Fundamentele limbajelor de programare

C05

Denisa Diaconescu
Traian Șerbănuță

Departamentul de Informatică, FMI, UB

Lambda calcul cu tipuri simple

Probleme cu lambda calculul fără tipuri

Proprietăți negative ale lambda calculului fără tipuri:

- Aplicări de forma $x\,x$ sau $M\,M$ sunt pemise, deși sunt contraintuitive.
- Existența formelor normale pentru λ-termeni nu este garantată și putem avea "calcule infinite" nedorite
- Orice λ-termen are un punct fix ceea ce nu este în armonie cu ceea ce știam despre funcții oarecare

Vrem să eliminăm aceste proprietăți negative, păstrându-le pe cele pozitive.

Proprietățile negative sunt eliminate prin adăugarea de tipuri ceea ce induce restricțiile dorite pe termeni.

Tipuri simple

Fie $\mathbb{V} = \{\alpha, \beta, \gamma, \ldots\}$ o mulțime infinită de tipuri variabilă.

Mulțimea tuturor tipurilor simple $\mathbb{T}$ este definită prin

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T}$$

- (Tipul variabilă) Dacă $\alpha \in \mathbb{V}$, atunci $\alpha \in \mathbb{T}$.
- (Tipul săgeată) Dacă $\sigma, \tau \in \mathbb{T}$, atunci $(\sigma \to \tau) \in \mathbb{T}$.

Câteodată vom nota tipurile simple și cu litere $A, B, \ldots$.

Tipurile variabilă sunt reprezentări abstracte pentru tipuri de bază cum ar fi *Nat* pentru numere naturale, *List* pentru liste etc.

Tipurile săgeată reprezintă tipuri pentru funcții cum ar fi

- $Nat \to Real$, mulțimea tuturor funcțiilor de la numere naturale la numere reale
- $(Nat \to Int) \to (Int \to Nat)$, mulțimea tuturor funcțiilor care au ca intrare o funcție de la numere naturale la întregi și produce o funcție de la întregi la numere naturale.

Tipuri simple

Mulțimea tipurilor simple $\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T}$

Exemple de tipuri simple:

- γ
- $(\beta \to \gamma)$
- $((\gamma \to \alpha) \to (\alpha \to (\beta \to \gamma)))$

În tipurile săgeată, parantezele exterioare pot fi omise.

Parantezele în tipurile săgeată sunt asociative la dreapta.

De exemplu,

- $\alpha_1 \to \alpha_2 \to \alpha_3 \to \alpha_4$ este abreviere pentru $(\alpha_1 \to (\alpha_2 \to (\alpha_3 \to \alpha_4)))$
- $x_1\ x_2\ x_3\ x_4$ este abreviere pentru $(((x_1\ x_2)\ x_3)\ x_4)$

Termeni și tipuri

Ce înseamnă că un termen M are un tip σ?

Vom nota acest lucru cu $M : \sigma$.

Termeni și tipuri

Ce înseamnă că un termen M are un tip σ?

Vom nota acest lucru cu $M : \sigma$.

Variabilă. Dacă o variabilă x are un tip σ, notăm cu $x : \sigma$.

Convenția Barendregt: variabilele legate sunt distincte.

Presupunem că orice variabilă din M are un unic tip.

Dacă $x : \sigma$ și $x : \tau$, atunci $\sigma \equiv \tau$.

Termeni și tipuri

Ce înseamnă că un termen M are un tip σ?

Vom nota acest lucru cu $M : \sigma$.

Variabilă. Dacă o variabilă x are un tip σ, notăm cu $x : \sigma$.

Convenția Barendregt: variabilele legate sunt distincte.

Presupunem că orice variabilă din M are un unic tip.

Dacă $x : \sigma$ și $x : \tau$, atunci $\sigma \equiv \tau$.

Aplicare. Pentru $M\,N$ este clar că vrem să știm tipurile lui M și N.

Intuitiv, $M\,N$ înseamnă că ("funcția") M este aplicată ("intrării") N.

Atunci M trebuie să aibă un tip funcție, adică $M : \sigma \to \tau$, iar N trebuie să fie "adecvat" pentru această funcție, adică $N : \sigma$.

$$\text{Dacă } M : \sigma \to \tau \text{ și } N : \sigma, \text{ atunci } M\,N : \tau.$$

Termeni și tipuri

Ce înseamnă că un termen M are un tip σ?

