre.

4. (2 pont) Jelöld be azokat az `A` típusokat, melyekhez létezik `t : Tm A`, hogy `t` típustörlése a `lam x . var x` típusozatlan AST-t eredményezi.

- `(∀ λX.X) ⇒ (∀ λX.X)`
- `(∀ λX.X ⇒ X) ⇒ (∀ λX.X)`
- `(∀ λX.X⇒X⇒X) ⇒ (∀ λX.X⇒X⇒X)`
- `(∀ λX.X⇒X) ⇒ (∀ λX.X⇒X)`
- `(∀ λX.X) ⇒ (∀ λX.X⇒X)`
- `(∀ λX.(X⇒X)⇒X) ⇒ (∀ λX.(X⇒X)⇒X)`

5. (2 pont) Az alábbiakban felsorolunk típusozatlan AST-ket. Egy típuskikövetkeztető ezek közül melyiket utasítja biztosan vissza? (Tehát mely AST-khez nem létezik olyan `A : Ty` és `t : Tm A`, hogy `t` típustörlése az AST-t adja vissza.)

   - `lam x . var x`
   - `lam x . lam y . var x · var y`
   - `lam x . lam y . var y · var y`
   - `lam x . lam y . var y · var x`
   - `lam x . lam y . var y · (var y · var x)`
   - `lam x . var x · var x`
   - `(lam x . var x) · (lam y . var y)`
   - Minden AST-hez létezik `A : Ty` és `t : Tm A`, hogy `t` típustörlése az AST-t adja vissza.

6. (6 pont) Az alábbiakban felsorolunk jól típusozott termeket a nyelvünkben. Írd melléjük a típusukat! Az elsőt megadtuk példaként.

```
Lam λX.lam {X} λx.x                                 : Tm (∀ λX.X ⇒ X)

Lam λX.lam {X ⇒ X} λf.lam {X} λx.f · (f · x)        : ________________________________

lam {∀ λX.X} λx.x                                   : ________________________________

Lam λX.lam {X} λx.Lam λ Y.lam {Y} λy.y              : ________________________________

Lam λX.lam {X} λx.lam {X} λy.x                      : ________________________________

Lam λX.lam {X ⇒ X} λf.lam {X ⇒ X} λg.f              : ________________________________

Lam λX.lam {X} λx.lam {X} λy.lam {X} λz.x           : ________________________________
```

7. (5 pont) Írj tetszőleges adott típusú termet, ha van! Ha nincs, írd oda, hogy nincs! Az elsőt megadtuk példaként.

```
Lam λX.lam λx.x                                : Tm (∀ λX.X ⇒ X)

______________________________________________ : Tm (∀ λX.X ⇒ (X ⇒ X))

______________________________________________ : Tm (∀ λX.(X ⇒ X) ⇒ X)

______________________________________________ : Tm (∀ λX.∀ λY.X ⇒ Y ⇒ X)

______________________________________________ : Tm (∀ λX.X)

______________________________________________ : Tm (∀ λX.∀ λY.∀ λZ.(X ⇒ Y) ⇒ (Y ⇒ Z) ⇒ X ⇒ Z)
```

4 További típusok megadása

A `Three` típust megadjuk az alábbi módon.

```
Three := ∀ λX.X ⇒ X ⇒ X ⇒ X
```

8. (4 pont) Add meg a `one`, `two` operátorokat úgy, hogy teljesüljenek a β szabályok, és bizonyítsd be a `Threeβ₃` szabályt egyenlőségi érveléssel! A `zero` operátort és az iterátort megadtuk.

```
zero : Tm Three

zero := Lam λX.lam λx.lam λy.lam λz.x

one : Tm Three

one := ______________________________________________________

two : Tm Three

two := ______________________________________________________

iteThree : Tm Three → Tm A → Tm A → Tm A → Tm A

iteThree t u v w := t • A · u · v · w

Threeβ₁ : iteThree zero u v w
                                                          =(definíció)
          (Lam λX.lam λx.lam λy.lam λz.x) • A · u · v · w
                                                          =(∀β)
          (lam λx.lam λy.lam λz.x) · u · v · w
                                                          =(⇒β)
          (lam λy.lam λz.u) · v · w
                                                          =(⇒β)
          (lam λz.u) · w
                                                          =(⇒β)
          u

Threeβ₂ : iteThree one u v w = v

Threeβ₃ : iteThree two u v w









          w
```

A szorzat típusokat az alábbi módon kódoljuk:

```
_×_ : Ty → Ty → Ty
A × B := ∀ λC.(A ⇒ B ⇒ C) ⇒ C
```

9. (5 pont) Add meg a szorzat típus konstruktorát, destruktorait és bizonyítsd be az egyik β szabályt! (A másiknak is teljesülnie kell, de nem kell levezetni.)

```
_,_ : Tm A → Tm B → Tm (A × B)

a , b := ________________________________________
```

```
fst : Tm (A × B) → Tm A

fst w := ________________________________________

snd : Tm (A × B) → Tm B

snd w := ________________________________________

×β₁ : fst (a , b)









      a

×β₂ : snd (a , b) = b
```

A `Three` és a szorzat segítségével elkódoljuk a homogén ternáris összeg típust az alábbi módon:

```
3*_ : Ty → Ty
3*_ := Three × A
```

