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

Grammar

$$
\begin{array}{lcl}
e & ::= & \mathrm{x} \,|\, \mathrm{c} \,|\, \lambda x.e \,|\, e\ e \,|\, \hat{\Lambda} p : \tau.e \,|\, e\ @ \,|\, \Lambda \alpha.e \,|\, e\,[\tau] \text{ hiding something}
\end{array}
$$

hiding something

$$
\begin{array}{lcl}
B & ::= & \mathtt{int} \,|\, \mathtt{bool} \,|\, \alpha \\
\mathbb{T}\,(\mathbb{B}) & ::= & \{v : B \mid \mathbb{B}\} \,|\, x : \mathbb{T}\,(\mathbb{B}) \to \mathbb{T}\,(\mathbb{B}) \text{ hiding something}
\end{array}
$$

$$
\begin{array}{lcl}
\mathbb{P}\,(\mathbb{B}) & ::= & \mathbb{T}\,(\mathbb{B}) \,|\, \forall p : \tau.\mathbb{P}\,(\mathbb{B}) \\
\mathbb{S}\,(\mathbb{B}) & ::= & \mathbb{P}\,(\mathbb{B}) \,|\, \forall \alpha.\mathbb{S}\,(\mathbb{B}) \\
\tau, \pi, \sigma & ::= & \mathbb{T}\,(\top),\ \mathbb{P}\,(\top),\ \mathbb{S}\,(\top)
\end{array}
$$

hiding something

$$
\begin{array}{lcl}
\hat{T}, \hat{P}, \hat{S} & ::= & \mathbb{T}\,(\mathbb{Q}),\ \mathbb{P}\,(\mathbb{Q}),\ \mathbb{S}\,(\mathbb{Q}) \\
T, P, S & ::= & \mathbb{T}(\mathrm{e}),\ \mathbb{P}(\mathrm{e}),\ \mathbb{S}(\mathrm{e}) \\
v & ::= & \mathrm{x} \,|\, \mathrm{c} \,|\, \lambda x.e \,|\, \hat{\Lambda} p : \tau.v \,|\, \Lambda \alpha.v
\end{array}
$$

Operational Semantics

$$\boxed{e \hookrightarrow e}$$

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

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

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

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

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

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

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

are the following two rules valid?

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

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

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

$$\boxed{\Gamma \vdash e : S}$$

$$\frac{\Gamma \vdash e : S_2 \qquad \Gamma \vdash S_2 <: S_1 \qquad \Gamma \vdash S_1}{\Gamma \vdash e : S_1}\ \textsc{T-Sub}$$

$$\frac{\Gamma(x) = \{v : B \mid e\}}{\Gamma \vdash x : \{v : B \mid e \wedge v = x\}}\ \textsc{T-Var-Base}$$

$$\frac{\Gamma(x) \neq \{v : B \mid e\}}{\Gamma \vdash x : \Gamma(x)}\ \textsc{T-Var}$$

$$\frac{}{\Gamma \vdash c : tc\,(\mathtt{c})}\ \textsc{T-Con}$$

$$\frac{\Gamma, x : T_x \vdash e : T \qquad \Gamma \vdash x : T_x \to T}{\Gamma \vdash \lambda x.e : x : T_x \to T}\ \textsc{T-Fun}$$

Rules

$$\frac{\Gamma \vdash e_1 : x : T_x \to T \qquad \Gamma \vdash e_2 : T_x}{\Gamma \vdash e_1\ e_2 : T\,[x \mapsto e_2]}\ \textsc{T-App}$$

$$\frac{\Gamma, p : \tau \to \mathtt{bool} \vdash e : S \qquad \Gamma \vdash p : (\tau \to \mathtt{bool}) \to S \qquad p \notin \mathtt{FreeVars}\,(e)}{\Gamma \vdash \hat{\Lambda} p : \tau.e : \forall p : \tau.S}\ \textsc{T-PGen}$$

