Introduction to Beamer

Akkapaka Saikiran

Department of Computer Science and Engineering
IIT Bombay.
Powai, Mumbai - 400076

psycherun@cse.iitb.ac.in

November 6, 2019

This is the title

Beamer is a LaTeX class for preparing presentations.

1. Slides are called frames in Beamer.
2. This is the usual ordered list in 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

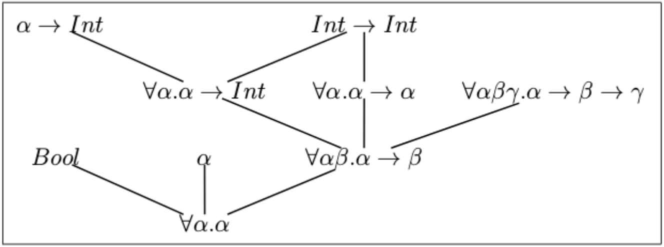

$\alpha \to Int$
$Int \to Int$
$\forall\alpha.\alpha \to Int$
$\forall\alpha.\alpha \to \alpha$
$\forall\alpha\beta\gamma.\alpha \to \beta \to \gamma$
$Bool$
α
$\forall\alpha\beta.\alpha \to \beta$
$\forall\alpha.\alpha$

Figure: This is the caption

Type Rules

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

Definition

Let $\sigma = \forall \alpha_1 \ldots \alpha_m.\tau$ and $\sigma_0 = \forall \beta_1 \ldots \beta_n.\tau_0$. Then σ_0 is a *generic instance* of σ , iff there is a substitution S acting only on $\{\alpha_1 \ldots \alpha_m\}$ such that $\tau' = S(\tau)$ and no β_i is free in σ.

- Clearly, the restriction that no β_i is free in σ is needed, else we would have absurdities like $\alpha \rightarrow Int \leq \forall \alpha.\alpha \rightarrow Int$.

Recapitulation Type rules for λ_2

$$\Gamma \cup \{x :: \sigma\} \vdash x :: \sigma \tag{Var}$$

$$\Gamma \cup \{x :: \sigma\} \vdash c :: \sigma \tag{Con}$$

$$\frac{\Gamma \vdash M :: \sigma \qquad \sigma' \geq \sigma}{\Gamma \vdash M :: \sigma'} \tag{Inst}$$

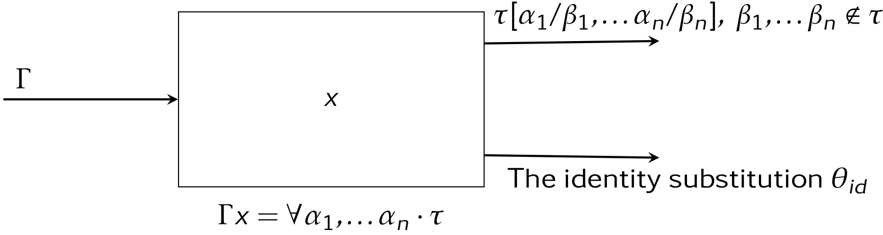
$$\frac{\Gamma \vdash M :: \sigma \qquad \alpha \notin FV(\Gamma)}{\Gamma \vdash M :: \forall \alpha . \sigma} \tag{Gen}$$

$$\frac{\Gamma \vdash M :: \tau_1 \rightarrow \tau_2 \qquad \Gamma \vdash N :: \tau}{\Gamma \vdash M \ \ N :: \tau_2} \tag{M-App}$$

$$\frac{\Gamma, x :: \tau_1 \vdash M :: \tau_2}{\Gamma \vdash \lambda x . M :: \tau_1 \rightarrow \tau_2} \tag{M-Abs}$$

Hindley-Milner - Type checking applications

1: t **is a variable** x



$\tau[\alpha_1/\beta_1, \ldots \alpha_n/\beta_n],\ \beta_1, \ldots \beta_n \notin \tau$

Γ

x

The identity substitution θ_{id}

$\Gamma x = \forall \alpha_1, \ldots \alpha_n \cdot \tau$

- $\beta_1, \ldots, \beta_n$ 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 Γ. Result is τ_1 and θ_1.

2 Typecheck e_2 with the environment $\theta_1\Gamma$. Result is τ_2 and θ_2.

3 Unify $\theta_2\tau_1$ and $\tau_2 \rightarrow \alpha$. Assume that unifier is θ. And the unified term $(\theta\ \alpha)$ is τ_3.

4 Type of the application is τ_3 and the final substitution is $\theta \circ \theta_2 \circ \theta_1$.