Vom nota acest lucru cu $M : \sigma$.

Variabilă. Dacă o variabilă x are un tip σ, notăm cu $x : \sigma$.

Convenția Barendregt: variabilele legate sunt distincte.

Presupunem că orice variabilă din M are un unic tip.

Dacă $x : \sigma$ și $x : \tau$, atunci $\sigma \equiv \tau$.

Aplicare. Pentru $M\,N$ este clar că vrem să știm tipurile lui M și N.

Intuitiv, $M\,N$ înseamnă că ("funcția") M este aplicată ("intrării") N.

Atunci M trebuie să aibă un tip funcție, adică $M : \sigma \to \tau$, iar N trebuie să fie "adecvat" pentru această funcție, adică $N : \sigma$.

Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\,N : \tau$.

Abstractizare. Dacă $M : \tau$, ce tip trebuie sa aibă $\lambda x.\,M$?

Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\,M : \sigma \to \tau$.

Termeni și tipuri

Variabilă. $x : \sigma$.

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\,N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\,M : \sigma \to \tau$.

M *are tip* (este *typeable*) dacă există un tip σ astfel încât $M : \sigma$.

Termeni și tipuri

Variabilă. $x : \sigma$.

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\, N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\, M : \sigma \to \tau$.

M *are tip* (este *typeable*) dacă există un tip σ astfel încât $M : \sigma$.

Exemple.

- Dacă $x : \sigma$, atunci funcția identitate are tipul $\lambda x.\, x : \sigma \to \sigma$.

Termeni și tipuri

Variabilă. $x : \sigma$.

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\,N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\, M : \sigma \to \tau$.

M *are tip* (este *typeable*) dacă există un tip σ astfel încât $M : \sigma$.

Exemple.

- Dacă $x : \sigma$, atunci funcția identitate are tipul $\lambda x.\, x : \sigma \to \sigma$.
- Conform convențiilor de la aplicare, $y\,x$ poate avea un tip doar dacă y are un tip săgeată de forma $\sigma \to \tau$ și tipul lui x se potrivește cu tipul domeniu σ. În acest caz, tipul lui $y\,x : \tau$.

Termeni și tipuri

Variabilă. $x : \sigma$.

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\,N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\,M : \sigma \to \tau$.

M *are tip* (este *typeable*) dacă există un tip σ astfel încât $M : \sigma$.

Exemple.

- Dacă $x : \sigma$, atunci funcția identitate are tipul $\lambda x.\,x : \sigma \to \sigma$.
- Conform convențiilor de la aplicare, $y\,x$ poate avea un tip doar dacă y are un tip săgeată de forma $\sigma \to \tau$ și tipul lui x se potrivește cu tipul domeniu σ. În acest caz, tipul lui $y\,x : \tau$.
- Termenul $x\,x$ nu poate avea nici un tip (nu este typeable). Pe de o parte, x ar trebui să aibă tipul $\sigma \to \tau$ (pentru prima apariție), pe de altă ar trebui să aibă tipul σ (pentru a doua apariție). Cum am stabilit că orice variabilă are un unic tip, obținem $\sigma \to \tau \equiv \sigma$, ceea ce este imposibil.

Discuție despre asocitativitate

Asociativitatea la dreapta pentru tipurile săgeată vs. asociativitatea la stânga pentru aplicare:

- Să presupunem că $f : \rho \to (\sigma \to \tau)$, $x : \rho$ și $y : \sigma$.
- Atunci $f\,x : \sigma \to \tau$ și $(f\,x)\,y : \tau$.

Discuție despre asocitativitate

Asociativitatea la dreapta pentru tipurile săgeată vs. asociativitatea la stânga pentru aplicare:

- Să presupunem că $f : \rho \to (\sigma \to \tau)$, $x : \rho$ și $y : \sigma$.
- Atunci $f\,x : \sigma \to \tau$ și $(f\,x)\,y : \tau$.
- Folosind ambele convenții pentru asociativitate pentru a elimina parantezele, avem

  $$f : \rho \to \sigma \to \tau$$
  $$f\,x\,y : \tau$$

  Convențiile pentru asociativitate sunt în armonie una cu cealaltă.

