1 Grammar

1.1 Untyped Expressions

$$\begin{array}{rcll} eu & ::= & \mathbf{1} & \text{Unit} \\ & | & v & \text{Var} \\ & | & \lambda x \,.\, eu & \text{Lambda} \\ & | & eu_1 \; eu_2 & \text{App} \end{array}$$

1.2 Typed Expressions

$$\begin{array}{rcll} e & ::= & \mathbf{1} & \text{Unit} \\ & | & v & \text{Var} \\ & | & \lambda \, (x \, : \, t) \, . \, e & \text{Lambda} \\ & | & e_1 \; e_2 & \text{App} \end{array}$$

1.3 Types

$$\begin{array}{rcll} t & ::= & \mathbf{1} & \text{Unit} \\ & | & t_1 \to t_2 & \text{Arrow} \\ & | & u & \text{UVar} \end{array}$$

1.4 Typing Evironments

$$\begin{array}{rcll} \Gamma & ::= & \cdot & \text{Empty} \\ & | & \Gamma, v \, : \, t & \text{Var with Type} \end{array}$$

1.5 Type Stores

$$\begin{array}{rcll} \mathcal{S} & ::= & \cdot & \text{Empty} \\ & | & \mathcal{S}, u = t & \text{UVar with Type} \end{array}$$

2 Typing

2.1 Declarative Typing

Judgements of the form:

$$\Gamma \vdash e \, : \, t$$

$$\frac{}{\Gamma \vdash \mathbf{1} \, : \, \mathbf{1}} \text{ (DeclUnit)} \qquad \frac{\Gamma(v) = t}{\Gamma \vdash v \, : \, t} \text{ (DeclVar)}$$

$$\frac{\Gamma, x \, : \, t_1 \vdash e \, : \, t_2}{\Gamma \vdash \lambda \, (x \, : \, t_1) \, . \, e \, : \, t_1 \to t_2} \text{ (DeclLam)}$$

$$\frac{\Gamma \vdash e_1 \, : \, t_1 \to t_2 \qquad \Gamma \vdash e_2 \, : \, t_1}{\Gamma \vdash e_1 \; e_2 \, : \, t_2} \text{ (DeclApp)}$$

2.2 Type stores

We define the relation $\mathcal{S}_1 \subseteq \mathcal{S}_2$, which states informally that $\mathcal{S}_1$ is a prefix of $\mathcal{S}_2$. The Nil rule is not strictly needed.

$$\frac{}{\cdot \subseteq \mathcal{S}} \text{ (Nil)} \qquad \frac{}{\mathcal{S} \subseteq \mathcal{S}} \text{ (Refl)} \qquad \frac{\mathcal{S}_1 \subseteq \mathcal{S}_2}{\mathcal{S}_1 \subseteq \mathcal{S}_2, u = t} \text{ (Extension)}$$

It follows from the definition that the transitive propery holds for $\subseteq$.

We define applying a type store $\mathcal{S}$ over type contexts, types, and typed expressions.

For contexts Γ,

$$\begin{array}{rcll} \mathcal{S}(\cdot) & ::= & \cdot & \text{Empty} \\ \mathcal{S}(\Gamma, v\ :\ t) & ::= & \mathcal{S}(\Gamma), v\ :\ \mathcal{S}(t) & \text{Var with Type} \end{array}$$

For types t,

$$\begin{array}{rcll} \mathcal{S}(\mathbf{1}) & ::= & \mathbf{1} & \text{Unit} \\ \mathcal{S}(t_1 \to t_2) & ::= & \mathcal{S}(t_1) \to \mathcal{S}(t_2) & \text{Arrow} \\ (\mathcal{S}, u_1 = t)\,(u_2) & ::= & \begin{cases} \mathcal{S}(u_2) & u_1 \neq u_2 \\ \mathcal{S}(t) & u_1 = u_2 \end{cases} & \text{UVar} \\ \cdot(t) & ::= & t & \text{Empty} \end{array}$$

Note: though it is assumed that no UVar occurs twice in $\mathcal{S}$, in the case of multiple UVars, the first appearing one is used (from left to right, i.e. first-defined UVar).

