missing:
* Proofs for Lemmata
* expr let where unification takes place
* expr case
* type classes?

Grammar

$$
\begin{array}{lll}
e & ::= & \mathrm{x} \mid \mathrm{c} \mid \lambda x.e \mid e\ e \mid \hat{\Lambda} p : \tau.e \mid e\ @ \mid \Lambda\alpha.e \mid e\,[\tau] \text{ hiding something} \\
\\
\text{hiding something} & & \\
B & ::= & \texttt{int} \mid \texttt{bool} \mid \alpha \\
\mathbb{T}(\mathbb{B}) & ::= & \{v : B \mid \mathbb{B}\} \mid x : \mathbb{T}(\mathbb{B}) \to \mathbb{T}(\mathbb{B}) \text{ hiding something} \\
\\
\mathbb{P}(\mathbb{B}) & ::= & \mathbb{T}(\mathbb{B}) \mid \forall p : \tau.\mathbb{P}(\mathbb{B}) \\
\mathbb{S}(\mathbb{B}) & ::= & \mathbb{P}(\mathbb{B}) \mid \forall\alpha.\mathbb{S}(\mathbb{B}) \\
\tau, \pi, \sigma & ::= & \mathbb{T}(\top),\ \mathbb{P}(\top),\ \mathbb{S}(\top) \\
\text{hiding something} & & \\
\hat{T}, \hat{P}, \hat{S} & ::= & \mathbb{T}(\mathbb{Q}),\ \mathbb{P}(\mathbb{Q}),\ \mathbb{S}(\mathbb{Q}) \\
T, P, S & ::= & \mathbb{T}(\mathrm{e}),\ \mathbb{P}(\mathrm{e}),\ \mathbb{S}(\mathrm{e}) \\
v & ::= & \mathrm{x} \mid \mathrm{c} \mid \lambda x.e \mid \hat{\Lambda} p : \tau.v \mid \Lambda\alpha.v
\end{array}
$$

Operational Semantics

$\boxed{e \not\hookrightarrow e}$

$$\frac{e_1 \not\hookrightarrow e_1'}{e_1\ e_2 \not\hookrightarrow e_1'\ e_2}\ \textsc{SP-PAppL}$$

$$\frac{e_2 \not\hookrightarrow e_2'}{v\ e_2 \not\hookrightarrow v\ e_2'}\ \textsc{SP-PAppR}$$

$$\frac{e \not\hookrightarrow e'}{e\,[\tau] \not\hookrightarrow e'\,[\tau]}\ \textsc{SP-PType-App}$$

$$\frac{e \not\hookrightarrow e'}{e\ @ \not\hookrightarrow e'\ @}\ \textsc{SP-PPred-App}$$

$$\frac{}{\lambda x.e\ v \not\hookrightarrow e\,[x \mapsto v]}\ \textsc{SP-EApp}$$

$$\frac{}{c\ v \not\hookrightarrow [|c|](v)}\ \textsc{SP-EApp-Con}$$

$$\frac{}{(\hat{\Lambda} p : \tau.v)\ @ \not\hookrightarrow v}\ \textsc{SP-EPred-App}$$

are the following two rules valid?

$$\frac{}{c\ @ \not\hookrightarrow [|c|]\ @}\ \textsc{SP-EPApp-Con}$$

$$\frac{}{c\,[\tau] \not\hookrightarrow [|c|]\,[\tau]}\ \textsc{SP-ETApp-Con}$$

$$\frac{}{(\Lambda\alpha.v)\,[\tau] \not\hookrightarrow v\,[\alpha \mapsto \tau]}\ \textsc{SP-EType-App}$$

$\boxed{e \hookrightarrow e}$

$$\frac{e_1 \hookrightarrow e_1'}{e_1\ e_2 \hookrightarrow e_1'\ e_2}\ \textsc{S-PAppL}$$

$$\frac{e_2 \hookrightarrow e_2'}{v\ e_2 \hookrightarrow v\ e_2'}\ \textsc{S-PAppR}$$

$$\frac{e \hookrightarrow e'}{e\,[\tau] \hookrightarrow e'\,[\tau]}\ \textsc{S-PType-App}$$

$$\frac{e \hookrightarrow e'}{e\ @ \hookrightarrow e'\ @}\ \textsc{S-PPred-App}$$

$$\frac{}{\lambda x.e\ v \hookrightarrow e\,[x \mapsto v]}\ \textsc{S-EApp}$$

$$\frac{}{c\ v \hookrightarrow [|c|](v)}\ \textsc{S-EApp-Con}$$

$$\frac{}{(\hat{\Lambda} p : \tau.v)\ @ \hookrightarrow v}\ \textsc{S-EPred-App}$$

are the following two rules valid?

