

Proofs as Programs

Exercises in Type Level Programming

Joachim Tilsted Kristensen

Univertity of Oslo

Kosice, 16 Nov. 2023

About the project

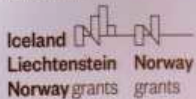
The **Modern Approaches and Tools for Teaching Classes at the University Level in Theoretical Computer Science Courses of Logic, Types, and Semantics** initiative no: **FBR-PDI-025** is funded by Iceland, Liechtenstein and Norway through the EEA Grants and Norway Grants.



Working together for a green, competitive and inclusive Europe

Talk given in context of

The Modern Approaches and Tools for Teaching Classes at the University Level in Theoretical Computer Science Courses of Logic, Types, and Semantics initiative no. FUR-FOR-025 is funded by Iceland, Liechtenstein and Norway through the EEA Grants and Norway Grants.



Working together for a green, competitive and inclusive Europe
<https://www.eea.grants.europa.eu/>

11/41

Agenda

- ▶ Propositions as types, proofs as programs.
- ▶ Exercises.
- ▶ What are dependent types.
- ▶ How can we simulate dependent types in Haskell.
- ▶ Exercises.
- ▶ Discussion.

The Curry-Howard-Lambek Correspondence

From Wikipedia

- ▶ Curry 1934 : Types look like axiom schemes for intuitionistic logic.
- ▶ Curry 1958 : Hilbert style deduction systems coincide with the typed fragment of combinatory logic.
- ▶ Howard 1969 : Natural deduction can be directly interpreted in a typed lambda calculus.
- ▶ Lambek 2005 : Lambda calculi are structurally equivalent to Cartesian closed categories.

Propositions as types, Proofs as Programs (Exercises)

`https://github.com/jtkristensen/
exercises-in-type-level-programming/tree/main/
tuke-2023/ProofsAsPrograms`

Lambda Cube (λ_{\rightarrow})

$$t ::= x \mid \lambda x : \tau. t \mid t_1 t_2$$

$$\tau ::= \tau_1 \rightarrow \tau_2$$

$\boxed{\Gamma \vdash t : \tau}$, for closed terms t :

$$\text{Var}_t : \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \text{Abs}_{t \rightarrow t} : \frac{\Gamma[x \mapsto \tau_1] \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2}$$

$$\text{App}_{tt} : \frac{\Gamma \vdash t_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1 t_2 : \tau_2}$$

Lambda Cube (λ_2)

$t ::= x \mid \lambda x : \tau. t \mid \Lambda X. t \mid t_1 t_2 \mid t[\tau]$

$\tau ::= X \mid \forall X. \tau \mid \tau_1 \rightarrow \tau_2$

$\boxed{\Gamma \vdash t : \tau}$, for closed terms t :

$$\text{Var}_t : \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \text{Abs}_{t \rightarrow t} : \frac{\Gamma[x \mapsto \tau_1] \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2}$$

$$\text{App}_{tt} : \frac{\Gamma \vdash t_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1 t_2 : \tau_2}$$

$\boxed{\Gamma \vdash t : \tau}$, for closed terms t :

$$\text{Abs}_{\tau \rightarrow t} : \frac{\Gamma \vdash t : \tau}{\Gamma \vdash \Lambda X. t : \forall X. \tau} \quad \text{App}_{t\tau} : \frac{\Gamma \vdash t : \forall X. \tau_1}{\Gamma \vdash t[\tau_2] : \tau_1[\tau_2/X]}$$

Lambda Cube (λ_ω)

$$t ::= x \mid \lambda x : \tau. t \mid t_1 t_2$$

$$\tau ::= X \mid \lambda X : \kappa. \tau \mid \tau_1 \rightarrow \tau_2 \mid \tau_1 \tau_2$$

$$\kappa ::= * \mid \kappa_1 \rightarrow \kappa_2$$

$\boxed{\Gamma \vdash t : \tau}$, for closed terms t :

$$\text{Var}_t : \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \text{Abs}_{t \rightarrow t} : \frac{\Gamma[x \mapsto \tau_1] \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2}$$

$$\text{App}_{tt} : \frac{\Gamma \vdash t_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1 t_2 : \tau_2}$$

repeat Var_t , $\text{Abs}_{t \rightarrow t}$ and App_{tt} to get Var_τ , $\text{Abs}_{\tau \rightarrow \tau}$ and $\text{App}_{\tau\tau}$.

Lambda Cube (λ_{Π})

$$t ::= x \mid \lambda x : \tau. B \mid t_1 t_2$$

$$\tau ::= \Pi x : A. B \mid \tau \rightarrow \tau$$

$$\kappa ::= * \mid \Pi x : \tau. \kappa$$

$\boxed{\Gamma \vdash t : \tau}$, for closed terms t :

$$\text{Var}_t : \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \text{Abs}_{t \rightarrow t} : \frac{\Gamma[x \mapsto \tau_1] \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2}$$

$$\text{App}_{tt} : \frac{\Gamma \vdash t_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1 t_2 : \tau_2}$$

$\boxed{\Gamma \vdash t : \tau}$, for closed terms t :

$$\text{Abs}_{t \rightarrow \tau} : \frac{\Gamma \vdash A : \tau \quad \Gamma[x \mapsto A] \vdash B : \sigma \quad \Gamma[x \mapsto A] \vdash B' : B}{\Gamma \vdash \lambda x : A. B' : \Pi x : A. B} \quad (\sigma \in \{\kappa, \tau\})$$

$$\text{App}_{\tau t} : \frac{\Gamma \vdash t_1 : \Pi x : \tau. B \quad \Gamma \vdash t_2 : \tau}{\Gamma \vdash t_1 t_2 : B[t_2/x]}$$

Simulating Dependent types in Haskell (Exercises)

`https://github.com/jtkristensen/
exercises-in-type-level-programming/tree/main/
tuke-2023/NatProperties`

Discussion