For typed expressions e,

$$\begin{array}{rcll} \mathcal{S}(\mathbf{1}) & ::= & \mathbf{1} & \text{Unit} \\ \mathcal{S}(v) & ::= & v & \text{Var} \\ \mathcal{S}(\lambda\,(x\ :\ t)\ .\,e) & ::= & \lambda\,(x\ :\ \mathcal{S}(t))\ .\mathcal{S}(e) & \text{Lambda} \\ \mathcal{S}(e_1\ e_2) & ::= & \mathcal{S}(e_1)\ \mathcal{S}(e_2) & \text{App} \end{array}$$

2.3 Algorithmic Typing

Jugements of the form:

$$\Gamma, \mathcal{S} \vdash eu \Rightarrow e\ :\ t \dashv \mathcal{S}'$$

$$\frac{}{\Gamma, \mathcal{S} \vdash \mathbf{1} \Rightarrow \mathbf{1}\ :\ \mathbf{1} \dashv \mathcal{S}} \text{ (Unit)} \qquad \frac{\Gamma(v) = t}{\Gamma, \mathcal{S} \vdash v \Rightarrow v\ :\ t \dashv \mathcal{S}} \text{ (Var)}$$

$$\frac{\begin{array}{cc} \text{Fresh } u & \Gamma, \mathcal{S}_0 \vdash eu_1 \Rightarrow e_1\ :\ t_{\text{arr}} \dashv \mathcal{S}_1 \\ \Gamma, \mathcal{S}_1 \vdash eu_2 \Rightarrow e_2\ :\ t_{\text{arg}} \dashv \mathcal{S}_2 & \mathcal{S}_2 \vdash t_{\text{arr}} == t_{\text{arg}} \to u \dashv \mathcal{S}_3 \end{array}}{\Gamma, \mathcal{S}_0 \vdash eu_1\ eu_2 \Rightarrow e_1\ e_2\ :\ u \dashv \mathcal{S}_3} \text{ (App)}$$

$$\frac{\text{Fresh } u \qquad \Gamma, x : u, \mathcal{S}_0 \vdash eu \Rightarrow e : t_{\text{ret}} \dashv \mathcal{S}_1}{\Gamma, \mathcal{S}_0 \vdash \lambda x \,.\, eu \Rightarrow \lambda\,(x : u)\,.\, e : u \to t_{\text{ret}} \dashv \mathcal{S}_1} \text{ (Arr)}$$

Judgements of the form:

$$\mathcal{S}_0 \vdash t_1 == t_2 \dashv \mathcal{S}_1$$

$$\frac{}{\mathcal{S} \vdash t == t \dashv \mathcal{S}} \text{ (Base Eq)} \qquad \frac{\mathcal{S}_0 \vdash t_1 == t_3 \dashv \mathcal{S}_1 \qquad \mathcal{S}_1 \vdash t_2 == t_4 \dashv \mathcal{S}_2}{\mathcal{S}_0 \vdash t_1 \to t_2 == t_3 \to t_4 \dashv \mathcal{S}_2} \text{ (Arr Eq)}$$

$$\frac{u \notin t \qquad u \notin \mathcal{S}}{\mathcal{S} \vdash u == t \dashv \mathcal{S}, u = t} \text{ (UVar Left Eq)} \qquad \frac{u \notin t \qquad u \notin \mathcal{S}}{\mathcal{S} \vdash t == u \dashv \mathcal{S}, u = t} \text{ (UVar Right Eq)}$$

3 Theorems

Lemma 1 (Equality)**.** *If there exists a judgement*

$$\mathcal{S} \vdash t_1 == t_2 \dashv \mathcal{S}',$$

then under context $\mathcal{S}'$, the types t_1 and t_2 are equal.

Proof. TBD □

Lemma 2 (Inversion (Decl))**.** *If*

$$\Gamma \vdash e : t,$$

we may deduce what its derivation is unambiguously from the form of its expression e.

Proof. Follows from the definition. □

Lemma 3 (Inversion (Algorithmic))**.** *If there exists a judgement*

$$\Gamma, \mathcal{S} \vdash eu \Rightarrow e : t \dashv \mathcal{S}',$$

we may deduce what its derivation is unambiguously from the form of its expression eu.

Proof. Follows from the definition. □

Lemma 4 (Extension 1)**.** *If*

$$\mathcal{S}_0 \vdash t_1 == t_2 \dashv \mathcal{S}_1,$$

then $\mathcal{S}_0 \subseteq \mathcal{S}_1$.

Proof. We prove this by induction over the deductions.