Church-typing vs. Curry-typing

A găsi tipul unui termen începe cu a găsi tipurile pentru variabile. Există două metode prin care putem asocia tipuri variabilelor.

Asociere explicită *(Church-typing)*.

- Constă în prescrierea unui unic tip pentru fiecare variabilă, la introducerea acesteia.
- Presupune că tipurile variabilelor sunt explicit stabilite.
- Tipurile termenilor mai complecși se obțin natural, ținând cont de convențiile pentru aplicare și abstractizare.

Asociere implicită *(Curry-typing)*.

- Constă în a nu prescriere un tip pentru fiecare variabilă, ci în a le lăsa "deschise" (implicite).
- În acest caz, termenii *typeable* sunt descoperiți printr-un proces de căutare, care poate presupune "ghicirea" anumitor tipuri.

Church-typing vs. Curry-typing

Exemplu. Asociere explicită *(Church-typing)*.

Vrem să calculăm tipul expresiei $(\lambda zu.\, z)\,(y\, x)$ știind că

1. $x : \alpha \to \alpha$
2. $y : (\alpha \to \alpha) \to \beta$
3. $z : \beta$
4. $u : \gamma$

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\, N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\, M : \sigma \to \tau$.

Church-typing vs. Curry-typing

Exemplu. Asociere explicită *(Church-typing)*.

Vrem să calculăm tipul expresiei $(\lambda zu.\, z)\,(y\, x)$ știind că

1. $x : \alpha \to \alpha$
2. $y : (\alpha \to \alpha) \to \beta$
3. $z : \beta$
4. $u : \gamma$

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\, N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\, M : \sigma \to \tau$.

Din (2) și (1), prin aplicare obținem (5): $y\, x : \beta$.

Din (4) și (3), prin abstractizare obținem (6): $\lambda u.\, z : \gamma \to \beta$.

Din (3) și (6), prin abstractizare obținem (7): $\lambda zu.\, z : \beta \to \gamma \to \beta$.
Nu uitați că $\beta \to \gamma \to \beta$ înseamnă $\beta \to (\gamma \to \beta)$.

Atunci, din (7) și (5), prin aplicare, avem $(\lambda zu.\, z)\,(y\, x) : \gamma \to \beta$.

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)*.

Considerăm termenul de mai devreme $M = (\lambda zu.\, z)\,(y\,x)$.

Putem să "ghicim" tipurile variabilelor astfel încât M să *aibă tip*?

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\,N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\, M : \sigma \to \tau$.

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)*.

Considerăm termenul de mai devreme $M = (\lambda zu.\, z)\,(y\, x)$.

Putem să "ghicim" tipurile variabilelor astfel încât M să *aibă tip*?

Aplicare. Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $M\,N : \tau$.

Abstractizare. Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.\, M : \sigma \to \tau$.

- Observăm că M este o aplicare a lui $\lambda zu.\, z$ termenului $y\, x$.
- Atunci $\lambda zu.\, z$ trebuie să aibă un tip săgeată, de exemplu $\lambda zu.\, z : A \to B$, și $y\, x$ să se potrivească, adică $y\, x : A$.
- În acest caz, avem $M : B$.

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)* (cont.)

Știm $M = (\lambda zu.\, z)\,(y\,x)$ și am dedus până acum:

$$\lambda zu.\, z : A \to B \qquad y\,x : A \qquad M : B$$

- Faptul că $\lambda zu.\, z : A \to B$ implică că $z : A$ și $\lambda u.\, z : B$.
- Deducem că B este tipul unei abstractizări, deci $B \equiv C \to D$, și obținem că $u : C$ și $z : D$.
- Pe de altă parte, $y\,x$ este o aplicare, deci trebuie să existe E și F astfel încât $y : E \to F$ și $x : E$. Atunci $y\,x : F$.

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)* (cont.)

Știm $M = (\lambda zu.\, z)\,(y\, x)$. Am dedus următoarele:

- $x : E$
- $y : E \to F$
- $z : A$ și $z : D$, deci $A \equiv D$
- $u : C$
- $B \equiv C \to D$
- $y\, x : A$ și $y\, x : F$, deci $A \equiv F$.

