Excercises

Chapter 2

Sean Li [1]

1. Reducted

Definition Some rules for reference.

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\,(\text{T-Var}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\,(\text{T-App})$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau}\,(\text{T-Abst})$$

In this document, convention is that all type judgements in a proof tree, unless stated otherwise, is derived from a single context per tree.

Problem

(2.1) Type the following terms

$$xxy \quad xyy \quad xyx \quad x(xy) \quad x(yx)$$

Solution. The first term cannot be typed.

Proof. $xxy = (xx)y$. Therefore, x is a function type, denote it as $\tau \to \sigma$. By the application rule, a subterm applied to x must be of τ, which means that the application xx is not legally typed. ■

The second one is typable where $x : \tau \to \tau \to \sigma$ and $y : \tau$.

1. $x : \tau \to \tau \to \sigma \quad \dashv \Gamma$
2. $y : \tau \quad \dashv \Gamma$

3. $xy : \tau \to \sigma$ **1,2 T-App**

4. $xyy : \sigma$ **3,2 T-App**

The third term is not typable.

Proof. Assume $xyx = (xy)x$ is typable. Therefore, $x : \tau$ where $\tau = \sigma \to \tau \to \alpha$ and $y : \sigma$. One can construct an infinite chain of function type by substituting τ: $\tau = \sigma \to (\sigma \to (\sigma \to ... \to \alpha) \to \alpha) \to \alpha$. By induction, it can be proven that only lambda abstractions can construct function types, meaning that the term is of form

$$(\lambda n : \tau.\lambda m : \tau....(\lambda a : \sigma.\lambda b : \sigma....))y(\lambda n : \tau.\lambda m : \tau....(\lambda a : \sigma.\lambda b : \sigma....))$$

meaning that an infinite reduction path is needed. This is impossible in STLC. ∎

The fourth type is typable where $x : (\tau \to \tau)$ and $y : \tau$.

1. $x : \tau \to \tau$ $\dashv \Gamma$
2. $y : \tau$ $\dashv \Gamma$
3. $xy : \tau$ **1,2 T-App**
4. $x(xy) : \tau$ **1,3 T-App**

The fifth term is typable where $x : (\tau \to \sigma)$ and $y : (\tau \to \sigma) \to \tau$:

1. $x : \tau \to \sigma$ $\dashv \Gamma$
2. $y : (\tau \to \sigma) \to \tau$ $\dashv \Gamma$
3. $yx : \tau$ **2,1 T-App**
4. $x(yx) : \sigma$ **1,3 T-App**

Problem

Find types for zero, one, and two

Solution. Term for zero is

$$\text{zero} := \lambda f x.x$$

Here x is only used as a

$$\text{zero} := \lambda f : \alpha.\lambda x : \beta.x$$

Type derivation shown as below:

1. $f : \alpha$ **Bound**
2. | $x : \beta$ **Bound**

3.	$x : \beta$	**T-Var**
4.	$\lambda x.x : \beta \to \beta$	**3 T-Abst**
5.	$\lambda f : \alpha.x : \beta.x : \alpha \to \beta \to \beta$	**4 T-Abst**

Term for one is

$$\text{one} \coloneqq \lambda fx.fx$$

Let f be an arbitrary function type that consumes x

$$\text{one} \coloneqq \lambda f : \alpha \to \beta.x : \alpha.fx$$

Type derivation shown as below

1.	$f : \alpha \to \beta$	**Bound**
2.	$x : \alpha$	**Bound**
3.	$f : \alpha \to \beta$	**T-Var**
4.	$x : \alpha$	**T-Var**
5.	$fx : \beta$	**3,4 T-App**
6.	$\lambda x.fx : \alpha \to \beta$	**5 T-Abst**
7.	$\lambda f : \alpha \to \beta.x : \alpha.fx : (\alpha \to \beta) \to \alpha \to \beta$	**6 T-Abst**

Same type signatures can be given to two

$$\text{two} \coloneqq \lambda f : \alpha \to \beta.\lambda x : \alpha.ffx$$

Type derivation shown as below

1.	$f : \alpha \to \beta$	**Bound**
2.	$x : \alpha$	**Bound**
3.	$f : \alpha \to \beta$	**T-Var**
4.	$x : \alpha$	**T-Var**
5.	$fx : \beta$	**3,4 T-App**
6.	$ffx : \beta$	**3,5 T-App**
7.	$\lambda x.ffx : \alpha \to \beta$	**6 T-Abst**
8.	$\lambda f : \alpha \to \beta.x : \alpha.ffx : (\alpha \to \beta) \to \alpha \to \beta$	**7 T-Abst**