The Base Eq judgement is one of our base cases, and is satisfied by the Refl rule of $\subseteq$. The other two base cases are UVar Left Eq and UVar Right Eq, which are satisfied by the Extension rule of $\subseteq$. Informally, we know that $\mathcal{S} \subseteq \mathcal{S}, u = t$ because the left is a strict prefix of the right and $\mathcal{S}_0 \subseteq \mathcal{S}_1$ means $\mathcal{S}_0$ is a prefix of $\mathcal{S}_1$.

The one recursive case is the Arr Eq judgement, reproduced below.

$$\frac{\mathcal{S}_0 \vdash t_1 == t_3 \dashv \mathcal{S}_1 \qquad \mathcal{S}_1 \vdash t_2 == t_4 \dashv \mathcal{S}_2}{\mathcal{S}_0 \vdash t_1 \to t_2 == t_3 \to t_4 \dashv \mathcal{S}_2} \text{ (Arr Eq)}$$

By inversion and the inductive hypothesis, $\mathcal{S}_0 \subseteq \mathcal{S}_1$ and $\mathcal{S}_1 \subseteq \mathcal{S}_2$. Because $\subseteq$ is transitive, we may conclude that $\mathcal{S}_0 \subseteq \mathcal{S}_2$, as desired.

By induction, this property holds for all judgements. □

Lemma 5 (Extension 2)**.** *If*

$$\Gamma, \mathcal{S} \vdash eu \Rightarrow e \,:\, t \dashv \mathcal{S}',$$

then $\mathcal{S} \subseteq \mathcal{S}'$.

Proof. We prove this by induction over the deductions.

Rules Unit and Var constitute base cases, and we observe that the Refl rule of $\subseteq$ satisfies the lemma.

There are two recursive cases: App and Arr. For Arr, we observe that the inductive hypothesis and inversion is sufficient to prove the lemma. For App, reproduced below, we will have to do a little work, starting with inversion.

$$\frac{\begin{array}{cc}\text{Fresh } u & \Gamma, \mathcal{S}_0 \vdash eu_1 \Rightarrow e_1 \,:\, t_{\text{arr}} \dashv \mathcal{S}_1 \\ \Gamma, \mathcal{S}_1 \vdash eu_2 \Rightarrow e_2 \,:\, t_{\text{arg}} \dashv \mathcal{S}_2 & \mathcal{S}_2 \vdash t_{\text{arr}} == t_{\text{arg}} \to u \dashv \mathcal{S}_3\end{array}}{\Gamma, \mathcal{S}_0 \vdash eu_1\ eu_2 \Rightarrow e_1\ e_2 \,:\, u \dashv \mathcal{S}_3} \text{ (App)}$$

We may apply the inductive hypothesis to conclude $\mathcal{S}_0 \subseteq \mathcal{S}_1$ and $\mathcal{S}_1 \subseteq \mathcal{S}_2$. We may apply the Extension 1 lemma to conclude $\mathcal{S}_2 \subseteq \mathcal{S}_3$. By the transitive property, we may conclude $\mathcal{S}_0 \subseteq \mathcal{S}_3$, as desired.

By induction, this property holds for all judgements. □

Lemma 6 (Substitution)**.** *If*

$$\Gamma \vdash e \,:\, t,$$

we may substitute every occurrence of the UVar u *with* t *and produce a valid judgement. Formally, we may derive the following judgement*

$$(\Gamma \vdash e \,:\, t)\,[u \mapsto t].$$

Proof. We observe that if u is not present in the judgement, the lemma holds trivially. Likewise, if u is in Γ but on an unused typing variable, the lemma

holds trivially. The rest of this proof will assume that u exists somewhere other than the typing context.

We prove it by induction on judgements.

Starting with base cases (and looking at their derivations, by inversion), DeclUnit holds as it is on a concrete type which may not be substituted. We consider now the other base case: DeclVar, reproduced below.

$$\frac{\Gamma(v) = u}{\Gamma \vdash v \,:\, u} \text{ (DeclVar)}$$

Note that we specify that $\Gamma(v) = u$, as the only meaningful place to substitute for u would be if it were a part of the resulting type. In this case, if we substitute u for t in Γ, it must be true that $\Gamma(v) = t$. From this, we may conclude that

$$\Gamma[u \mapsto t] \vdash v \,:\, t,$$

which is a less complex way of stating the desired (below).