10. (6 pont) Add meg a konstruktorokat és a destruktort, és bizonyítsd az egyik β szabályt! (A többinek is teljesülnie kell.)

```
inj₁ : Tm A → Tm (3* A)

inj₁ u := __________________________________________________

inj₂ : Tm A → Tm (3* A)

inj₂ u := ___________________________________________________

inj₃ : Tm A → Tm (3* A)

inj₃ u := ___________________________________________________

case : Tm (3* A) → (Tm A → Tm C) → (Tm A → Tm C) → (Tm A → Tm C) → Tm C

case w u₁ u₂ u₃ := ___________________________________________________

3*β₁ : case (inj₁ w) u₁ u₂ u₃









       u₁ w
```

```
3*β₂ : case (inj₂ w) u₁ u₂ u₃ = u₂ w

3*β₃ : case (inj₃ w) u₁ u₂ u₃ = u₃ w
```

11. (5 pont) Add meg, hogy az alábbi típusoknak hány elemük van! Az elsőt megadtuk példaként.

```
∀ λX.X ⇒ X                                  1

∀ λX.X ⇒ X ⇒ X                              ___

∀ λX.(X ⇒ X) ⇒ (X ⇒ X)                      ___

∀ λX.(X ⇒ X ⇒ X) ⇒ (X ⇒ X)                  ___

∀ λX.(X ⇒ X) ⇒ X                            ___

∀ λX.X                                      ___

∀ λX.X ⇒ X ⇒ X ⇒ X ⇒ X                      ___

∀ λX.∀ λY.X⇒Y                               ___
```

Elkódoljuk a létezik típusoperátort, a konstruktorát, destruktorát és a β szabályát az alábbi módon.

```
∃   : (Ty → Ty) → Ty
∃ F := ∀ λC.(∀ λX.F X ⇒ C) ⇒ C
mk∃ : (X : Ty) → Tm (F X) → Tm (∃ F)
mk∃ X w := Lam λC.lam λu.u•X·w
un∃ : ((X : Ty) → Tm (F X) → Tm C) → Tm (∃ F) → Tm C
un∃ f u := u•C·(Lam λX.lam λw.f X w)
∃β  : un∃ f (mk∃ X w)
                                                        =(def)
      (Lam λC.lam λu.u•X·w)•C·(Lam λX.lam λw.f X w)
                                                        =(∀β)
      (lam λu.u•X·w)·(Lam λX.lam λw.f X w)
                                                        =(⇒β)
      (Lam λX.lam λw.f X w)•X·w
                                                        =(∀β)
      (lam λw.f X w)·w
                                                        =(⇒β)
      f X w
```

12. (8 pont) Kódold el a végtelen mély bináris fák típusát, melyek a csomópontoknál `A`-t tartalmaznak! Add meg a három destruktort (amely kiveszi az `A`-t, megadja a fa bal illetve jobb oldali gyerekét), a generátorát, és bizonyítsd be a `Treeβ₁` és `Treeβ₃` szabályokat (`Treeβ₂` is teljesüljön)!

```
Tree  : Ty → Ty


Tree A  := ____________________________________

item  : Tm (Tree A) → Tm A


item  w := ____________________________________

left  : Tm (Tree A) → Tm (Tree A)


left  w := ____________________________________

right : Tm (Tree A) → Tm (Tree A)


right w := ____________________________________
```

```
gen   : (Tm C → Tm A) → (Tm C → Tm C) → (Tm C → Tm C) → Tm C → Tm (Tree A)

gen {C} i l r c := _______________________________________________________________

Treeβ₁ : item (gen i l r c)
```

```
         i c

Treeβ₂ : left (gen i l r c) = gen i l r (l c)

Treeβ₃ : right (gen i l r c) =
```

```
         gen i l r (r c)
```

13. (5 pont) Add meg, hogy melyik típushoz melyik definíció tartozik! Kettőt megadtunk példaként. Az utolsónál használhatod a természetes számokat, rövidítsd `Nat`-nak!

```
természetes számok (Nat)                             ∀ λC.(C⇒C)⇒C⇒C

véges mélységű bináris fák extra információ nélkül   ____________________________________

véges mélységű bináris fák, levélnél Three egy eleme ____________________________________

kételemű típus                                       ____________________________________

háromfelé ágazó végtelen mélységű fák,               ∃ λC.(C ⇒ C × C × C) × (C ⇒ A) × C
csomópontoknál A

_____________________________________________________ ∃ λC.(C ⇒ C) × (C ⇒ A) × C

Végtelenfelé ágazó végtelen mélységű fák,
csomópontoknál Three egy eleme                       ____________________________________
```

5 Programozás

Most a korábban megadott típusokat használjuk programozásra. Az alábbiakban használd a `Tree` és `Three` típusok konstruktorait és destruktorait közvetlenül, nem kell a Church-kódolást kifejteni (akkor is megoldhatók ezek a feladatok, ha nem sikerült megoldani a 8. és a 12. feladatokat).

14. (3 pont) Add meg a háromelemű típusra a modulo 3 rákövetkezőt! Tehát egy olyan `suc : Tm Three ⇒ Tm Three` függvényt, melyre `suc zero = one`, `suc one = two`, `suc two = zero`.

15. (2 pont) Add meg azt a végtelen mélységű bináris fát, amely mindenhol `one`-t tartalmaz (`Tm (Tree Three)` egy eleme)!

16. (3 pont) Add meg azt a végtelen mélységű bináris fát, amelynek gyökerénél `zero` van, következő szintjén mindenhol `one` van, következő szintjén mindenhol `two`, következő szintjén mindenhol `zero`, következő szintjén mindenhol `one`, és így tovább!

17. (4 pont) Mutasd meg, hogy ha `zero = one`, akkor minden `u, v : Tm A`-ra `u = v`!

18. (3 pont) Igaz-e, hogy minden típusnak van eleme a nyelvünkben? Indoklást is kérünk.

19. (3 pont) Belátható-e, hogy minden `t : Tm (A ⇒ A)`-ra `lam (λ x . t · x) = t`? Ha igen, vezesd le! Ha nem, indokold!