În concluzie, $A \equiv D \equiv F$, și eliminând redundanțele obținem

$$(*) \qquad x : E \qquad y : E \to A \qquad z : A \qquad u : C$$

Reamintim că aveam $M : B$, adică $M : C \to A$.

Am obținut o schemă generală $(*)$ pentru tipurile lui x, y, z, u care induc un tip pentru M.

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)* (cont.)

Știm $M = (\lambda zu.\, z)\,(y\, x)$. Am obținut schema generală

$$(*) \quad x : E \quad y : E \to A \quad z : A \quad u : C \quad M : C \to A$$

În schema de mai sus, putem considera tipuri "reale":

- $x : \beta,\ \ y : \beta \to \alpha,\ \ z : \alpha,\ \ u : \delta,\ \ M : \delta \to \alpha$

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)* (cont.)

Știm $M = (\lambda zu.\, z)\,(y\, x)$. Am obținut schema generală

$$(*) \quad x : E \quad y : E \to A \quad z : A \quad u : C \quad M : C \to A$$

În schema de mai sus, putem considera tipuri "reale":

- $x : \beta$, $y : \beta \to \alpha$, $z : \alpha$, $u : \delta$, $M : \delta \to \alpha$
- $x : \alpha \to \alpha$, $y : (\alpha \to \alpha) \to \beta$, $z : \beta$, $u : \gamma$, $M : \gamma \to \beta$
  (soluția discutată la Church-typing)

Church-typing vs. Curry-typing

Exemplu. Asociere implicită *(Curry-typing)* (cont.)

Știm $M = (\lambda zu.\, z)\,(y\, x)$. Am obținut schema generală

$$(*) \quad x : E \quad y : E \to A \quad z : A \quad u : C \quad M : C \to A$$

În schema de mai sus, putem considera tipuri "reale":

- $x : \beta, \quad y : \beta \to \alpha, \quad z : \alpha, \quad u : \delta, \quad M : \delta \to \alpha$
- $x : \alpha \to \alpha, \quad y : (\alpha \to \alpha) \to \beta, \quad z : \beta, \quad u : \gamma, \quad M : \gamma \to \beta$
  (soluția discutată la Church-typing)
- $x : \alpha, \quad y : \alpha \to \alpha \to \beta, \quad z : \alpha \to \beta, \quad u : \alpha \to \alpha,$
  $M : (\alpha \to \alpha) \to \alpha \to \beta$

Church-typing vs. Curry-typing

Asocierea implicită de tipuri *(Curry-typing)* are proprietăți interesante, cum am văzut în exemplul anterior.

Totuși, în continuare vom folosi asocierea explicită *(Church-typing)* deoarece de obicei tipurile sunt cunoscute dinainte (și declararea tipurilor pentru argumentele unei funcții este o bună-practică).

Marcăm tipurile variabilelor legate imediat după introducerea lor cu o abstractizare. Tipurile variabilelor libere sunt date de un context.

Church-typing

Exemplu. Să considerăm exemplul anterior $(\lambda zu.\, z)\,(y\, x)$.

Observați că z și u sunt legate, iar x și y sunt libere.

Presupunând că $z : \beta$ și $u : \gamma$, scriem termenul astfel

$$(\lambda z : \beta.\, \lambda u : \gamma.\, z)\,(y\, x)$$

Church-typing

Exemplu. Să considerăm exemplul anterior $(\lambda zu.\, z)\,(y\, x)$.

Observați că z și u sunt legate, iar x și y sunt libere.

Presupunând că $z : \beta$ și $u : \gamma$, scriem termenul astfel

$$(\lambda z : \beta.\, \lambda u : \gamma.\, z)\,(y\, x)$$

Dacă presupunem un context în care despre variabilele libere știm, de exemplu, că $x : \alpha \to \alpha$ și $y : (\alpha \to \alpha) \to \beta$, atunci folosim notația:

$$x : \alpha \to \alpha, y : (\alpha \to \alpha) \to \beta \vdash (\lambda z : \beta.\, \lambda u : \gamma.\, z)\,(y\, x)$$

Church-typing

Exemplu. Să considerăm exemplul anterior $(\lambda zu.\, z)\,(y\, x)$.

Observați că z și u sunt legate, iar x și y sunt libere.

