Fundamentele limbajelor de programare

C06

Denisa Diaconescu
Traian Șerbănuță

Departamentul de Informatică, FMI, UB

Lambda calcul cu tipuri simple (cont.)

Tipuri simple

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

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

Mulțimea λ-termenilor cu pre-tipuri $\Lambda_{\mathbb{T}}$ $\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\, \Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\, \Lambda_{\mathbb{T}}$

- O afirmație este o expresie de forma $M : \sigma$, unde $M \in \Lambda_{\mathbb{T}}$ și $\sigma \in \mathbb{T}$.
- O declarație este o afirmație de forma $x : \sigma$.
- Un context Γ este o listă de declarații cu subiecți diferiți.
- O judecată este o expresie de forma $\Gamma \vdash M : \sigma$.

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

$$\frac{}{\Gamma \vdash x : \sigma}\ (var) \quad \text{dacă } x : \sigma \in \Gamma \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (\to_E) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.\, M) : \sigma \to \tau}\ (\to_I)$$

Un termen M în calculul $\lambda\!\to$ este legal dacă $\Gamma \vdash M : \rho$.

Ce probleme putem să rezolvăm în teoria tipurilor?

Type Checking

Se reduce la a verifica că putem găsi o derivare pentru

$$context \vdash term : type$$

Ce probleme putem să rezolvăm în teoria tipurilor?

Well-typedness (Typability)

Se reduce la a verifica dacă un termen este legal. Concret, trebuie să găsim un context și un tip dacă termenul este legal, altfel să arătăm de ce nu se poate.

$$? \vdash \mathit{term} : ?$$

O variațiune a problemei este *Type Assignment* în care contextul este dat și trebuie să găsim tipul.

$$\mathit{context} \vdash \mathit{term} : ?$$

Ce probleme putem să rezolvăm în teoria tipurilor?

Term Finding (Inhabitation)

Dându-se un context și un tip, să stabilim dacă există un termen cu acel tip, în contextul dat.

$$\mathit{context} \vdash ? : \mathit{type}$$

Toate aceste probleme sunt decidabile pentru calculul Church $\lambda\!\!\rightarrow$!

Limitări ale lambda-calculului cu tipuri simple

Nu mai avem recursie nelimitată deoarece combinatorii de punct fix nu sunt *typeable*.

De exemplu, $\mathbf{Y} \triangleq \lambda y.\,(\lambda x.\,y\,(x\,x))\,(\lambda x.\,y\,(x\,x))$ nu este typeable.

Dar avem recursie primitivă (recursie care permite doar *looping* în care numărul de iterații este cunoscut dinainte).

De exemplu, $\mathbf{add} \triangleq \lambda mnfx.\,m\,f\,(n\,f\,x)$ este o funcție primitiv recursivă.

Faptul că orice evaluare se termină este important pentru implementări ale logicilor folosind lambda-calculul.

Limitări ale lambda-calculului cu tipuri simple

Tipurile pot fi prea restrictive.

De exemplu, am putea gândi că termenul $(\lambda f.\, \textbf{if}\, (f\, \textbf{T})\, (f\, 3)\, (f\, 5))\, (\lambda x.\, x)$ ar trebui să aibă un tip. Dar nu are!

Soluții posibile:

- Let-polymorphism unde variabilele libere din tipul lui f se redenumesc la fiecare folosire. De exemplu, am putea scrie

$$\textbf{let}\, f = \lambda x.\, x\, \textbf{in}$$
$$\textbf{if}\, (f\, \textbf{T})\, (f\, 3)\, (f\, 5)$$

- Cuantificatori de tipuri. De exemplu, am avea

$$\lambda x.\, x : \Pi\alpha\, .\, \alpha \to \alpha$$

  Operatorul de legare Π face explicit faptul că variabila de tip α nu este rigidă.

Alte tipuri

Tipul Unit și constructorul unit

Mulțimea tipurilor

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

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

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

$$\frac{}{\Gamma \vdash \text{unit} : \texttt{Unit}}\ (\mathit{unit})$$

Tipul Void

Mulțimea tipurilor

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \texttt{Unit} \mid \texttt{Void}$$

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

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

Nu există regulă de tipuri pentru deoarece tipul `Void` nu are inhabitant.

