

Eval and Machine*

inductive
 eval :: "lam \Rightarrow lam \Rightarrow bool" ("_ \Downarrow _")
where
 e_Lam: "Lam [x].t \Downarrow Lam [x].t"
 | e_App: "[t₁ \Downarrow Lam [x].t; t₂ \Downarrow v; t[x::=v] \Downarrow v] \Rightarrow App t₁ t₂ \Downarrow v"

inductive
 machines :: "lam \Rightarrow ctxs \Rightarrow lam \Rightarrow ctxs \Rightarrow bool" ("<_> \mapsto^* <_>")
where
 ms₁: "<e,Es> \mapsto^* <e,Es>"
 | ms₂: "[<e₁,Es_{1\mapsto <e₂,Es₂₂,Es_{2\mapsto^* <e₃,Es_{3\Rightarrow <e₁,Es_{1\mapsto^* <e₃,Es₃}}}}

Princeton, 15. October 2008 - p. 1/36

Definition for Types

nominal_datatype ty =
 tVar "string"
 | tArr "ty" "ty" ("_ \rightarrow _")

$$\frac{(x:T) \in \Gamma \text{ valid } \Gamma}{\Gamma \vdash x : T} \quad \frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 t_2 : T_2}$$

$$\frac{x \# \Gamma \quad (x:T_1) :: \Gamma \vdash t : T_2}{\Gamma \vdash \lambda x.t : T_1 \rightarrow T_2}$$

$$\frac{}{\text{valid } []} \quad \frac{x \# \Gamma \quad \text{valid } \Gamma}{\text{valid } (x:T) :: \Gamma}$$

Princeton, 15. October 2008 - p. 18/36

Typing Judgement

types ty_ctx = "(name \times ty) list"

inductive
 valid :: "ty_ctx \Rightarrow bool"
where
 v₁: "valid []"
 | v₂: "[valid Γ ; x# Γ] \Rightarrow valid ((x,T)# Γ)"

inductive
 typing :: "ty_ctx \Rightarrow lam \Rightarrow ty \Rightarrow bool" ("_ \vdash _ : _")
where
 t_Var: "[valid Γ ; (x,T) \in set Γ] \Rightarrow $\Gamma \vdash$ Var x : T"
 | t_App: "[$\Gamma \vdash t_1 : T_1 \rightarrow T_2$; $\Gamma \vdash t_2 : T_1$] \Rightarrow $\Gamma \vdash$ App t₁ t₂ : T₂"
 | t_Lam: "[x# Γ ; (x,T₁)# $\Gamma \vdash t : T_2$] \Rightarrow $\Gamma \vdash$ Lam [x].t : T₁ \rightarrow T₂"

Princeton, 15. October 2008 - p. 19/36

Induction Principle for Typing

- The induction principle that comes with the typing definition is as follows:

$$\forall \Gamma x T. (x:T) \in \Gamma \wedge \text{valid } \Gamma \Rightarrow P \Gamma (x) T$$

$$\forall \Gamma t_1 t_2 T_1 T_2. P \Gamma t_1 (T_1 \rightarrow T_2) \wedge P \Gamma t_2 T_1 \Rightarrow P \Gamma (t_1 t_2) T_2$$

$$\frac{\forall \Gamma x t T_1 T_2. x \# \Gamma \wedge P ((x:T_1) :: \Gamma) t T_2 \Rightarrow P \Gamma (\lambda x.t) (T_1 \rightarrow T_2)}{\Gamma \vdash t : T \Rightarrow P \Gamma t T}$$

Note the quantifiers!

Princeton, 15. October 2008 - p. 25/36

Faulty Reasoning

- Consider the two-place relation **foo**:

$$\frac{x \mapsto x}{x \mapsto x} \quad \frac{t_1 t_2 \mapsto t_1 t_2}{t_1 t_2 \mapsto t_1 t_2} \quad \frac{t \mapsto t'}{\lambda x.t \mapsto t'}$$

- The lemma we going to prove:

Let $t \mapsto t'$. If $y \# t$ then $y \# t'$.

- Case 3:

- We know $y \# \lambda x.t$. We have to show $y \# t'$.
- The IH says: if $y \# t$ then $y \# t'$.
- So we have $y \# t$. Hence $y \# t'$ by IH. Done!

Princeton, 15. October 2008 - p. 34/36

Strong Induction Principle

$$\forall \Gamma x T c. (x:T) \in \Gamma \wedge \text{valid } \Gamma \Rightarrow P c \Gamma (x) T$$

$$\frac{\forall \Gamma t_1 t_2 T_1 T_2 c. (\forall d. P d \Gamma t_1 (T_1 \rightarrow T_2)) \wedge (\forall d. P d \Gamma t_2 T_1) \Rightarrow P c \Gamma (t_1 t_2) T_2}{\forall \Gamma x t T_1 T_2 c. x \# \Gamma \wedge x \# c \wedge (\forall d. P d ((x:T_1) :: \Gamma) t T_2) \Rightarrow P c \Gamma (\lambda x.t) (T_1 \rightarrow T_2)}$$

$$\frac{}{\Gamma \vdash t : T \Rightarrow P c \Gamma t T}$$

- Instead we are going to use the strong induction principle and set up the induction so that the binder "avoids" T_2 .

Princeton, 15. October 2008 - p. 35/36