The last two pages of this document contain a reference on the syntax, operational semantics, type system, and (unary) logical relation for System F.

Problem 1. Prove (by induction on a thing of your choice) the following statement about System F's type system.

If $\cdot;\cdot \vdash e : \tau$, then τ is closed (that is, $\mathsf{FTV}(\tau) = \emptyset$).

Hint: This is similar to Problem 1 on Homework 3, except that we are talking about the free *type* variables of the *type* τ, rather than the free term variables of e.

Hint: You will need a similar generalization as you did with Problem 1 on Homework 3, except at the type variable level rather than the term variable level. There is also an additional complication due to the interaction between Γ and Δ in the typing judgment. Proceed carefully!

Lemma 0. forall Δ, and type variables α, β, if $\alpha \neq \beta$ and $\Delta, \alpha \vdash \beta$ then $\Delta \vdash \beta$.

Proof. By induction on $\Delta, \alpha \vdash \beta$.

- Case $\dfrac{\beta \in \Delta, \alpha}{\Delta, \alpha \vdash \beta}$. Since $\beta \neq \alpha$, it must be the case that $\beta \in \Delta$. Therefore, $\Delta \vdash \beta$.
- Other cases are trivial since β is not a type variable.

□

Lemma 1-0: forall Δ, α, β, if $\beta \in \mathsf{FTV}(\alpha)$ and $\Delta \vdash \alpha$ then $\Delta \vdash \beta$.

Proof. By induction on $\Delta \vdash \alpha$.

- Case $\dfrac{\alpha \in \Delta}{\Delta \vdash \alpha}$. In this case, $\mathsf{FTV}(\alpha) = \emptyset$, therefore this case is vacuous.
- Case $\dfrac{\Delta \vdash \tau_1 \qquad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \to \tau_2}$. $\beta \in \mathsf{FTV}(\tau_1) \cup \mathsf{FTV}(\tau_2)$, by case split on the occurence of β:
  (a) $\beta \in \mathsf{FTV}(\tau_1)$. According to the induction hypothesis, $\Delta \vdash \beta$
  (b) $\beta \in \mathsf{FTV}(\tau_2)$. Similar as previous case.
- Case $\dfrac{\alpha \notin \Delta \qquad \Delta, \alpha \vdash \tau}{\Delta \vdash \forall \alpha.\ \tau}$. Since $\beta \in \mathsf{FTV}(\forall \alpha.\ \tau)$, according to the definition of FTV, $\beta \in \mathsf{FTV}(\tau) - \{\alpha\}$. By monotonicity, $\beta \in \mathsf{FTV}(\tau)$. According to the induction hypothesis, $\Delta, \alpha \vdash \beta$. Since according to the definition of FTV, $\beta \neq \alpha$, therefore, according to *Lemma 0*, $\Delta \vdash \beta$.

□

Lemma 1-1: forall Δ, Γ, α, β, e, if $\beta \in \mathsf{FTV}(\alpha)$ and $\Delta; \Gamma \vdash e : \alpha$ then $\Delta \vdash \beta$

Proof. By induction on $\Delta; \Gamma \vdash e : \alpha$.

- Case $\dfrac{x \in \mathrm{dom}\ \Gamma \qquad \Gamma(x) = \alpha \qquad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : \alpha}$. In this case, α is a type variable, and $\mathsf{FTV}(\alpha) = \alpha$, thus $\beta = \alpha$. Since $\Delta \vdash \Gamma$ and $\Gamma(x) = \alpha$, $\Delta \vdash \alpha$ and thus $\Delta \vdash \beta$
- Case $\dfrac{\Delta \vdash \tau_1 \qquad \Delta; \Gamma[x \mapsto \tau_1] \vdash e : \tau_2}{\Delta; \Gamma \vdash \lambda x : \tau.\ e : \tau_1 \to \tau_2}$. According to the definition of FTV, $\beta \in \mathsf{FTV}(\tau_1) \cup \mathsf{FTV}(\tau_2)$. By case split on the occurence of β.