Tipul produs și constructorul pereche

Mulțimea tipurilor

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \texttt{Unit} \mid \texttt{Void} \mid \mathbb{T} \times \mathbb{T}$$

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

$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\, \Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\, \Lambda_{\mathbb{T}} \mid \text{unit} \mid \langle \Lambda_{\mathbb{T}}, \Lambda_{\mathbb{T}} \rangle$$

$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau}\ (\times_I)$$

Tipul produs și constructorul pereche

Mulțimea tipurilor

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \texttt{Unit} \mid \texttt{Void} \mid \mathbb{T} \times \mathbb{T}$$

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

$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\, \Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\, \Lambda_{\mathbb{T}} \mid \text{unit} \mid \langle \Lambda_{\mathbb{T}}, \Lambda_{\mathbb{T}} \rangle \mid \mathit{fst}\ \Lambda_{\mathbb{T}} \mid \mathit{snd}\ \Lambda_{\mathbb{T}}$$

$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau}\ (\times_I)$$

$$\frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \mathit{fst}\ M : \sigma}\ (\times_{E_1}) \qquad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \mathit{snd}\ M : \tau}\ (\times_{E_2})$$

Tipul sumă și constructorii Left/Right

Mulțimea tipurilor

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid \texttt{Unit} \mid \texttt{Void} \mid \mathbb{T} \times \mathbb{T} \mid \mathbb{T} + \mathbb{T}$$

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

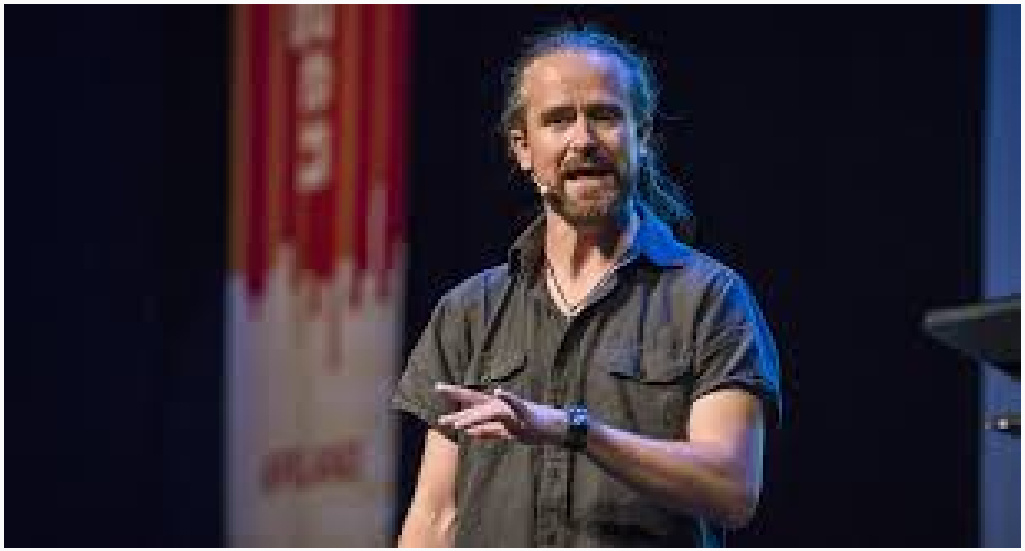
$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\, \Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\, \Lambda_{\mathbb{T}} \mid \text{unit} \mid \langle \Lambda_{\mathbb{T}}, \Lambda_{\mathbb{T}} \rangle \mid \mathit{fst}\ \Lambda_{\mathbb{T}} \mid \mathit{snd}\ \Lambda_{\mathbb{T}}$$
$$\mid \text{Left}\ \Lambda_{\mathbb{T}} \mid \text{Right}\ \Lambda_{\mathbb{T}} \mid \text{case}\ \Lambda_{\mathbb{T}}\ \text{of}\ \Lambda_{\mathbb{T}}\ ;\ \Lambda_{\mathbb{T}}$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \text{Left}\ M : \sigma + \tau}\ (+_{I_1}) \qquad \frac{\Gamma \vdash M : \tau}{\Gamma \vdash \text{Right}\ M : \sigma + \tau}\ (+_{I_2})$$

$$\frac{\Gamma \vdash M : \sigma + \tau \qquad \Gamma \vdash M_1 : \sigma \to \gamma \qquad \Gamma \vdash M_2 : \tau \to \gamma}{\Gamma \vdash \text{case}\ M\ \text{of}\ M_1\ ;\ M_2 : \gamma}\ (+_E)$$

