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

Grammar

$$
\begin{array}{lcl}
e & ::= & \mathrm{x} \,|\, \mathrm{c} \,|\, \lambda x.e \,|\, e\ e \,|\, \forall p:\tau.e \,|\, e\ @ \,|\, \forall\alpha.e \,|\, e\,[\tau] \\
B & ::= & \texttt{int} \,|\, \texttt{bool} \,|\, \alpha \\
T & ::= & \{v : B \mid e\} \,|\, x : T \to T \\
P & ::= & \mathrm{T} \,|\, \forall p:\tau.P \\
S & ::= & \mathrm{P} \,|\, \forall\alpha.S \\
v & ::= & \mathrm{x} \,|\, \mathrm{c} \,|\, \lambda x.e \,|\, \forall p:\tau.v \,|\, \forall\alpha.v
\end{array}
$$

Operational Semantics

$\boxed{e \hookrightarrow e}$

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

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

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

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

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

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

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

are the following two rules valid?

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

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

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

Rules

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

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

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

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

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

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

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

$$\frac{\Gamma, p : T_p \vdash e : S \qquad \mathtt{Schema}\,(T_p) = \tau \rightarrow \mathtt{bool} \qquad p \notin \mathtt{FreeVars}\,(e)}{\Gamma \vdash \forall p : \tau.e : \forall p : \tau.S}\ \textsc{T-PGen}$$

$$\frac{\Gamma \vdash e : \forall p : \tau.S \qquad \Gamma \vdash v : T \qquad \mathtt{Schema}\,(T) = \tau \rightarrow \mathtt{bool} \qquad \Gamma \models T}{\Gamma \vdash e\ @ : S\,[p \mapsto v]}\ \textsc{T-PInst}$$

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

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

$$\boxed{\Gamma \models \rho}$$

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

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

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

$$\boxed{\Gamma \models \theta}$$

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

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

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* $\emptyset \models T$, $[|c|]\,[\tau]$ *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* $\emptyset \vdash v : T$, *where* $\mathtt{Schema}(T) = \tau \to \mathtt{bool}$ *and* $\emptyset \models T$, $[|c|]$ @ *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* $\Gamma \models \rho$ *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* $\Gamma \models \theta$ *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, 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,
    $$\emptyset \vdash e'_1 : x : T_x \to T$$
    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 $\forall p : \tau.e'$ nor $\forall \alpha.e'$, 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 $x : T_x \models [x \mapsto v_2]$. 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 $tc\,(\mathtt{c}) \equiv x : T_x \to T$. Which, with 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

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

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:

– $e_1 \equiv \forall \alpha.v_1$, 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 $\emptyset \models [\alpha \mapsto T]$. Which, by 13 and 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

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

By inversion, we have

$$\emptyset \vdash e_1 : \forall p : \tau.S_1 \tag{14}$$
$$\emptyset \vdash v : T \tag{15}$$
$$\texttt{Schema}\,(T) = \tau \to \texttt{bool} \tag{16}$$
$$\emptyset \models T \tag{17}$$

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- 17 and T-PINST gives $\emptyset \vdash e_1' \ @ : S_1 \, [p \mapsto v]$, or $\emptyset \vdash e' : S$.

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

- $e_1 \equiv \forall p : \tau.v_1$, 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

$$p : T_p \vdash v_1 : S_1 \tag{18}$$
$$\texttt{Schema}\,(T_p) = \tau \to \texttt{bool} \tag{19}$$
$$p \notin \texttt{FreeVars}\,(v_1) \tag{20}$$

  By WS-EXTand 18 we have $p : T_p \models [p \mapsto v]$, also by 20 we get $v_1 \equiv v_1 \, [p \mapsto v]$ Which, by 18 and Lemma 2 gives $\emptyset \vdash v_1 \, [p \mapsto v] : S_1 \, [p \mapsto v]$ or $\emptyset \vdash e' : S$.

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

  By rule 14 and T-CON we have $tc\,(\texttt{c}) \equiv \forall p : \tau.S_1$. Which, with Definition 1 and 15 - 17, gives us $\emptyset \vdash [|c|] \ @ : S_1 \, [p \mapsto v]$, or $\emptyset \vdash e' : S$.

$\square$

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

  $$\emptyset \vdash e : S' \tag{21}$$
  $$\emptyset \vdash S' <: S \tag{22}$$

  If e is a value, it is trivial, as the assumptions of the theorem are not true. Otherwise, by IH and 21, 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

  $$\emptyset \vdash e_1 : x : T_x \rightarrow T \tag{23}$$
  $$\emptyset \vdash e_2 : T_x \tag{24}$$
  $$S \equiv T\,[x \mapsto e_1] \tag{25}$$

  - e_1 is not a value. From IH and 23 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 24 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 $\forall p : \tau.e'$ nor $\forall \alpha.e'$, 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 23, $tc\,(\mathtt{c}) = x : T_x \rightarrow T$ and by 23 $\emptyset \vdash v_2 : T_x$, 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

  $$\emptyset \vdash e_1 : \forall \alpha.S_1 \tag{26}$$
  $$\mathtt{Schema}\,(T) = \tau \tag{27}$$
  $$\emptyset \models T \tag{28}$$

– 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 26 there are two cases:

  * $e_1 \equiv \forall \alpha . v_1$, so $e' \equiv v_1 \, [\alpha \mapsto \tau]$.
  * $e_1 \equiv c$. From 27, $tc\,(\mathtt{c}) = \forall \alpha . S_1$. Which, by 27, 28 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

$$\emptyset \vdash e_1 : \forall p : \tau . S_1 \tag{29}$$
$$\emptyset \vdash v : T \tag{30}$$
$$\mathtt{Schema}\,(T) = \tau \rightarrow \mathtt{bool} \tag{31}$$
$$\emptyset \models T \tag{32}$$

– 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 29 there are two cases:

  * $e_1 \equiv \forall p : \tau . v_1$, so $e' \equiv v_1$.
  * $e_1 \equiv c$. From 29, $tc\,(\mathtt{c}) = \forall p : \tau . S_1$, which, by 30 - 32 and Definition 1, gives that $[|c|] \ @$ is defined. So, $e' \equiv [|c|] \ @$

□

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

ProofIdea. *By induction on the typing derivation*