$$\frac{}{c\ @ \hookrightarrow [|c|]\ @}\ \text{S-EPApp-Con}$$

$$\frac{}{c\,[\tau] \hookrightarrow [|c|]\,[\tau]}\ \text{S-ETApp-Con}$$

$$\frac{}{(\Lambda\alpha.v)\,[\tau] \hookrightarrow v\,[\alpha \mapsto \tau]}\ \text{S-EType-App}$$

Proves

Definition 1 (Constants)**.** *Each constant c has type $tc\,(\mathtt{c})$, such that*

1. $\emptyset \Vdash c : tc\,(\mathtt{c})$
2. *if* $tc\,(\mathtt{c}) \equiv x : T_x \to T$ *then for all values* $v \in T_x$, $[|c|](v)$ *is defined and* $\emptyset \Vdash [|c|](v) : T\,[x \mapsto v]$.
3. *if* $tc\,(\mathtt{c}) \equiv \forall\alpha.S$ *then for all types* τ, *if for some* T, *then* $\mathtt{Schema}\,(T) = \tau$ *and* $\emptyset \vdash T$, $[|c|]\,[\tau]$ *is defined and* $\emptyset \Vdash [|c|]\,[\tau] : S\,[\alpha \mapsto T]$.
4. *if* $tc\,(\mathtt{c}) \equiv \forall p : \tau.S$ *then for all values* v, *if* $\emptyset \Vdash v : \tau \to \mathtt{bool}$, $[|c|]\ @$ *is defined and* $\emptyset \Vdash [|c|]\ @ : S\,[p \mapsto v]$.

Translate

$tr(x)$

$$\begin{aligned}
tr(x) &= x \\
tr(c) &= c && tr([|c|]) = [|c|],\ tr([|c|]\ @) = [|c|](v),\ tr([|c|]\,[\tau]) = [|c|]\,[\tau] \\
tr(\lambda x.e) &= \lambda x.tr(e) \\
tr(e_1\ e_2) &= tr(e_1)\ tr(e_2) \\
tr(\hat{\Lambda} p : \tau.e) &= \lambda p.tr(e) && , p \notin \mathtt{FreeVars}\,(tr(e)) \\
tr(e\ @) &= tr(e)\ v && v \text{ for which it type-checked} \\
tr(\Lambda\alpha.e) &= \Lambda\alpha.tr(e) \\
tr(e\,[\tau]) &= tr(e)\,[\tau]
\end{aligned}$$

Theorem 1. *If* $e_1 \not\hookrightarrow e_2$ *then* $tr(e_1) \hookrightarrow tr(e_2)$

Proof. We will prove that by induction on the evaluation relation.

- SP-PAppL

$$e_1\ e_2 \not\mapsto e_1'\ e_2$$

Inverting the rule we get

$$e_1 \not\mapsto e_1'$$

By IH we have

$$tr(e_1, \mathbb{Q}_V) \hookrightarrow tr(e_1', \mathbb{Q}_V) \tag{1}$$

By definition of $tr(\star)$ we have

$$tr(e_1\ e_2) = tr(e_1)\ tr(e_2) \tag{2}$$
$$tr(e_1'\ e_2) = tr(e_1')\ tr(e_2) \tag{3}$$

By -3 and S-PAppL, we have

$$tr(e_1\ e_2) \hookrightarrow tr(e_1'\ e_2)$$

- SP-PAppR

$$v\ e_2 \not\mapsto v\ e_2'$$

Inverting the rule we get

$$e_2 \not\mapsto e_2'$$

By IH we have

$$tr(e_2) \hookrightarrow tr(e_2') \tag{4}$$

By definition of $tr(\star)$ we have

$$tr(v\ e_2) = tr(v)\ tr(e_2) \tag{5}$$
$$tr(v\ e_2') = tr(v)\ tr(e_2') \tag{6}$$

By 4-6 and S-PAppR, we have

$$tr(v\ e_2) \hookrightarrow tr(v\ e_2')$$

- SP-PType-App

$$e\ [\tau] \not\mapsto e'\ [\tau]$$

Inverting the rule we get

$$e \not\mapsto e'$$

By IH we have

$$tr(e) \hookrightarrow tr(e') \tag{7}$$

By definition of $tr(\star)$ we have

$$tr(e\,[\tau]) = tr(e)\,[\tau] \tag{8}$$
$$tr(e'\,[\tau]) = tr(e')\,[\tau] \tag{9}$$