Corespondența Curry-Howard

Schimbați perspectiva



Roger Antonsen

Universitatea din Oslo

TED Talk: Math is the hidden secret to understanding the world

"... înțelegerea constă în abilitatea de a-ți schimba perspectiva"

https://www.ted.com/talks/roger_antonsen_math_is_the_hidden_secret_to_understanding_the_world

Un program simplu în Haskell

```haskell
data Point = Point Int Int

makePoint :: Int -> Int -> Point
makePoint x y = Point x y

getX :: Point -> Int
getX (Point x y) = x

getY :: Point -> Int
getY (Point x y) = y

origin :: Point
origin = makePoint 0 0
```

Un program simplu în Haskell

Hai să schimbăm perspectiva!

```
data Point = Point Int Int

makePoint :: Int -> Int -> Point
makePoint x y = Point x y

getX :: Point -> Int
getX (Point x y) = x

getY :: Point -> Int
getY (Point x y) = y
```

$$\frac{x : Int \quad y : Int}{makePoint\ x\ y : Point}\ (Point_I)$$

$$\frac{p : Point}{getX\ p : Int}\ (Point_{E_1})$$

$$\frac{p : Point}{getY\ p : Int}\ (Point_{E_2})$$

Generalizare

$$\frac{x : \mathit{Int} \quad y : \mathit{Int}}{\mathit{makePoint}\ x\ y : \mathit{Point}}\ (\mathit{Point}_I) \qquad \frac{M : \sigma \quad N : \tau}{\langle M, N \rangle : \sigma \times \tau}\ (\times_I)$$

$$\frac{p : \mathit{Point}}{\mathit{getX}\ p : \mathit{Int}}\ (\mathit{Point}_{E_1}) \qquad \frac{M : \sigma \times \tau}{\mathit{fst}\ M : \sigma}\ (\times_{E_1})$$

$$\frac{p : \mathit{Point}}{\mathit{getY}\ p : \mathit{Int}}\ (\mathit{Point}_{E_2}) \qquad \frac{M : \sigma \times \tau}{\mathit{snd}\ M : \tau}\ (\times_{E_2})$$

Generalizare

$$\frac{x : Int \quad y : Int}{makePoint\ x\ y : Point}\ (Point_I)$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau}\ (\times_I)$$

$$\frac{M : Point}{getX\ M : Int}\ (Point_{E_1})$$

$$\frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash fst\ M : \sigma}\ (\times_{E_1})$$

$$\frac{M : Point}{getY\ M : Int}\ (Point_{E_2})$$

$$\frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash snd\ M : \tau}\ (\times_{E_2})$$

Alt exemplu simplu

```
f = (\x -> x * 3) :: Int -> Int
```

$$\frac{\{x : Int\} \vdash x * 3 : Int}{\lambda x.x * 3 : Int \to Int}\ (fun_I)$$

```
> f 5
15
```

$$\frac{f : Int \to Int \quad 5 : Int}{f\ 5 : Int}\ (fun_E)$$

Generalizare

$$\frac{\{x : Int\} \vdash x * 3 : Int}{\lambda x.x * 3 : Int \to Int}\ (fun_I)$$

$$\frac{\{x : \sigma\} \vdash M : \tau}{\lambda x.M : \sigma \to \tau}\ (\to_I)$$

$$\frac{f : Int \to Int \quad 5 : Int}{f\ 5 : Int}\ (fun_E)$$

$$\frac{M : \sigma \to \tau \quad N : \sigma}{M\ N : \tau}\ (\to_E)$$

Generalizare

$$\frac{\{x : Int\} \vdash x * 3 : Int}{\lambda x.x * 3 : Int \to Int}\ (fun_I)$$

$$\frac{\Gamma \cup \{x : \sigma\} \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to_I)$$

$$\frac{f : Int \to Int \quad 5 : Int}{f\ 5 : Int}\ (fun_E)$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (\to_E)$$

Logica. Ce este adevăt și ce este fals?

Hai să schimbăm perspectiva iar!

Logica. Ce este adevăt și ce este fals?

Hai să schimbăm perspectiva iar!