$$\Gamma[u \mapsto t] \vdash v[u \mapsto t] \,:\, u[u \mapsto t]$$

Now for the recursive cases, DeclLam and DeclApp. We'll provide a thorough derivation of DeclLam; DeclApp follows in a similar manner. Suppose we have a judgement of the form

$$\Gamma \vdash \lambda\,(x \,:\, t_1)\,.\,e \,:\, t_1 \to t_2.$$

We'd like to know that substituting t for u in this judgement will produce another valid judgement. By inversion, we know the following derivation exists

$$\frac{\Gamma, x \,:\, t_1 \vdash e \,:\, t_2}{\Gamma \vdash \lambda\,(x \,:\, t_1)\,.\,e \,:\, t_1 \to t_2} \text{ (DeclLam).}$$

By the inductive hypothesis, we know that

$$(\Gamma, x \,:\, t_1 \vdash e \,:\, t_2)\,[u \mapsto t].$$

If we apply the substitution, we get

$$\Gamma[u \mapsto t], x \,:\, t_1[u \mapsto t] \vdash e[u \mapsto t] \,:\, t_2[u \mapsto t].$$

We may apply the DeclLam rule to this judgement to obtain

$$\frac{\Gamma[u \mapsto t], x \,:\, t_1[u \mapsto t] \vdash e[u \mapsto t] \,:\, t_2[u \mapsto t]}{\Gamma[u \mapsto t] \vdash \lambda\,(x \,:\, t_1[u \mapsto t])\,.\,e[u \mapsto t] \,:\, t_1[u \mapsto t] \to t_2[u \mapsto t]} \text{ (DeclLam).}$$

Skipping a few steps, we see that the resulting judgement is the same as

$$(\Gamma \vdash \lambda\,(x \,:\, t_1)\,.\,e \,:\, t_1 \to t_2)\,[u \mapsto t],$$

as desired.

DeclApp follows in a similar way, the general step being applying inversion, then the inductive hypothesis, then re-deriving the desired judgement.

By induction, this lemma holds for all valid judgements.

□

Lemma 7 (Weakening)**.** *If*

$$\mathcal{S}(\Gamma) \vdash \mathcal{S}(e) \ : \ \mathcal{S}(t)$$

and know that $\mathcal{S} \subseteq \mathcal{S}'$*, then we may conclude that*

$$\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e) \ : \ \mathcal{S}'(t).$$

Proof. We prove this lemma by induction on $\mathcal{S}'$.

We start with the base case where $\mathcal{S} = \mathcal{S}'$, which trivially holds.

Now assume that the judgement holds for all $\mathcal{S}'$ which are n elements larger than $\mathcal{S}$. Consider $\mathcal{S}_{n+1}$, which is $n+1$ elements larger than $\mathcal{S}$. We may assume that $\mathcal{S}_{n+1} = \mathcal{S}_n, u = t$, where we know by the inductive hypothesis that weakening holds for $\mathcal{S}_n$, i.e.

$$\mathcal{S}_n(\Gamma) \vdash \mathcal{S}_n(e) \ : \ \mathcal{S}_n(t).$$

Our goal is to derive the same for $\mathcal{S}_{n+1}$. We apply the Substitution lemma using the above, u, and t, to derive

$$(\mathcal{S}_n(\Gamma) \vdash \mathcal{S}_n(e) \ : \ \mathcal{S}_n(t))\,[u \mapsto t].$$

Since applying a store is the same thing as substitution, we may rewrite the above to the identical judgement

$$(\mathcal{S}_n, u = t)(\Gamma) \vdash (\mathcal{S}_n, u = t)(e) \ : \ (\mathcal{S}_n, u = t)(t).$$

$\mathcal{S}_n, u = t = \mathcal{S}_{n+1}$, proving the inductive step.

By induction, this lemma holds for all $\mathcal{S}' \supseteq \mathcal{S}$. □

Theorem 1 (Soundness)**.** *If*

$$\Gamma, \mathcal{S} \vdash eu \Rightarrow e \ : \ t \dashv \mathcal{S}'$$

then we may derive a declarative judgement

$$\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e) \ : \ \mathcal{S}'(t).$$

Proof. The proof is by induction over the expressions.

The base cases for the rules Unit and Var are trivial.