(a) Case $\beta \in \mathsf{FTV}(\tau_1)$. According to *Lemma 1-0*, $\Delta \vdash \beta$.

(b) Case $\beta \in \mathsf{FTV}(\tau_2)$. According to the induction hypothesis and *Lemma 1-0*, $\Delta \vdash \beta$.

- Case $\dfrac{\Delta; \Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Delta; \Gamma \vdash e_2 : \tau_1}{\Delta; \Gamma \vdash e_1\ e_2 : \tau_2}$. Since $\beta \in \mathsf{FTV}(\tau_2)$, $\beta \in \mathsf{FTV}(\tau_1) \cup \mathsf{FTV}(\tau_2)$. Therefore, according to the the induction hypothesis, $\Delta \vdash \beta$.
- Case $\dfrac{\Delta; \Gamma \vdash e : \forall \alpha.\ \tau}{\Delta; \Gamma \vdash e\ \tau_1 : \tau[\tau_1/\alpha]}$. Since there is no α in $\tau[\tau_1/\alpha]$, so $\beta \neq \alpha$. Moreover, $\beta \in \mathsf{FTV}(\tau[\tau_1/\alpha])$, so directly we can get $\beta \in \mathsf{FTV}(\tau) - \{\alpha\}$. Thus according to the induction hypothesis, $\Delta \vdash \beta$.

□

If $\cdot; \cdot \vdash e : \tau$, then $\mathsf{FTV}(\tau) = \emptyset$.

Proof. Suppose $\mathsf{FTV}(\tau) \neq \emptyset$, then $\exists \beta \in \mathsf{FTV}(\tau)$. According to *Lemma 1-1*, $\cdot \vdash \beta$. However, Δ in this case is an empty set, this contradicts with $\cdot \vdash \beta$. Therefore, $\mathsf{FTV}(\tau)$ must be $\emptyset$. □

Problem 2. Consider the following pseudo untyped λ-calculus program, which assumes a language with built-in integers, booleans, and pairs.

$$\lambda f.\ (f\ 0, f\ \mathtt{true})$$

(a) Suppose you transcribed this program into OCaml (no need to turn in any such transcription). Explain briefly and informally why it would not typecheck. (It's ok to base your explanation purely on your intuition about how the OCaml type system works, not on any formal system.)

(b) Show how to transcribe (by adding type annotations, type abstractions, and type applications) this program into System F (assume you have built-in integers, booleans, and pairs) such that it has the following type.

$$(\forall \alpha.\ \alpha \to \alpha) \to \mathtt{int} \times \mathtt{bool}$$

No need to prove formally that your transcription has this type. Just convince yourself.

(c) Using your understanding of parametricity, say what the "only" possible value to pass for f is in your transcription in part (b). No need to prove your answer.

(d) Show how to *again* transcribe (by adding type annotations, type abstractions, and type applications) this program into System F (assume you have built-in integers, booleans, and pairs) such that it has the following (different!) type.

$$\forall \alpha.\ (\forall \beta.\ \beta \to \alpha) \to \alpha \times \alpha$$

No need to prove formally that your transcription has this type. Just convince yourself.

(e) Using your understanding of parametricity, describe the possible values to pass in for f in your transcription form part (d). No need to prove your answer.

Hint: There are infinitely many, but they all have a clean description.

(f) Given your answer to part (e), what can you say about the pair returned by the System F program from part (d)? No need to prove your answer.

As an aside, the examples in this problem demonstrate the lack of "principle types" for System F. A principle type for an expression is its most general type, in the sense that if it has any other type, then it is a special case of its principle type. Principle types exist in ML, but not in System F, as demonstrated by this problem. The lack of principle types poses a serious difficulty to type inference, because it means there is no "best answer" to return for the type of an expression.

Problem 3. We will use a dot "$\cdot$" to represent an empty partial function for the ρ argument to R.