Dacă afară este întuneric atunci,
dacă porcii zboară atunci este întuneric afară.

σ = afară este întuneric
τ = porcii zboară

$$\sigma \supset (\tau \supset \sigma)$$

Logica. Ce este adevăt și ce este fals?

Hai să schimbăm perspectiva iar!

Dacă afară este întuneric atunci,
dacă porcii zboară atunci este întuneric afară.

σ = afară este întuneric
τ = porcii zboară

$$\sigma \supset (\tau \supset \sigma)$$

Este adevărată această afirmație? Da!

σ	τ	$\tau \supset \sigma$	$\sigma \supset (\tau \supset \sigma)$
false	false	true	true
false	true	false	true
true	false	true	true
true	true	true	true

Semantica unei logici

Dăm valori variabilelor în mulțimea $\{0, 1\}$,
definim o evaluare $e : V \to \{0, 1\}$.

Putem să o extindem o evaluare la formule:

$\wedge : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$

σ	τ	$\sigma \wedge \tau$
0	0	0
0	1	0
1	0	0
1	1	1

$\supset : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$

σ	τ	$\sigma \supset \tau$
0	0	1
0	1	1
1	0	0
1	1	1

Dacă pentru toate evaluările posibile, o formulă are valoarea 1,
atunci spunem că este o tautologie.

Sintaxa unei logici

Dăm metode pentru a manipula simbolurile din logică (i.e., $\supset$, $\wedge$) pentru a stabili când o formulă este demonstrabilă/teoremă .

Corectitudine	=	sintaxa implică semantica
Completitudine	=	sintaxa și semantica coincid

Un sistem de deducție naturală

Reguli pentru a manevra fiecare conector logic (introducerea si eliminarea conectorilor).

$$\frac{\Gamma \vdash \sigma \quad \Gamma \vdash \tau}{\Gamma \vdash \sigma \wedge \tau}\ (\wedge_I) \qquad \frac{\Gamma \vdash \sigma \wedge \tau}{\Gamma \vdash \sigma}\ (\wedge_{E_1}) \qquad \frac{\Gamma \vdash \sigma \wedge \tau}{\Gamma \vdash \tau}\ (\wedge_{E_2})$$

$$\frac{\Gamma \cup \{\sigma\} \vdash \tau}{\Gamma \vdash \sigma \supset \tau}\ (\supset_I) \qquad \frac{\Gamma \vdash \sigma \supset \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}\ (\supset_E)$$

Arată cunoscut?

Corespondența Curry-Howard

λ-calcul cu tipuri	Deducție naturală
$\dfrac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau}\ (\times_I)$	$\dfrac{\Gamma \vdash \sigma \quad \Gamma \vdash \tau}{\Gamma \vdash \sigma \wedge \tau}\ (\wedge_I)$
$\dfrac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \mathit{fst}\ M : \sigma}\ (\times_{E_1})$	$\dfrac{\Gamma \vdash \sigma \wedge \tau}{\Gamma \vdash \sigma}\ (\wedge_{E_1})$
$\dfrac{\Gamma \vdash p : \sigma \times \tau}{\Gamma \vdash \mathit{snd}\ p : \tau}\ (\times_{E_2})$	$\dfrac{\Gamma \vdash \sigma \wedge \tau}{\Gamma \vdash \tau}\ (\wedge_{E_2})$
$\dfrac{\Gamma \cup \{x : \sigma\} \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to_I)$	$\dfrac{\Gamma \cup \{\sigma\} \vdash \tau}{\Gamma \vdash \sigma \supset \tau}\ (\supset_I)$
$\dfrac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\ (\to_E)$	$\dfrac{\Gamma \vdash \sigma \supset \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}\ (\supset_E)$

Propositions are types! ♡

Să analizăm mai atent

λ-calcul cu tipuri	Deducție naturală
$\Gamma \vdash M : \sigma$	$\Gamma \vdash \sigma$

Faptul că există un termen de tip σ *(inhabitation of type σ)*
înseamnă că σ este teoremă/are o demonstrație în logică! $\heartsuit$

Să analizăm mai atent

λ-calcul cu tipuri

$$\frac{\{x : \sigma\} \vdash x : \sigma}{\vdash \lambda x.\, x : \sigma \to \sigma}\ (\to_I)$$