Problem

Find types for

$$K := \lambda xy.x$$
$$S := \lambda xyz.xz(yz)$$

Solution. There are no occurences of application in K's subterms. Therefore all its binding variables could be given a simple base type.

$$K := \lambda x : \alpha.\lambda y : \beta.x$$

Type derivation shown as below

1.	$x : \alpha$	**Bound**
2.	$y : \beta$	**Bound**
3.	$x : \alpha$	**T-Var**
4.	$\lambda y : \beta.x : \beta \to \alpha$	**3 T-Abst**
5.	$\lambda x : \alpha.\lambda y : \beta.x : \alpha \to \beta \to \alpha$	**4 T-Abst**

For the S combinator, no term was applied to z. Therefore it can be given a simple base type α. As z was applied to y, it implies that $y : \alpha \to \beta$ for some output type β. As x takes z and (yz), it must be of type $\alpha \to \beta \to \delta$.

$$S := \lambda x : \alpha \to \beta \to \delta.\lambda y : \alpha \to \beta.\lambda z.\alpha.xz(yz)$$

Complete type derivation shown as below:

1.	$x : \alpha \to \beta \to \delta$	**Bound**
2.	$y : \alpha \to \beta$	**Bound**
3.	$z : \alpha$	**Bound**
4.	$y : \alpha \to \beta$	**T-Var**
5.	$z : \alpha$	**T-Var**
6.	$yz : \beta$	**4,5 T-App**
7.	$x : \alpha \to \beta \to \delta$	**T-Var**
8.	$xz : \beta \to \delta$	**7,5 T-App**
9.	$xz(yz) : \delta$	**8,6 T-App**
10.	$\lambda z : \alpha.xz(yz) : \alpha \to \delta$	**9 T-Abstr**
11.	$\lambda y : \alpha \to \beta.\lambda z.\alpha.xz(yz) : (\alpha \to \beta) \to \alpha \to \delta$	**10 T-Abstr**

12.

$$\lambda x : \alpha \to \beta \to \delta . \lambda y : \alpha \to \beta . \lambda z$$

$: \alpha . xz(yz) : (\alpha \to \beta \to \delta) \to (\alpha \to \beta) \to \alpha \to \delta$ **11 T-Abstr**

Problem

Type the bound variables

$$\lambda xyz.x(yz)$$

$$\lambda xyz.y(xz)z$$

Solution. For the first term, z had nothing applied to it. Therefore it could be given a simple base type α. z was applied to y, therefore $y : \alpha \to \beta$ to satisfy the application rule. Because the application yielded a type of β, by the application rule $x : \beta \to \delta$ for some type δ.

$$\lambda x : \beta \to \delta . \lambda y : \alpha \to \beta . \lambda z : \alpha . x(yz)$$

Complete type derivation shown below

1.	$x : \beta \to \delta$	**Bound**
2.	$y : \alpha \to \beta$	**Bound**
3.	$z : \alpha$	**Bound**
4.	$y : \alpha \to \beta$	**T-Var**
5.	$z : \alpha$	**T-Var**
6.	$yz : \beta$	**4,5 T-App**
7.	$x : \beta \to \delta$	**T-Var**
8.	$x(yz) : \delta$	**7,6 T-App**
9.	$\lambda z : \alpha . x(yz) : \alpha \to \delta$	**8 T-Abst**
10.	$\lambda y : \alpha \to \beta . \lambda z : \alpha . x(yz) : (\alpha \to \beta) \to \alpha \to \delta$	**9 T-Abst**
11.		

