

# EXERCISES

## CHAPTER 2

SEAN LI <sup>1</sup>

### 1. Reducted

---

**Definition** Some rules for reference.

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

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

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

Type the following terms

$xx y \quad xy y \quad xy x \quad x(xy) \quad x(yx)$

*Solution.* The first term cannot be typed.

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

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

1.  $x : \tau \rightarrow \tau \rightarrow \sigma \quad \neg \Gamma$

2.  $y : \tau \quad \neg \Gamma$

3.  $xy : \tau \rightarrow \sigma$       **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  $\tau = \sigma \rightarrow \tau \rightarrow \alpha$  and  $y : \sigma$ . One can construct an infinite chain of function type by substituting  $\tau$ :  $\tau = \sigma \rightarrow (\sigma \rightarrow (\sigma \rightarrow \dots \rightarrow \alpha) \rightarrow \alpha) \rightarrow \alpha$ . By induction, it can be proven that only lambda abstractions can construct function types, meaning that the term is of form

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

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

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

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

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

1.  $x : \tau \rightarrow \sigma$        $\vdash \Gamma$
2.  $y : (\tau \rightarrow \sigma) \rightarrow \tau$        $\vdash \Gamma$
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.  $\mid x : \beta$       **Bound**

3.  $\frac{}{x : \beta}$  **T-Var**
4.  $\frac{}{\lambda x.x : \beta \rightarrow \beta}$  **3 T-Abst**
5.  $\lambda f : \alpha.x : \beta.x : \alpha \rightarrow \beta \rightarrow \beta$  **4 T-Abst**

Term for one is

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

Let  $f$  be an arbitrary function type that consumes  $x$

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

Type derivation shown as below

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

Same type signatures can be given to two

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

Type derivation shown as below

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

## Problem

Find types for

$$K := \lambda xy.x$$

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

*Solution.* There are no occurrences 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.	$x : \alpha$	<b>Bound</b>
2.	$y : \beta$	<b>Bound</b>
3.	$x : \alpha$	<b>T-Var</b>
4.	$\lambda y : \beta. x : \beta \rightarrow \alpha$	<b>3 T-Abst</b>
5.	$\lambda x : \alpha. \lambda y : \beta. x : \alpha \rightarrow \beta \rightarrow \alpha$	<b>4 T-Abst</b>

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

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

Complete type derivation shown as below:

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

12.

$$\lambda x : \alpha \rightarrow \beta \rightarrow \delta. \lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. xz(yz) : (\alpha \rightarrow \beta \rightarrow \delta) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \delta \quad \mathbf{11 \ T-Abstr}$$

### 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  $\alpha$ .  $z$  was applied to  $y$ , therefore  $y : \alpha \rightarrow \beta$  to satisfy the application rule. Because the application yielded a type of  $\beta$ , by the application rule  $x : \beta \rightarrow \delta$  for some type  $\delta$ .

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

Complete type derivation shown below

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

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

$$: (\beta \rightarrow \delta) \rightarrow (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \delta \quad \mathbf{10 \ T-Abst}$$

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

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

. Complete type derivation shown below

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

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

$(\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \alpha \rightarrow \delta) \rightarrow \alpha \rightarrow \delta$

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

11.

$$\begin{aligned} \lambda x : ((\delta \rightarrow \alpha) \rightarrow \alpha \rightarrow \beta) \lambda y : \alpha. x(\lambda z : \delta. y) y \\ : ((\delta \rightarrow \alpha) \rightarrow \alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta \end{aligned} \quad \mathbf{10\ T-Abst}$$

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  $\Gamma'$ . 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  $\tau_x$ . Therefore,  $\Gamma' \vdash \lambda z : \alpha. x : \alpha \rightarrow \tau_x$ . Therefore,  $\tau_x = (\alpha \rightarrow \tau_x) \rightarrow \tau$ . 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.

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

Using the tree format and the flag format.

