

# Introduction to Beamer

Firstname Lastname

Department of Computer Science and Engineering  
IIT Bombay.

Powai, Mumbai - 400076

`userid@cse.iitb.ac.in`

October 31, 2019

# This is the title

Beamer is a  $\text{\LaTeX}$  class for preparing presentations.

- 1 Slides are called frames in Beamer.
- 2 This is the usual ordered list in  $\text{\LaTeX}$ .
- 3 Following slides will contain random content which will show you various ways of using it. You need to replicate it.
- 4 Of course! we will give you boilerplate code!

# Type Rules

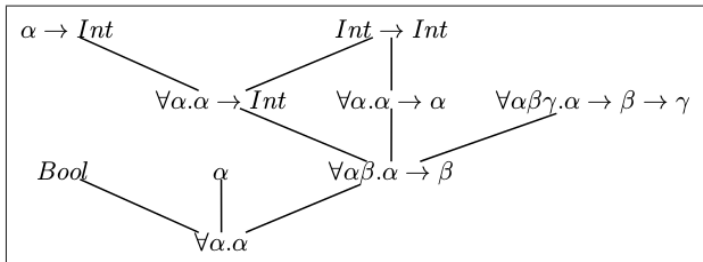


Figure: This is the caption.

# Type Rules

- A substitution is a list of pairs denoted as  $S = \{\alpha_1/\tau_1 \dots \alpha_n/\tau_n\}$ .
- A substitution  $S$  applied on a type expression  $\sigma$ , denoted by  $S(\sigma)$  involves simultaneous substitution of the variables  $\alpha_1 \dots \alpha_n$ , if they occur free in  $\sigma$ , by the corresponding type expressions  $\tau_1 \dots \tau_n$ .

## Definition

Let  $\sigma = \forall \alpha_1 \dots \alpha_m. \tau$  and  $\sigma' = \forall \beta_1 \dots \beta_n. \tau'$ . Then  $\sigma'$  is a generic instance of  $\sigma$ , iff there is a substitution  $S$  acting only on  $\{\alpha_1 \dots \alpha_m\}$  such that  $\tau' = S(\tau)$  and no  $\beta_i$  is free in  $\sigma$ .

- Clearly, the restriction that no  $\beta_i$  is free in  $\sigma$  is needed, else we would have absurdities like  $\alpha \rightarrow \text{Int} \leq \forall \alpha. \alpha \rightarrow \text{Int}$ .

## Recapitulation – Type rules for $\lambda_2$

$$\Gamma \cup \{x :: \sigma\} \vdash x :: \sigma \quad (\text{VAR})$$

$$\Gamma \cup \{c :: \sigma\} \vdash c :: \sigma \quad (\text{CON})$$

$$\frac{\Gamma \vdash M :: \sigma \quad \sigma' \geq \sigma}{\Gamma \vdash M :: \sigma'} \quad (\text{INST})$$

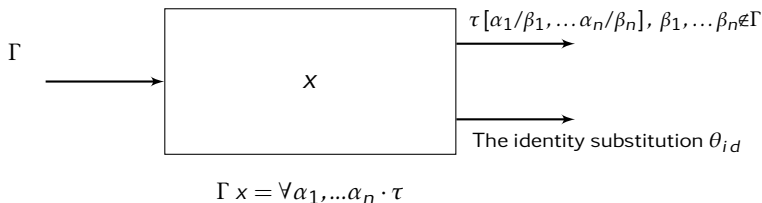
$$\frac{\Gamma \vdash M :: \sigma \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash M :: \forall \alpha. \sigma} \quad (\text{GEN})$$

$$\frac{\Gamma \vdash M :: \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash N :: \tau_1}{\Gamma \vdash M :: \sigma'} \quad (\text{M-APP})$$

$$\frac{\Gamma, x :: \tau_1 \vdash M :: \tau_2}{\Gamma \vdash \lambda x. M :: \tau_1 \rightarrow \tau_2} \quad (\text{M-ABS})$$

# Hindley-Milner - Type checking variables

1:  $t$  is a variable  $x$



- $\beta_1, \dots, \beta_n$  are fresh variables.
- Reason for monomorphising the type of  $x$ : We try to find the type of a variable only in the context of an application, and our application is monomorphic.

# Hindley-Milner - Type checking applications

- 1 Typecheck  $e_1$  with the initial environment  $\Gamma$ . Result is  $\tau_1$  and  $\theta_1$ .
- 2 Typecheck  $e_2$  with the environment  $\theta_1\Gamma$ . Result is  $\tau_2$  and  $\theta_2$ .
- 3 Unify  $\theta_2\tau_1$  and  $\tau_2 \rightarrow \alpha$ . Assume that unifier is  $\theta$ . And the unified term  $(\theta\alpha)$  is  $\tau_3$ .
- 4 Type of the application is  $\tau_3$  and the final substitution is  $\theta \circ \theta_2 \circ \theta_1$ .