$\lambda x : \beta \to \delta . \lambda y : \alpha \to \beta . \lambda z : \alpha . x(yz)$

$: (\beta \to \delta) \to (\alpha \to \beta) \to \alpha \to \delta$ **10 T-Abst**

In the second term z could still be given a simple base type $z : \alpha$. Therefore $x : \alpha \to \beta$ for some type β. y takes $xz : \beta$ and $z : \alpha$, therefore it is of type $y : \beta \to \alpha \to \delta$ for some δ.

$$\lambda x : \alpha \to \beta . \lambda y : \beta \to \alpha \to \delta . \lambda z : \alpha . y(xz)z$$

. Complete type derivation shown below

1.	$x : \alpha \to \beta$	**Bound**
2.	$y : \beta \to \alpha \to \delta$	**Bound**
3.	$z : \alpha$	**Bound**
4.	$x : \alpha \to \beta$	**T-Var**
5.	$z : \alpha$	**T-Var**
6.	$xz : \beta$	**4,5 T-App**
7.	$y : \beta \to \alpha \to \delta$	**T-Var**
8.	$y(xz) : \alpha \to \delta$	**7,6 T-App**
9.	$y(xz)z : \delta$	**8,5 T-App**
10.	$\lambda z : \alpha . y(xz)z : \alpha \to \delta$	**9 T-Abst**
11.	$\lambda y : \beta \to \alpha \to \delta . \lambda z . y(xz)z : (\beta \to \alpha \to \delta) \to \alpha \to \delta$	**10 T-Abst**
12.		

$\lambda x : \alpha \to \beta . \lambda y : \beta \to \alpha \to \delta . \lambda z . y(xz)z :$
$(\alpha \to \beta) \to (\beta \to \alpha \to \delta) \to \alpha \to \delta$ **11 T-Abst**

Problem

Try to type the following terms, and prove if not typable.

$$\lambda xy . x(\lambda z . y)y$$
$$\lambda xy . x(\lambda z . x)y.$$

Solution. The first term is trivially typable.

1.	$x : (\delta \to \alpha) \to \alpha \to \beta$	**Bound**
2.	$y : \alpha$	**Bound**
3.	$x : (\delta \to \alpha) \to \alpha \to \beta$	**T-Var**
4.	$z : \delta$	**Bound**
5.	$y : \alpha$	**T-Var**
6.	$\lambda z : \delta . y : \delta \to \alpha$	**5 T-Abst**
7.	$x(\lambda z : \delta . y) : \alpha \to \beta$	**3,6 T-App**
8.	$y : \alpha$	**T-Var**
9.	$x(\lambda z : \delta . y)y : \beta$	**7,8 T-App**
10.	$\lambda y : \alpha . x(\lambda z : \delta . y)y : \alpha \to \beta$	**9 T-Abst**

11.

$$\lambda x : ((\delta \to \alpha) \to \alpha \to \beta)\lambda y : \alpha.x(\lambda z : \delta.y)y$$
$$: ((\delta \to \alpha) \to \alpha \to \beta) \to \alpha \to \beta \qquad \textbf{10 T-Abst}$$

The second term is not typable in STLC.

Proof. By induction on the type inference rule that constructed the type judgement for subterm $x(\lambda z.x)$. Because the term is an application, the only rule that applies is the application rule.

We denote the context inside the abstraction as Γ'. Suppose $\mathcal{J} \equiv \Gamma' \vdash x(\lambda z.x) : \tau$. By the inference rule of application, x must be a function type that accepts the type of $(\lambda z.x)$. Let $\Gamma' \vdash z : \alpha$, and type of x as τ_x. Therefore, $\Gamma' \vdash \lambda z : \alpha.x : \alpha \to \tau_x$. Therefore, $\tau_x = (\alpha \to \tau_x) \to \tau$. This is a recursive type, which is not constructable as it requires infinitely nested lambda abstractions that requires infinite reduction paths to reach a normal form. ■

Problem

Prove the pretyped term below is legal.

$$\lambda x : ((\alpha \to \beta) \to \alpha).x(\lambda z : \alpha.y)$$

Using the tree format and the flag format.

Solution. We suppose a context $\Gamma \vdash y : \beta$ that obviously exists.