(a) Translate the meaning of $R^{\cdot}_{\forall\alpha.\,\alpha\to\alpha}$ into English. (You can use symbols in your English.)

(b) Show directly from the definition of R that

$$\Lambda\alpha.\,\lambda x:\alpha.\,x \quad\in\quad R^{\cdot}_{\forall\alpha.\,\alpha\to\alpha}$$

Problem 4. This question is about the definition of R itself, and specifically its "presupposition". A presupposition is kind of like a precondition, but on a mathematical object instead of a program. It means that the mathematical object doesn't make sense unless the presupposition is true. According to the last page of this document, the presupposition of R^{ρ}_{τ} is that $\mathsf{FTV}(\tau) \subseteq \mathrm{dom}\ \rho$.

(a) In the base case of the definition of R, when looking at a type variable α, we look up the type variable in ρ. Since ρ is a partial function, this only makes sense if $\alpha \in \mathrm{dom}\ \rho$. Prove in one short sentence that the presupposition of R guarantees $\alpha \in \mathrm{dom}\ \rho$.

(b) Since R is defined by recursion on τ, we should technically check that any recursive calls to R satisfy their presupposition, *assuming* the presupposition of the "outer" R. There are three recursive calls in the definition of R. Prove that each of them satisfy the presupposition.

Problem 5. In Homework 3 (programming part) we saw how to Church-encode pairs in the untyped λ-calculus, as follows

```
pair = \x. \y. \f. f x y
fst = \p. p (\x. \y. x)
```

This encoding can be typed in System F as follows. The type of pairs whose first components have type τ_1 and whose second components have type τ_2 will be *abbreviated* $\mathsf{Pair}\ \tau_1\ \tau_2$, which is defined as follows:

$$\mathsf{Pair}\ \tau_1\ \tau_2 = \forall\alpha.\,(\tau_1 \to \tau_2 \to \alpha) \to \alpha.$$

(a) The type of `pair` is then

$$\forall\alpha.\,\forall\beta.\,\alpha \to \beta \to \mathsf{Pair}\ \alpha\ \beta,$$

or, expanding the definition of Pair,

$$\forall\alpha.\,\forall\beta.\,\alpha \to \beta \to \forall\gamma.\,(\alpha \to \beta \to \gamma) \to \gamma.$$

Show how to transcribe the untyped program `pair` from Homework 3 given above into System F (by adding type annotations, type abstractions, and type applications) such that it has the above type. No need to formally prove it has the type. Just convince yourself.

(b) Similarly, the type of `fst` is then

$$\forall\alpha.\,\forall\beta.\,\mathsf{Pair}\ \alpha\ \beta \to \alpha,$$

or, expanding the definition of Pair,

$$\forall\alpha.\,\forall\beta.\,(\forall\gamma.\,(\alpha \to \beta \to \gamma) \to \gamma) \to \alpha,$$

Show how to transcribe the untyped program `fst` from Homework 3 given above into System F (by adding type annotations, type abstractions, and type applications) such that it has the above type. No need to formally prove it has the type. Just convince yourself.

(c) Prove directly using the operational semantics that, for any values $v_1 : \tau_1$ and $v_2 : \tau_2$,

$$\texttt{fst}\ \tau_1\ \tau_2\ (\texttt{pair}\ \tau_1\ \tau_2\ v_1\ v_2) \to^* v_1.$$

where `fst` and `pair` refer to your transcribed versions in System F.

(d) Now suppose p is *any* System F expression such that

$$\cdot;\cdot \vdash p : \forall\alpha.\,\forall\beta.\,\alpha \to \beta \to \forall\gamma.\,(\alpha \to \beta \to \gamma) \to \gamma.$$

In other words, p is just some program with the same type as `pair`. Similarly, suppose that f is some System F expression such that

$$\cdot;\cdot \vdash f : \forall\alpha.\,\forall\beta.\,(\forall\gamma.\,(\alpha \to \beta \to \gamma) \to \gamma) \to \alpha.$$

In other words, f has the same type as `fst`.