By 7-9 and S-PType-App, we have

$$tr(e\,[\tau]) \hookrightarrow tr(e'\,[\tau])$$

- SP-PPred-App

$$e\ @ \not\mapsto e'\ @$$

Inverting the rule we get

$$e \not\mapsto e'$$

By IH we have

$$tr(e) \hookrightarrow tr(e') \tag{10}$$

By definition of $tr(\star)$ we have

$$tr(e\ @) = tr(e)\ v_1 \tag{11}$$
$$tr(e'\ @) = tr(e')\ v_2 \tag{12}$$

missing $v_1 = v_2$, but $\Gamma = \emptyset$, so v true or false predicate

By 10-12 and S-PType-App, we have

$$tr(e\,[\tau]) \hookrightarrow tr(e'\,[\tau])$$

- SP-SP-EApp

$$\lambda x.e\ v \not\mapsto e\,[x \mapsto v]$$

By definition of $tr(\star)$ we have

$$tr(\lambda x.e\ v) = \lambda x.tr(e)\ tr(v) \tag{13}$$

By S-EApp and Lemma $tr(v)$ is value, we have

$$\lambda x.tr(e)\ tr(v) \hookrightarrow tr(e)\,[x \mapsto tr(v)] \tag{14}$$

By Lemma someLemma we have

$$tr(e\,[x \mapsto v]) = tr(e)\,[x \mapsto tr(v)]$$

So,

$$tr(\lambda x.e\ v) \hookrightarrow tr(e\,[x \mapsto v])$$

- SP-EApp-Con

$$c\ v \not\mapsto [|c|](v)$$

By definition of $tr(\star)$ we have

$$tr(c\ v) = tr(c)\ tr(v) \tag{15}$$
$$tr(c) = c \tag{16}$$

By S-EApp-Con and 15-18, we have

$$tr(c\ v) \hookrightarrow [|c|]\ tr(v) \tag{17}$$

By definition of $tr(\star)$ we have

$$tr([|c|]) = [|c|] \tag{18}$$

So,

$$tr(c\ v) \hookrightarrow tr([|c|](v))$$

- SP-EPred-App

$$(\hat{\Lambda} p : \tau.v)\ @ \not\mapsto v$$

By definition of $tr(\star)$ we have

$$tr((\hat{\Lambda} p : \tau.v)\ @) = \lambda p.tr(v)\ v_0 \qquad , p \notin \texttt{FreeVars}\,(tr(v)) \tag{19}$$

By S-EApp and 19, we have

$$\lambda p.tr(v)\ v_0 \hookrightarrow tr(v)\,[p \mapsto v_0] \tag{20}$$

But since $p \notin \texttt{FreeVars}\,(tr(v))$,

$$tr(v)\,[p \mapsto v_0] \equiv tr(v)$$

So,

$$(\hat{\Lambda} p : \tau.v)\ @ \hookrightarrow tr(v)$$

- SP-EPApp-Con

$$c\ @ \not\mapsto [|c|]\ @$$

By definition of $tr(\star)$ we have

$$tr(c\ @) = tr(c)\ v \tag{21}$$
$$tr(c) = c \tag{22}$$
$$tr([|c|]\ @) = [|c|]\ (v) \tag{23}$$

By S-EApp-Con and 23-22, we have

$$tr(c\ @) \hookrightarrow [|c|]\ (v) \tag{24}$$

So,

$$tr(c\ @) \hookrightarrow tr([|c|]\ @)$$

- SP-ETApp-Con

$$c\,[\tau] \not\mapsto [|c|]\,[\tau]$$

By definition of $tr(\star)$ we have

$$tr(c\,[\tau]) = tr(c)\,[tau] \tag{25}$$
$$tr(c) = c \tag{26}$$
$$tr([|c|]\,[\tau]) = [|c|]\,[\tau] \tag{27}$$

By S-ETApp-Con and 31-30, we have

$$tr(c\,[\tau]) \hookrightarrow [|c|]\,[\tau] \tag{28}$$

So,

$$tr(c\,[\tau]) \hookrightarrow tr([|c|]\,[\tau])$$

- SP-EType-App

$$(\Lambda\alpha.v)\,[\tau] \not\mapsto v\,[\alpha \mapsto \tau]$$

By definition of $tr(\star)$ we have

$$tr(c\,[\tau]) = tr(c)\,[tau] \tag{29}$$
$$tr(c) = c \tag{30}$$
$$tr([|c|]\,[\tau]) = [|c|]\,[\tau] \tag{31}$$

By S-ETApp-Con and 31-30, we have

$$tr(c\,[\tau]) \hookrightarrow [|c|]\,[\tau] \tag{32}$$

So,

$$tr(c\,[\tau]) \hookrightarrow tr([|c|]\,[\tau])$$

—— SP-EType-App

□