$$\frac{\Gamma \vdash e : \forall p : \tau.S \qquad \Gamma \vdash v : \tau \to \mathtt{bool}}{\Gamma \vdash e\ @ : S\,[p \mapsto v]}\ \textsc{T-PInst}$$

$$\frac{\Gamma \vdash e : S \qquad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha.e : \forall\alpha.S}\ \textsc{T-Gen}$$

$$\frac{\Gamma \vdash e : \forall\alpha.S \qquad \mathtt{Schema}\,(T) = \tau \qquad \Gamma \vdash T}{\Gamma \vdash e\,[\tau] : S\,[\alpha \mapsto T]}\ \textsc{T-Inst}$$

hiding something

$$\boxed{\Gamma \vdash S}$$

$$\frac{\Gamma, v : B \vdash e : \mathtt{bool}}{\Gamma \vdash \{v : B \mid e\}}\ \textsc{WT-Base}$$

$$\frac{\Gamma \vdash T_x \qquad \Gamma, x : T_x \vdash T}{\Gamma \vdash x : T_x \to T}\ \textsc{WT-Fun}$$

$$\frac{\Gamma, p : x : \tau \to \mathtt{bool} \vdash P}{\Gamma \vdash \forall p : \tau.P}\ \textsc{WT-Pred}$$

$$\frac{\Gamma \vdash T}{\Gamma \vdash \forall\alpha.T}\ \textsc{WT-Poly}$$

hiding something

$$\boxed{\Gamma \vdash S_1 <: S_2}$$

$$\frac{\Gamma, v : B \vdash e_1 \Rightarrow e_2}{\Gamma \vdash \{v : B \mid e_1\} <: \{v : B \mid e_2\}} \ \text{<:-Base}$$

$$\frac{\Gamma \vdash T_{x_2} <: T_{x_1} \qquad \Gamma, x_2 : T_{x_2} \vdash T_1 \left[x_1 \mapsto x_2\right] <: T_2}{\Gamma \vdash x_1 : T_{x_1} \to T_1 <: x_2 : T_{x_2} \to T_2} \ \text{<:-Fun}$$

$$\frac{\Gamma, p_1 : x : \tau \to \texttt{bool} \vdash P_1 <: P_2 \left[p_2 \mapsto p_1\right]}{\Gamma \vdash \forall p_1 : \tau . P_1 <: \forall p_2 : \tau . P_2} \ \text{<:-Pred}$$

$$\frac{\Gamma \vdash S_1 <: S_2}{\Gamma \vdash \forall \alpha . S_1 <: \forall \alpha . S_2} \ \text{<:-Poly}$$

hiding something

$\boxed{\Gamma \vdash \rho}$

$$\frac{}{\emptyset \vdash \emptyset}\ \textsc{WS-Empty}$$

$$\frac{\Gamma \vdash \rho \qquad \emptyset \vdash v : \rho S}{\Gamma; x : S \vdash \rho; [x \mapsto v]}\ \textsc{WS-Ext}$$

define type sub, $\texttt{Schema}\,(T) = \tau$, sub τ on exprs and T on types

$\boxed{\Gamma \vdash \theta}$

$$\frac{}{\emptyset \vdash \emptyset}\ \textsc{WTS-Empty}$$

$$\frac{\Gamma \vdash \theta \qquad \emptyset \vdash \theta S}{\Gamma \vdash \theta; [a \mapsto S]}\ \textsc{WTS-Ext}$$

Proves

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

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

Lemma 1. *If* $e \hookrightarrow e'$ *and* $\emptyset \vdash e : S_e$ *and* $\emptyset \vdash e' : S_e$ *then* $\Gamma \vdash S\,[x \mapsto e'] <: S\,[x \mapsto e]$

ProofIdea. $[|\star|]$ *is defined to preserve operational semantics*

Lemma 2 (Value Substitution)**.** *If* $\Gamma \vdash \rho$ *then if* $\Gamma; \Gamma' \vdash e : S$ *then* $\rho\Gamma' \vdash \rho e : \rho S$

ProofIdea. *Pat-Ming Lemma 10*

Lemma 3 (Type Substitution)**.** *If* $\Gamma \vdash \theta$ *then if* $\Gamma; \Gamma' \vdash e : S$ *then* $\rho\Gamma' \vdash \theta e : \theta S$

ProofIdea. *???*

Theorem 1 (Preservation)**.** *If* $\emptyset \vdash e : S$ *and* $e \hookrightarrow e'$ *then* $\emptyset \vdash e' : S$

Proof. By induction on the typing derivation $\emptyset \vdash e : S$. We split cases on the rule used on the top of the derivation.