Use the fundamental theorem of the logical relation to prove that, for any values $v_1 : \tau_1$ and $v_2 : \tau_2$,

$$f\ \tau_1\ \tau_2\ (p\ \tau_1\ \tau_2\ v_1\ v_2) \to^* v_1.$$

The remaining problem is extra credit.

Problem 6. This extra credit problem considers adding a new expression to System F, called `choose`. The idea is that `choose` v_1 v_2 nondeterministically evaluates to either v_1 or v_2.

(a) Give operational semantics for `choose` that first evaluate its first argument to a value, then evaluate its second argument to a value, and then *either* evaluate to the first value *or* the second.

Hint: Use four rules. One to make recursive progress on the first argument, and a similar one for the second argument. Then one to "choose" the first value, and one to "choose" the second.

(b) Give a typing rule for `choose`.

Hint: Use one rule. It is vaguely similar to `if`, except there is no branch condition.

(c) Use `choose` to define a Church boolean that is not "equivalent" to true or false, in the sense that can return either its first argument or its second, and change its mind each time it's called.

(d) Prove that the fundamental theorem of the (unary) logical relation still holds on this extended language extending the proof with a case for `choose`.

(e) Explain how the existence of your program from part (c) does *not* contradict the result we proved on slide 13 of Lecture 15 about Church booleans using the (unary) logical relation.

(f) Extra extra credit (requires bonus material from Lecture 15 on binary logical relations). Explain how the existence of your program from part (c) *does* contradict the result we proved on slide 17 of Lecture 15 about Church booleans using the *binary* logical relation.

(g) Extra extra credit (requires bonus material from Lecture 15 on binary logical relations). Attempt to prove the case for `choose` in the fundamental theorem of the *binary* logical relation. Point to exactly where you get stuck.

System F

Syntax

$$
\begin{array}{rcl}
e & ::= & x \mid e\ e \mid \lambda x:\tau.\,e \mid \Lambda\alpha.\,e \mid e\ \tau \\
v & ::= & \lambda x.\,e \mid \Lambda\alpha.\,e \\
\tau & ::= & \alpha \mid \tau \to \tau \mid \forall\alpha.\tau \\
\Gamma & \in & \mathit{Var} \rightharpoonup \mathit{Type} \\
\Delta & \subseteq & \mathit{TyVar}
\end{array}
$$

$\boxed{e \to e}$

Operational Semantics

$$
\frac{e_1 \to e_1'}{e_1\ e_2 \to e_1'\ e_2} \qquad \frac{e_2 \to e_2'}{v_1\ e_2 \to v_1\ e_2'} \qquad \frac{}{(\lambda x:\tau.\,e)\ v \to e[v/x]} \qquad \frac{e \to e'}{e\ \tau \to e'\ \tau} \qquad \frac{}{(\Lambda\alpha.\,e)\ \tau \to e[\tau/\alpha]}
$$

$\boxed{e \to^* e}$

$$
\frac{}{e \to^* e} \qquad \frac{e_1 \to^* e_2 \qquad e_2 \to e_3}{e_1 \to^* e_3}
$$

$\boxed{e[e_1/x]}$

Substitution functions

$$
\begin{array}{rcll}
x[e_1/x] & = & e_1 & \\
y[e_1/x] & = & y & y \neq x \\
(e_2\ e_3)[e_1/x] & = & e_2[e_1/x]\ e_3[e_1/x] & \\
(\lambda y:\tau.\,e)[e_1/x] & = & \lambda y:\tau.\,e[e_1/x] & y \neq x \text{ and } y \notin FV(e_1) \\
(e\ \tau)[e_1/x] & = & e[e_1/x]\ \tau & \\
(\Lambda\alpha.\,e)[e_1/x] & = & \Lambda\alpha.\,e[e_1/x] & \alpha \notin FTV(e_1)
\end{array}
$$

$\boxed{e[\tau/\alpha]}$

$$
\begin{array}{rcll}
x[\tau/\alpha] & = & e_1 & \\
(e_2\ e_3)[\tau/\alpha] & = & e_2[\tau/\alpha]\ e_3[\tau/\alpha] & \\
(\lambda x:\tau_1.\,e)[\tau/\alpha] & = & \lambda x:\tau_1[\tau/\alpha].\,e[\tau/\alpha] & \\
(e\ \tau_1)[\tau/\alpha] & = & e[\tau/\alpha]\ \tau_1[\tau/\alpha] & \\
(\Lambda\beta.\,e)[\tau/\alpha] & = & \Lambda\beta.\,e[\tau/\alpha] & \beta \neq \alpha \text{ and } \beta \notin FTV(\tau)
\end{array}
$$

$\boxed{\tau[\tau_1/\alpha]}$

$$
\begin{array}{rcll}
\alpha[\tau_1/\alpha] & = & \tau_1 & \\
\beta[\tau_1/\alpha] & = & \beta & \beta \neq \alpha \\
(\tau_2 \to \tau_3)[\tau_1/\alpha] & = & \tau_2[\tau_1/\alpha] \to \tau_3[\tau_1/\alpha] & \\
(\forall\beta.\,\tau)[\tau_1/\alpha] & = & \forall\beta.\,\tau[\tau_1/\alpha] & \beta \neq \alpha \text{ and } \beta \notin FTV(\tau_1)
\end{array}
$$

$\boxed{FTV(\tau)}$

Free type variables of a type or expression

$$
\begin{array}{rcl}
FTV(\alpha) & = & \{\alpha\} \\
FTV(\tau_1 \to \tau_2) & = & FTV(\tau_1) \cup FTV(\tau_2) \\
FTV(\forall\alpha.\,\tau) & = & FTV(\tau) - \{\alpha\}
\end{array}
$$

$\boxed{FTV(e)}$

Note that we overload FTV on expressions and types.

$$
\begin{array}{rcl}
FTV(x) & = & \emptyset \\
FTV(\lambda x:\tau.\,e) & = & FTV(\tau) \cup FTV(e) \\
FTV(e_1\ e_2) & = & FTV(e_1) \cup FTV(e_2) \\
FTV(\Lambda\alpha.\,e) & = & FTV(e) - \{\alpha\} \\
FTV(e\ \tau) & = & FTV(e) \cup FTV(\tau)
\end{array}
$$

$\boxed{\mathrm{FV}(e)}$ Free variables of an expression

$$\begin{array}{rcl} \mathrm{FV}(x) & = & \{x\} \\ \mathrm{FV}(\lambda x:\tau.\, e) & = & \mathrm{FV}(e) - \{x\} \\ \mathrm{FV}(e_1\ e_2) & = & \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\ \mathrm{FV}(\Lambda\alpha.\, e) & = & \mathrm{FV}(e) \\ \mathrm{FV}(e\ \tau) & = & \mathrm{FV}(e) \end{array}$$

$\boxed{\Delta;\Gamma \vdash e:\tau}$ Type System

$$\frac{x \in \mathrm{dom}\ \Gamma \qquad \Gamma(x) = \tau}{\Delta;\Gamma \vdash x:\tau}$$

$$\frac{\Delta \vdash \tau_1 \qquad \Delta;\Gamma[x \mapsto \tau_1] \vdash e:\tau_2}{\Delta;\Gamma \vdash \lambda x:\tau.\, e:\tau_1 \to \tau_2} \qquad \frac{\Delta;\Gamma \vdash e_1:\tau_1 \to \tau_2 \qquad \Delta;\Gamma \vdash e_2:\tau_1}{\Delta;\Gamma \vdash e_1\ e_2:\tau_2}$$

$$\frac{\alpha \notin \Delta \qquad \Delta,\alpha;\Gamma \vdash e:\tau}{\Delta;\Gamma \vdash \Lambda\alpha.\, e:\forall\alpha.\,\tau} \qquad \frac{\Delta \vdash \tau_1 \qquad \Delta;\Gamma \vdash e:\forall\alpha.\,\tau}{\Delta;\Gamma \vdash e\ \tau_1:\tau[\tau_1/\alpha]}$$