Proof.

$$\dfrac{\dfrac{\dfrac{}{x : (\alpha \to \beta) \to \alpha}\,(\text{Bound}) \qquad \dfrac{\Gamma, z : \alpha \vdash y : \beta}{\Gamma \vdash (\lambda z : \alpha.y) : \alpha \to \beta}\,(\text{T-Abst})}{\Gamma, x : (\alpha \to \beta) \to \alpha \vdash (x(\lambda z : \alpha.y)) : \alpha}\,(\text{T-App})}{\Gamma \vdash \lambda x : ((\alpha \to \beta) \to \alpha).x(\lambda z : \alpha.y) : ((\alpha \to \beta) \to \beta) \to \alpha}\,(\text{T-Abst})$$

A valid type could be given to the term. Therefore, the term is typable under an existing context. ■

The flag derivation is given below:

1.	$x : (\alpha \to \beta) \to \alpha$	**Bound**
2.	$z : \alpha$	**Bound**
3.	$y : \beta$	$\dashv \Gamma$
4.	$(\lambda z : \alpha.y) : \alpha \to \beta$	**3 T-Abst**
5.	$x : (\alpha \to \beta) \to \alpha$	**T-Var**

6. $x(\lambda z : \alpha . y) : \beta$ **5,4 T-App**

7.

$\lambda x : ((\alpha \to \beta) \to \beta) . x(\lambda z : \alpha . y)$
$: (\alpha \to \beta) \to \beta \to \alpha$ **6 T-Abst**

Problem

Derive

$$f : A \to B \wedge g : B \to C \Rightarrow g \circ f : A \to C$$

Using the rules

$$\frac{f : A \to B, x \in A}{f(x) \in B} \text{(F-App)} \qquad \frac{\forall x \in A, f(x) \in B}{f : A \to B} \text{(F-Abst)}$$

Solution.

Proof.

1. $f : A \to B \wedge g : B \to C$ **Assumption**
2. $f : A \to B$ $1 \wedge E$
3. $g : B \to C$ $1 \wedge E$
4. $a \in A$
5. $f(a) \in B$ **3, 4 F-App**
6. $g(f(a)) \in C$ **5, 4 F-App**
7. $(g \circ f)(a) \in C$ **6 Compose Def**
8. $\forall x \in A, (g \circ f)(x) \in C$ $7 \forall E$
9. $g \circ f : A \to C$ **8 F-Abst**
10. $f : A \to B, g : B \to C \Rightarrow g \circ f : A \to C$ $9 \Rightarrow I$

■

Problem

Give a derivation in natural deduction of the following:

$$(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$$

Using the rules

$$\frac{A \Rightarrow B \qquad A}{B}\ (\Rightarrow E)$$

1. A **Premise**
2. $\quad \ldots$
3. $\quad B$

$$\overline{\quad A \Rightarrow B \quad}\ (\Rightarrow I)$$

Solution.

Proof.

1.	$A \Rightarrow B$	**Premise**
2.	$\quad B \Rightarrow C$	**Premise**
3.	$\quad\quad A$	**Premise**
4.	$\quad\quad\quad B$	$1, 3 \Rightarrow E$
5.	$\quad\quad\quad C$	$2, 4 \Rightarrow E$
6.	$\quad\quad A \Rightarrow C$	3-5 $\Rightarrow I$
7.	$\quad (B \Rightarrow C) \Rightarrow (A \Rightarrow C)$	2-6 $\Rightarrow I$
8.	$(A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$	1-7 $\Rightarrow I$

■

Problem

Prove the following pre-typed term is legal using flag notation

$$\lambda z : \alpha . y(xz)$$

Solution.

Proof. Let $\Gamma \vdash x : \alpha \to \beta, y : \beta \to \delta$ for some type β and δ.

1.	$z : \alpha$	**Bound**
2.	$\quad x : \alpha \to \beta$	$\dashv \Gamma$
3.	$\quad z : \alpha$	**T-Var**

4.	$xz : \beta$	**2,3 T-App**
5.	$y : \beta \to \delta$	$\dashv \Gamma$
6.	$y(xz) : \delta$	**5,4 T-App**
7.	$\lambda z : \alpha . y(xz) : \alpha \to \delta$	**6 T-Abst**

∎

Problem

State the similarity between Q. 2.7 (a), (b), and (c).

Solution. All of these examples requires proving something about composing two maps together as like this:

$$\begin{tikzcd}
B \arrow[r, "g"] & C \\
A \arrow[u, "f"] \arrow[ur, "g \circ f"] &
\end{tikzcd}$$

Problem

Pre-type the bounding variables for the following term

$$\lambda xy.y(\lambda z.yx) : (\gamma \to \beta) \to ((\gamma \to \beta) \to \beta) \to \beta$$

Solution.

$$\lambda x : (\gamma \to \beta).y : ((\gamma \to \beta) \to \beta).y(\lambda z : \gamma.yx)$$

Problem