- T-Sub

  $$\emptyset \vdash e : S \qquad\qquad e \hookrightarrow e'$$

  By inversion, there exists an S' such that

  $$\emptyset \vdash e : S' \tag{1}$$
  $$\emptyset \vdash S' <: S \tag{2}$$

  By IH and 1

  $$\emptyset \vdash e' : S' \tag{3}$$

  Which, with 2 and rule T-Sub gives

  $$\emptyset \vdash e' : S \tag{4}$$

- T-Var-Base, T-Var, T-Con, T-Fun, T-PGen and T-Gen cases are trivial, since there can be no e' such that $e \hookrightarrow e'$

- T-App

  $$\emptyset \vdash e_1\ e_2 : S \qquad\qquad e_1\ e_2 \hookrightarrow e'$$

  By inversion, there exist x and T_x such that

  $$\emptyset \vdash e_1 : x : T_x \to T \tag{5}$$
  $$\emptyset \vdash e_2 : T_x \tag{6}$$
  $$S \equiv T\,[x \mapsto e_1] \tag{7}$$

  – exits e'_1 so that $e_1 \hookrightarrow e'_1$, so $e' \equiv e'_1\ e_2$
    From IH and 5,
    $$\emptyset \vdash e'_1 : x : T_x \to T$$
    Which, with 6 and rule T-App gives
    $$\emptyset \vdash e'_1\ e_2 : T\,[x \mapsto e'_1] \tag{8}$$
    From Lemma 1 we get
    $$\emptyset \vdash T\,[x \mapsto e'_1] <: T\,[x \mapsto e_1]$$
    Which with 8 and T-Sub gives
    $$\emptyset \vdash e' : S$$
    .

  – e_1 is a value, $e_1 \equiv v$

* exits e'_2 so that $e_2 \hookrightarrow e'_2$, so $e' \equiv v\ e'_2$
  From IH and 6, $\emptyset \vdash e'_2 : T_x$. Which, whith 5 and T-App gives $\emptyset \vdash e' : S$.
* e_2 is a value, so $e_2 \equiv v_2$. Since e_1 is a value, it can not be variable, as e_1 is closed, and can not be of the form $\hat{\Lambda} p : \tau.e'$ nor $\Lambda\alpha.e'$, as these values can not have the desired type.
  - $e_1 \equiv \lambda x.e_{11}$, so $e' \equiv e_{11}\,[x \mapsto v_2]$ By inversion of the rule 5, and if we push the T-Sub rules down in the derivation tree we get

$$x : T_x \vdash e_{11} : T \tag{9}$$

    From 6 and WS-Ext we get $x : T_x \vdash [x \mapsto v_2]$. Which, with 9 and Lemma 2 gives $\emptyset \vdash e_{11}\,[x \mapsto v_2] : T\,[x \mapsto v_2]$, or $\emptyset \vdash e' : S$.
  - $e_1 \equiv c$, so $e' \equiv [|c|](v)$
    By rule 5 and T-Con we have $tc\,(\mathtt{c}) \equiv x : T_x \rightarrow T$. Which, with Definition 1 gives us $\emptyset \vdash [|c|](v_2) : T\,[x \mapsto v_2]$, or $\emptyset \vdash e' : S$.