$\boxed{\Delta \vdash \Gamma}$

$$\Delta \vdash \Gamma = \forall x \in \mathrm{dom}\ \Gamma.\ \Delta \vdash \Gamma(x)$$

$\boxed{\Delta \vdash \tau}$

$$\frac{\alpha \in \Delta}{\Delta \vdash \alpha} \qquad \frac{\Delta \vdash \tau_1 \qquad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \to \tau_2} \qquad \frac{\Delta,\alpha \vdash \tau}{\Delta \vdash \forall\alpha.\,\tau}$$

Preparation for the definition of the logical relation

$$\begin{array}{rcl} \mathit{Spec} & = & \{S \subseteq \mathit{Val} \mid \forall e \in S.\ e \text{ closed}\} \\ \rho & \in & \mathit{TyVar} \rightharpoonup \mathit{Spec} \\ \gamma & \in & \mathit{Var} \rightharpoonup \mathit{Val} \\ \mathrm{T}(S) & = & \{e \mid \exists v.\ e \to^* v \wedge v \in S\} \end{array}$$

Definition of the logical relation on closed terms

$\boxed{R^\rho_\tau}$ presupposes $\mathrm{FTV}(\tau) \subseteq \mathrm{dom}\ \rho$

$$\begin{array}{rcl} R^\rho_\alpha & = & \rho(\alpha) \\ R^\rho_{\tau_1 \to \tau_2} & = & \{\lambda x.\, e \mid \forall v \in R^\rho_{\tau_1}.\ e[x/v] \in \mathrm{T}(R^\rho_{\tau_2})\} \\ R^\rho_{\forall\alpha.\,\tau} & = & \{\Lambda\alpha.\, e \mid \forall S \in \mathit{Spec}.\ e \in \mathrm{T}(R^{\rho[\alpha \mapsto S]}_\tau)\} \end{array}$$

$\boxed{e[\gamma]}$ Multisubstitution

$$\begin{array}{rcll} x[\gamma] & = & \gamma(x) & \text{if } x \in \mathrm{dom}\ \gamma \\ x[\gamma] & = & x & \text{if } x \notin \mathrm{dom}\ \gamma \\ (e_2\ e_3)[\gamma] & = & e_2[\gamma]\ e_3[\gamma] & \\ (\lambda x:\tau.\, e)[\gamma] & = & \lambda x:\tau.\, e[\gamma] & x \notin \mathrm{dom}\ \gamma \text{ and } \forall y \in \mathrm{dom}\ \gamma.\ x \notin \mathrm{FV}(\gamma(y)) \\ (e\ \tau)[\gamma] & = & e[\gamma]\ \tau & \\ (\Lambda\alpha.\, e)[\gamma] & = & \Lambda\alpha.\, e[\gamma] & \forall x \in \mathrm{dom}\ \gamma.\ \alpha \notin \mathrm{FTV}(\gamma(x)) \end{array}$$

Preparation and definition of the open logical relation

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

$$\rho;\Gamma \vdash \gamma = \forall x \in \mathrm{dom}\ \Gamma.\ x \in \mathrm{dom}\ \gamma \wedge \gamma(x) \in R^\rho_{\Gamma(x)}$$

$\boxed{\Delta;\Gamma \vDash e:\tau}$

$$\Delta;\Gamma \vDash e:\tau \quad = \quad \forall\rho.\ \mathrm{dom}\ \Delta \subseteq \mathrm{dom}\ \rho \Rightarrow \forall\gamma.\ \rho;\Gamma \vdash \gamma \Rightarrow e[\gamma] \in \mathrm{T}(R^\rho_\tau)$$

Theorem 1 (Fundamental theorem of the logical relation)**.** If $\Delta;\Gamma \vdash e:\tau$ then $\Delta;\Gamma \vDash e:\tau$.