For App, we start with the given judgement, by inversion:

$$\frac{\begin{array}{cc} \text{Fresh } u & \Gamma, \mathcal{S}_0 \vdash eu_1 \Rightarrow e_1 \ : \ t_{\text{arr}} \dashv \mathcal{S}_1 \\ \Gamma, \mathcal{S}_1 \vdash eu_2 \Rightarrow e_2 \ : \ t_{\text{arg}} \dashv \mathcal{S}_2 & \mathcal{S}_2 \vdash t_{\text{arr}} == t_{\text{arg}} \to u \dashv \mathcal{S}' \end{array}}{\Gamma, \mathcal{S}_0 \vdash eu_1 \ eu_2 \Rightarrow e_1 \ e_2 \ : \ u \dashv \mathcal{S}'} \ (\text{App})$$

By induction, we know that there are declarative judgements

$$\mathcal{S}_1(\Gamma) \vdash \mathcal{S}_1(e_1) \,:\, \mathcal{S}_1(t_{\text{arr}})$$

and

$$\mathcal{S}_2(\Gamma) \vdash \mathcal{S}_2(e_2) \,:\, \mathcal{S}_2(t_{\text{arg}}).$$

We may apply the weakening lemma to conclude

$$\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_1) \,:\, \mathcal{S}'(t_{\text{arr}})$$

and

$$\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_2) \,:\, \mathcal{S}'(t_{\text{arg}}).$$

By the equality lemma, we know that $t_{\text{arr}} = t_{\text{arg}} \to u$, so we may substitute for it:

$$\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_1) \,:\, \mathcal{S}'(t_{\text{arg}} \to u).$$

By property of substitution, we get

$$\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_1) \,:\, \mathcal{S}'(t_{\text{arg}}) \to \mathcal{S}'(u).$$

Thus we may construct the declarative judgement

$$\frac{\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_1) \,:\, \mathcal{S}'(t_{\text{arg}}) \to \mathcal{S}'(u) \qquad \mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_2) \,:\, \mathcal{S}'(t_{\text{arg}})}{\mathcal{S}'(\Gamma) \vdash \mathcal{S}'(e_1)\ \mathcal{S}'(e_2) \,:\, \mathcal{S}'(u)}$$

By property of substitution, we know that $\mathcal{S}'(e_1)\ \mathcal{S}'(e_2)$ is the same as $\mathcal{S}'(e_1\ e_2)$, which gives the desired.

For Arr, we start with the given judgement, again by inversion:

$$\frac{\Gamma, x \,:\, u, \mathcal{S}_0 \vdash eu \Rightarrow e \,:\, t_{\text{ret}} \dashv \mathcal{S}'}{\Gamma, \mathcal{S}_0 \vdash \lambda x \,.\, eu \Rightarrow \lambda\,(x \,:\, u)\,.\, e \,:\, u \to t_{\text{ret}} \dashv \mathcal{S}'}\ (\text{Arr})$$

By induction, we can conclude that there exists a derivation for the declarative judgement

$$\mathcal{S}'(\Gamma), x \,:\, \mathcal{S}'(u) \vdash \mathcal{S}'(e) \,:\, \mathcal{S}'(t_{\text{ret}}).$$

From this, we may derive the declarative judgement

$$\frac{\mathcal{S}'(\Gamma), x \,:\, \mathcal{S}'(u) \vdash \mathcal{S}'(e) \,:\, \mathcal{S}'(t_{\text{ret}})}{\mathcal{S}'(\Gamma) \vdash \lambda x \,.\, \mathcal{S}'(e) \,:\, \mathcal{S}'(u) \to \mathcal{S}'(t_{\text{ret}})}$$

By the property of substitution, we know that $\mathcal{S}'(u) \to \mathcal{S}'(t_{\text{ret}})$ is the same as $\mathcal{S}'(u \to t_{\text{ret}})$, which gives the desired.

□

4 To Do

- Replace subseteq with square subseteq (and supseteq) - Replace occurrences of substitution $[u \mapsto t]$ with a singleton type store of $\cdot, u = t$. Have substitution lemma be $\mathcal{S}(\Gamma) \vdash \mathcal{S}(e) : \mathcal{S}t \Rightarrow (\mathcal{S}, u = t')(\Gamma) \vdash (\mathcal{S}, u = t')(e) : (\mathcal{S}, u = t')(t)$. - Clean up DeclVar case in Substitution. - Make substitution apply stores from beginning to end. (S,u1 = t')(t) := S(t)[u1 |-> t']. - See about writing up the judgements as witnesses in LH.

two options for proving infer works - Make 'infer' take Γ, $\mathcal{S}$, and eu as input and return e, t, $\mathcal{S}'$ and a witness that of the declarative judgement (from soundness). - Take the same inputs and only return the first three outputs. Write a proof that the function produces the delcarative judgement itself.