- T-Inst There exist e_1, S_1, α and τ such that

$$e \equiv e_1\,[\tau] \qquad\qquad S \equiv S_1\,[\alpha \mapsto T]$$

  By inversion, we have

$$\emptyset \vdash e_1 : \forall\alpha.S_1 \tag{10}$$
$$\mathtt{Schema}\,(T) = \tau \tag{11}$$
$$\emptyset \vdash T \tag{12}$$

  If there exists e'_1, such that $e_1 \hookrightarrow e'_1$, then $e' \equiv e'_1\,[\tau]$. By IH and 10, we have $\emptyset \vdash e'_1 : \forall\alpha.S_1$. This, with 11, 12 and T-Inst gives $\emptyset \vdash e'_1\,[\tau] : S_1\,[\alpha \mapsto T]$, or $\emptyset \vdash e' : S$.

  Otherwise, e_1 is a value. From 10 there are two cases:

  - $e_1 \equiv \Lambda\alpha.v_1$, so $e' \equiv v_1\,[\alpha \mapsto \tau]$. By inverting the rule T-Gen and if we push the T-Sub rules down in the derivation tree, we get

$$\emptyset \vdash v_1 : S_1 \tag{13}$$

    By WTS-Extand 12 we have $\emptyset \vdash [\alpha \mapsto T]$. Which, by 13 and Lemma 3 gives $\emptyset \vdash v_1\,[\alpha \mapsto \tau] : S_1\,[\alpha \mapsto T]$ or $\emptyset \vdash e' : S$.
  - $e_1 \equiv c$, so $e' \equiv [|c|]\,[\tau]$
    By rule 5 and T-Con we have $tc\,(\mathtt{c}) \equiv \forall\alpha.S_1$. Which, with 1 gives us $\emptyset \vdash [|c|]\,[\tau] : S_1\,[\alpha \mapsto T]$, or $\emptyset \vdash e' : S$.

T-PInst There exist e_1, S_1, p and v such that

$$e \equiv e_1 \; @ \qquad\qquad S \equiv S_1 \left[p \mapsto v\right]$$

By inversion, we have

$$\emptyset \vdash e_1 : \forall p : \tau . S_1 \tag{14}$$

$$\emptyset \vdash v : \tau \rightarrow \texttt{bool} \tag{15}$$

If there exists e_1', such that $e_1 \hookrightarrow e_1'$, then $e' \equiv e_1' \; @$. By IH and 14, we have $\emptyset \vdash e_1' : \forall p : \tau . S_1$. This, with 15 and T-PInst gives $\emptyset \vdash e_1' \; @ : S_1 \left[p \mapsto v\right]$, or $\emptyset \vdash e' : S$.

Otherwise, e_1 is a value. From 14 there are two cases:

- $e_1 \equiv \hat{\Lambda} p : \tau . v_1$, so $e' \equiv v_1$. By inverting the rule T-PGen and if we push the T-Sub rules down in the derivation tree, we get

  $$p : \tau \rightarrow \texttt{bool} \vdash v_1 : S_1 \tag{16}$$

  $$p \notin \texttt{FreeVars}\left(v_1\right) \tag{17}$$

  By WS-Extand 16 we have $p : \tau \rightarrow \texttt{bool} \vdash \left[p \mapsto v\right]$, also by 17 we get $v_1 \equiv v_1 \left[p \mapsto v\right]$ Which, by 16 and Lemma 2 gives $\emptyset \vdash v_1 \left[p \mapsto v\right] : S_1 \left[p \mapsto v\right]$ or $\emptyset \vdash e' : S$.

- $e_1 \equiv c$, so $e' \equiv [|c|] \; @$

  By rule 14 and T-Con we have $tc\left(\texttt{c}\right) \equiv \forall p : \tau . S_1$. Which, with Definition 1 and 15, gives us $\emptyset \vdash [|c|] \; @ : S_1 \left[p \mapsto v\right]$, or $\emptyset \vdash e' : S$.

□

Theorem 2 (Progress)**.** *If $\emptyset \vdash e : S$ and e is not a value, then there exists an e' such that $e \hookrightarrow e'$.*

Proof. By induction on the typing derivation $\emptyset \vdash e : S$. We split cases on the rule used on the top of the derivation.