Presupunând că $z : \beta$ și $u : \gamma$, scriem termenul astfel

$$(\lambda z : \beta.\, \lambda u : \gamma.\, z)\,(y\, x)$$

Dacă presupunem un context în care despre variabilele libere știm, de exemplu, că $x : \alpha \to \alpha$ și $y : (\alpha \to \alpha) \to \beta$, atunci folosim notația:

$$x : \alpha \to \alpha, y : (\alpha \to \alpha) \to \beta \vdash (\lambda z : \beta.\, \lambda u : \gamma.\, z)\,(y\, x)$$

Încă nu avem o noțiune de β-reducție pentru termeni cu tipuri, dar ne-am putea gândi că am avea:

$$(\lambda z : \beta.\, \lambda u : \gamma.\, z)\,(y\, x) \to_\beta \lambda u : \gamma.\, y\, x.$$

Observați că am dori să deducem că $(\lambda u : \gamma.\, y\, x) : \gamma \to \beta$.

Sistem de deducție pentru Church $\lambda\!\to$

Deoarece am convenit cum să decorăm cu informații despre tipuri variabilele legate, trebuie să actualizăm definiția λ-termenilor.

Mulțimea λ-termenilor cu pre-tipuri $\Lambda_{\mathbb{T}}$ este

$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\, \Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\, \Lambda_{\mathbb{T}}$$

O afirmație este o expresie de forma $M : \sigma$, unde $M \in \Lambda_{\mathbb{T}}$ și $\sigma \in \mathbb{T}$. Într-o astfel de afirmație, M se numește subiect și σ tip.

O declarație este o afirmație în care subiectul este o variabilă ($x : \sigma$).

Un context este o listă de declarații cu subiecți diferiți.

O judecată este o expresie de forma $\Gamma \vdash M : \sigma$, unde Γ este context și $M : \sigma$ este o afirmație.

Sistem de deducție pentru Church $\lambda{\rightarrow}$

Deoarece suntem în general interesați de termeni *typeable*, am dori să avem o metodă prin care să putem stabili dacă un termen $t \in \Lambda_{\mathbb{T}}$ este *typeable* și dacă da, să calculăm un tip pentru t.

Vom da niște reguli care să ne permită să stabilim dacă o judecată $\Gamma \vdash M : \sigma$ poate fi dedusă, adică dacă M are tipul σ în contextul Γ.

Sistem de deducție pentru calculul Church $\lambda\!\rightarrow$

$$\frac{}{\Gamma \vdash x : \sigma} \text{ dacă } x : \sigma \in \Gamma \; (\mathit{var})$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau} \; (\mathit{app})$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.\, M) : \sigma \to \tau} \; (\mathit{abs})$$

Un termen M în calculul $\lambda\!\rightarrow$ este legal dacă există un context Γ și un tip ρ astfel încât $\Gamma \vdash M : \rho$.

Sistem de deducție pentru calculul Church $\lambda{\to}$

$$\frac{}{\Gamma \vdash x : \sigma}\ (var) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (app) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.\, M) : \sigma \to \tau}\ (abs)$$

dacă $x : \sigma \in \Gamma$

Exemplu. Să arătăm că termenul $\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz$ are tipul $(\alpha \to \beta) \to \alpha \to \beta$ în contextul vid.

$$\emptyset \vdash (\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz) : (\alpha \to \beta) \to \alpha \to \beta$$

Sistem de deducție pentru calculul Church $\lambda{\rightarrow}$

$$\frac{}{\Gamma \vdash x : \sigma}\ (var) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (app) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.\, M) : \sigma \to \tau}\ (abs)$$

dacă $x : \sigma \in \Gamma$

Exemplu. Să arătăm că termenul $\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz$ are tipul $(\alpha \to \beta) \to \alpha \to \beta$ în contextul vid.

$$\emptyset \vdash (\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz) : (\alpha \to \beta) \to \alpha \to \beta$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y : \alpha \to \beta, z : \alpha \vdash y : \alpha \to \beta}\ (var) \qquad \dfrac{}{y : \alpha \to \beta, z : \alpha \vdash z : \alpha}\ (var)}{y : \alpha \to \beta, z : \alpha \vdash (yz) : \beta}\ (app)}{y : \alpha \to \beta \vdash (\lambda z : \alpha.\, yz) : \alpha \to \beta}\ (abs)}{\emptyset \vdash (\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz) : (\alpha \to \beta) \to \alpha \to \beta}\ (abs)$$

Diferite stiluri pentru a scrie deducții

În exemplul anterior, am scris derivarea în stilul arbore.