Deducție naturală

$$\frac{\{\sigma\} \vdash \sigma}{\vdash \sigma \supset \sigma}\ (\supset_I)$$

Să analizăm mai atent

λ-calcul cu tipuri

$$\frac{\{x : \sigma\} \vdash x : \sigma}{\vdash \lambda x.\, x : \sigma \to \sigma}\ (\to_I)$$

$$\frac{\dfrac{\overline{\{x : \sigma, y : \tau\} \vdash x : \sigma}}{\{x : \sigma\} \vdash \lambda y.\, x : \tau \to \sigma}\ (\to_I)}{\vdash \lambda x.\, (\lambda y.\, x) : \sigma \to (\tau \to \sigma)}\ (\to_I)$$

Deducție naturală

$$\frac{\{\sigma\} \vdash \sigma}{\vdash \sigma \supset \sigma}\ (\supset_I)$$

$$\frac{\dfrac{\overline{\{\sigma, \tau\} \vdash \sigma}}{\{\sigma\} \vdash \tau \to \sigma}\ (\supset_I)}{\vdash \sigma \to (\tau \to \sigma)}\ (\supset_I)$$

Să analizăm mai atent

λ-calcul cu tipuri

$$\frac{\{x : \sigma\} \vdash x : \sigma}{\vdash \lambda x.\, x : \sigma \to \sigma}\ (\to_I)$$

$$\frac{\dfrac{\overline{\{x : \sigma, y : \tau\} \vdash x : \sigma}}{\{x : \sigma\} \vdash \lambda y.\, x : \tau \to \sigma}\ (\to_I)}{\vdash \lambda x.\, (\lambda y.\, x) : \sigma \to (\tau \to \sigma)}\ (\to_I)$$

Deducție naturală

$$\frac{\{\sigma\} \vdash \sigma}{\vdash \sigma \supset \sigma}\ (\supset_I)$$

$$\frac{\dfrac{\overline{\{\sigma, \tau\} \vdash \sigma}}{\{\sigma\} \vdash \tau \to \sigma}\ (\supset_I)}{\vdash \sigma \to (\tau \to \sigma)}\ (\supset_I)$$

Proofs are Terms! ♡

Demonstrațiile sunt termeni!

Corespondența Curry-Howard

Teoria Tipurilor	Logică
tipuri	formule
termeni	demonstrații
inhabitation a tipului σ	demonstrație a lui σ

Corespondența Curry-Howard

Teoria Tipurilor	Logică
tipuri	formule
termeni	demonstrații
inhabitation a tipului σ	demonstrație a lui σ
tip produs	conjuncție
tip funcție	implicație

Corespondența Curry-Howard

Teoria Tipurilor	Logică
tipuri	formule
termeni	demonstrații
inhabitation a tipului σ	demonstrație a lui σ
tip produs	conjuncție
tip funcție	implicație
tip sumă	disjuncție
tipul void	false
tipul unit	true

Logica intuiționistă

- Logică constructivistă
- Bazată pe noțiunea de demonstrație
- Utilă deoarece demonstrațiile sunt executabile și produc exemple
  Permite "extragererea" de programe demonstrate a fi corecte.
- Baza pentru *proof assistants* (e.g., Coq, Agda, Idris)
- Următoarele formule echivalente nu sunt demonstrabile în logica intuiționistă!
  - dubla negație: $\neg\neg\varphi \supset \varphi$
  - excluded middle: $\varphi \vee \neg\varphi$
  - legea lui Pierce: $((\varphi \supset \tau) \supset \varphi) \supset \varphi$
- Nu există semantică cu tabele de adevăr pentru logica intuiționistă! Semantici alternative (e.g., semantica de tip Kripke)

Corespondența Curry-Howard

Inițial, corespondența Curry-Howard a fost între

Calculul Church $\lambda \rightarrow$	Sistemul de deducție naturală al lui Gentzen pentru logica intuiționistă

De ce?

- Este pur si simplu fascinant
- Nu gândiți logica și informatica ca domenii diferite.
- Gândind din perspective diferite ne poate ajuta să știm ce este posibil/imposibil.
- Teoria tipurilor nu ar trebui să fie o adunătură *ad hoc* de reguli!

Quiz time!

https://tinyurl.com/C06-Quiz1

Pe săptămâna viitoare!