- T-Sub

$$\emptyset \vdash e : S$$

  By inversion, there exists an S' such that

$$\emptyset \vdash e : S' \tag{18}$$
$$\emptyset \vdash S' <: S \tag{19}$$

  If e is a value, it is trivial, as the assumptions of the theorem are not true. Otherwise, by IH and 18, there exists an e', such that $e \hookrightarrow e'$.

- T-Var-Base, T-Var, T-Con, T-Fun T-PGen, T-Gen cases are trivial, since e is a value.

- T-App

$$\emptyset \vdash e_1\ e_2 : S$$

  By inversion, there exist x and T_x such that

$$\emptyset \vdash e_1 : x : T_x \to T \tag{20}$$
$$\emptyset \vdash e_2 : T_x \tag{21}$$
$$S \equiv T\,[x \mapsto e_1] \tag{22}$$

  - e_1 is not a value. From IH and 20 there exits e_1' so that $e_1 \hookrightarrow e_1'$, so $e' \equiv e_1'\ e_2$.
  - e_1 is a value, $e_1 \equiv v$
    * e_2 is not a value. From IH and 21 these exits e_2' so that $e_2 \hookrightarrow e_2'$, so $e' \equiv v\ e_2'$
    * e_2 is a value, so $e_2 \equiv v_2$. Since e_1 is a value, it can not be variable, as e_1 is closed, and can not be of the form $\hat{\Lambda} p : \tau . e'$ nor $\Lambda \alpha . e'$, as these values can not have the desired type.
      · $e_1 \equiv \lambda x . e_{11}$, so $e' \equiv e_{11}\,[x \mapsto v_2]$.
      · $e_1 \equiv c$. By 20, $tc\,(\mathtt{c}) = x : T_x \to T$ and by 20 $\emptyset \vdash v_2 : T_x$, so, by Definition 1, $[|c|](v)$ is defined. So, $e' \equiv [|c|](v)$.

- T-Inst There exist e_1, S_1, α and τ such that

$$e \equiv e_1\,[\tau] \qquad\qquad S \equiv S_1\,[\alpha \mapsto T]$$

  By inversion, we have

$$\emptyset \vdash e_1 : \forall \alpha . S_1 \tag{23}$$
$$\mathtt{Schema}\,(T) = \tau \tag{24}$$
$$\emptyset \vdash T \tag{25}$$

- If e_1 is not a value, there exists e_1', such that $e_1 \hookrightarrow e_1'$, and $e' \equiv e_1' [\tau]$.
- If e_1 is a value. From 23 there are two cases:
  * $e_1 \equiv \Lambda\alpha.v_1$, so $e' \equiv v_1 [\alpha \mapsto \tau]$.
  * $e_1 \equiv c$. From 24, $tc(\texttt{c}) = \forall\alpha.S_1$. Which, by 24, 25 and Definition 1 gives that $[|c|] [\tau]$ is defined. So, $e' \equiv [|c|] [\tau]$.

- T-PInst There exist e_1, S_1, p and v such that

$$e \equiv e_1 \ @ \qquad\qquad S \equiv S_1 [p \mapsto v]$$

By inversion, we have

$$\emptyset \vdash e_1 : \forall p : \tau.S_1 \tag{26}$$

$$\emptyset \vdash v : x : \tau \to \texttt{bool} \tag{27}$$

- If e_1 is not a value, there exists e_1', such that $e_1 \hookrightarrow e_1'$, so, $e' \equiv e_1' \ @$.
- If e_1 is a value, From 26 there are two cases:
  * $e_1 \equiv \hat{\Lambda}p : \tau.v_1$, so $e' \equiv v_1$.
  * $e_1 \equiv c$. From 26, $tc(\texttt{c}) = \forall p : \tau.S_1$, which, by 27 and Definition 1, gives that $[|c|] \ @$ is defined. So, $e' \equiv [|c|] \ @$

$\square$

Theorem 3 (Soundness of Decidable Type Checking)**.** *If* $\Gamma, P \vdash_{Q,P} e : S$ *then* $\Gamma \vdash e : S$

ProofIdea. *By induction on the typing derivation*