În stilul liniar, derivarea precedentă ar arăta astfel:

1.	$y : \alpha \to \beta, z : \alpha \vdash y : \alpha \to \beta$	(var)
2.	$y : \alpha \to \beta, z : \alpha \vdash z : \alpha$	(var)
3.	$y : \alpha \to \beta, z : \alpha \vdash (yz) : \beta$	(app) cu 1 și 2
4.	$y : \alpha \to \beta \vdash (\lambda z : \alpha.\, yz) : \alpha \to \beta$	(abs) cu 3
5.	$\emptyset \vdash (\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz) : (\alpha \to \beta) \to \alpha \to \beta$	(abs) cu 4

Diferite stiluri pentru a scrie deducții

În stilul cu cutii, afișăm fiecare declarație la începutul unei cutii și considerăm că declarația respectivă face parte din contextul pentru toate afirmațiile din cutia respectivă.

Când închidem o cutie, abstractizăm după variabila din declarația de la începutul cutiei.

$$(\lambda y : \alpha \to \beta .\, \lambda z : \alpha .\, y z) : (\alpha \to \beta) \to \alpha \to \beta$$

Diferite stiluri pentru a scrie deducții

În stilul cu cutii, afișăm fiecare declarație la începutul unei cutii și considerăm că declarația respectivă face parte din contextul pentru toate afirmațiile din cutia respectivă.

Când închidem o cutie, abstractizăm după variabila din declarația de la începutul cutiei.

$$\boxed{\begin{array}{ll} y : \alpha \to \beta & (\textit{context}) \\ (\lambda z : \alpha.\, yz) : \alpha \to \beta & \end{array}}$$

$$(\lambda y : \alpha \to \beta.\, \lambda z : \alpha.\, yz) : (\alpha \to \beta) \to \alpha \to \beta \quad (\textit{abs})$$

Diferite stiluri pentru a scrie deducții

În stilul cu cutii, afișăm fiecare declarație la începutul unei cutii și considerăm că declarația respectivă face parte din contextul pentru toate afirmațiile din cutia respectivă.

Când închidem o cutie, abstractizăm după variabila din declarația de la începutul cutiei.

$$
\begin{array}{|ll}
\hline
y : \alpha \to \beta & (\mathit{context}) \\
\begin{array}{|ll}
\hline
z : \alpha & (\mathit{context}) \\
(yz) : \beta & \\
\hline
\end{array} & \\
(\lambda z : \alpha .\, yz) : \alpha \to \beta & (\mathit{abs}) \\
\hline
\end{array}
$$

$$(\lambda y : \alpha \to \beta .\, \lambda z : \alpha .\, yz) : (\alpha \to \beta) \to \alpha \to \beta \quad (\mathit{abs})$$

Diferite stiluri pentru a scrie deducții

În stilul cu cutii, afișăm fiecare declarație la începutul unei cutii și considerăm că declarația respectivă face parte din contextul pentru toate afirmațiile din cutia respectivă.

Când închidem o cutie, abstractizăm după variabila din declarația de la începutul cutiei.

1.	$y : \alpha \to \beta$	($context$)
2.	$z : \alpha$	($context$)
3.	$(yz) : \beta$	(app) cu 1 și 2
4.	$(\lambda z : \alpha .\, yz) : \alpha \to \beta$	(abs) cu 3
5.	$(\lambda y : \alpha \to \beta .\, \lambda z : \alpha .\, yz) : (\alpha \to \beta) \to \alpha \to \beta$	(abs) cu 4

Sistem de deducție pentru Church $\lambda{\rightarrow}$

Exercițiu. Arătați că termenul $\lambda x : ((\alpha \to \beta) \to \alpha).\, x\, (\lambda z : \alpha.\, y)$ are tipul $((\alpha \to \beta) \to \alpha) \to \alpha$ în contextul $y : \beta$.

Quiz time!

https://tinyurl.com/2p9xf67e

Pe săptămâna viitoare!