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

*Proof.*

$$\frac{\frac{}{x : (\alpha \rightarrow \beta) \rightarrow \alpha} \text{(Bound)} \quad \frac{\Gamma, z : \alpha \vdash y : \beta}{\Gamma \vdash (\lambda z : \alpha. y) : \alpha \rightarrow \beta} \text{(T-Abst)}}{\Gamma \vdash (\lambda z : \alpha. y) : \alpha \rightarrow \beta} \text{(T-App)} \quad \frac{\Gamma, x : (\alpha \rightarrow \beta) \rightarrow \alpha \vdash (x(\lambda z : \alpha. y)) : \alpha}{\Gamma \vdash \lambda x : ((\alpha \rightarrow \beta) \rightarrow \alpha). x(\lambda z : \alpha. y) : ((\alpha \rightarrow \beta) \rightarrow \beta) \rightarrow \alpha} \text{(T-Abst)}$$

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.	$x : (\alpha \rightarrow \beta) \rightarrow \alpha$	<b>Bound</b>
2.	$z : \alpha$	<b>Bound</b>
3.	$y : \beta$	$\dashv \Gamma$
4.	$(\lambda z : \alpha. y) : \alpha \rightarrow \beta$	<b>3 T-Abst</b>
5.	$x : (\alpha \rightarrow \beta) \rightarrow \alpha$	<b>T-Var</b>

6.  $\boxed{x(\lambda z : \alpha.y) : \beta}$  **5,4 T-App**
- 7.
- $\lambda x : ((\alpha \rightarrow \beta) \rightarrow \beta).x(\lambda z : \alpha.y)$
- $: (\alpha \rightarrow \beta) \rightarrow \beta \rightarrow \alpha$  **6 T-Abst**

### Problem

Derive

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

Using the rules

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

*Solution.*

*Proof.*

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

■



### Problem

Give a derivation in natural deduction of the following:

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

Using the rules

$$\frac{A \Rightarrow B \quad A}{B} (\Rightarrow E) \qquad \frac{\begin{array}{l} 1. \quad A \quad \text{Premise} \\ 2. \quad \vdots \\ 3. \quad B \end{array}}{A \Rightarrow B} (\Rightarrow I)$$

*Solution.*

*Proof.*

$$\begin{array}{ll} 1. & A \Rightarrow B \quad \text{Premise} \\ 2. & \begin{array}{|l} B \Rightarrow C \end{array} \quad \text{Premise} \\ 3. & \begin{array}{|l|l} A \end{array} \quad \text{Premise} \\ 4. & \begin{array}{|l|l|l} B \end{array} \quad 1, 3 \Rightarrow E \\ 5. & \begin{array}{|l|l|l} C \end{array} \quad 2, 4 \Rightarrow E \\ 6. & \begin{array}{|l|l} A \Rightarrow C \end{array} \quad 3-5 \Rightarrow I \\ 7. & \begin{array}{|l} (B \Rightarrow C) \Rightarrow (A \Rightarrow C) \end{array} \quad 2-6 \Rightarrow I \\ 8. & (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C)) \quad 1-7 \Rightarrow I \end{array}$$

■

### Problem

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

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

*Solution.*

*Proof.* Let  $\Gamma \vdash x : \alpha \rightarrow \beta, y : \beta \rightarrow \delta$  for some type  $\beta$  and  $\delta$ .

$$\begin{array}{ll} 1. & z : \alpha \quad \text{Bound} \\ 2. & \begin{array}{|l} x : \alpha \rightarrow \beta \end{array} \quad \neg \Gamma \\ 3. & \begin{array}{|l} z : \alpha \end{array} \quad \text{T-Var} \end{array}$$

- |    |   |                  |
|----|---|------------------|
| 4. | $xz : \beta$  | <b>2,3 T-App</b> |
| 5. | $y : \beta \rightarrow \delta$                          | $\vdash \Gamma$  |
| 6. | $y(xz) : \delta$  | <b>5,4 T-App</b> |
| 7. | $\lambda z : \alpha. y(xz) : \alpha \rightarrow \delta$ | <b>6 T-Abst</b>  |

■

### 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:

