

EXERCISES

CHAPTER 2

SEAN LI ¹

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

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

Solution. The first term cannot be typed.

Proof. $xxy = (xx)y$. Therefore, x is a function type, denote it as $\tau \rightarrow \sigma$. 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 $x : \tau \rightarrow \tau \rightarrow \sigma$ and $y : \tau$.

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

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 τ : $\tau = \sigma \rightarrow (\sigma \rightarrow (\sigma \rightarrow \dots \rightarrow \alpha) \rightarrow \alpha) \rightarrow \alpha$. 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. $x : \tau \rightarrow \tau$ $\dashv \Gamma$
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 $x : (\tau \rightarrow \sigma)$ and $y : (\tau \rightarrow \sigma) \rightarrow \tau$:

1. $x : \tau \rightarrow \sigma$ $\dashv \Gamma$
2. $y : (\tau \rightarrow \sigma) \rightarrow \tau$ $\dashv \Gamma$
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.	$\lambda x.x : \beta \rightarrow \beta$	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.	$fx : \beta$	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.	$fx : \beta$	3,4 T-App
6.	$ffx : \beta$	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.	$x : \alpha$	Bound
2.	$y : \beta$	Bound
3.	$x : \alpha$	T-Var
4.	$\lambda y : \beta. x : \beta \rightarrow \alpha$	3 T-Abst
5.	$\lambda x : \alpha. \lambda y : \beta. x : \alpha \rightarrow \beta \rightarrow \alpha$	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 $y : \alpha \rightarrow \beta$ for some output type β . 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.	$x : \alpha \rightarrow \beta \rightarrow \delta$	Bound
2.	$y : \alpha \rightarrow \beta$	Bound
3.	$z : \alpha$	Bound
4.	$y : \alpha \rightarrow \beta$	T-Var
5.	$z : \alpha$	T-Var
6.	$yz : \beta$	4,5 T-App
7.	$x : \alpha \rightarrow \beta \rightarrow \delta$	T-Var
8.	$xz : \beta \rightarrow \delta$	7,5 T-App
9.	$xz(yz) : \delta$	8,6 T-App
10.	$\lambda z. \alpha. xz(yz) : \alpha \rightarrow \delta$	9 T-Abstr
11.	$\lambda y : \alpha \rightarrow \beta. \lambda z. \alpha. xz(yz) : (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \delta$	10 T-Abstr

12.

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

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 $y : \alpha \rightarrow \beta$ to satisfy the application rule. Because the application yielded a type of β , by the application rule $x : \beta \rightarrow \delta$ for some type δ .

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

Complete type derivation shown below

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

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

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 β . y takes $xz : \beta$ and $z : \alpha$, therefore it is of type $y : \beta \rightarrow \alpha \rightarrow \delta$ for some δ .

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

. Complete type derivation shown below

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

$$\begin{aligned} & \lambda xy.x(\lambda z.y)y \\ & \lambda xy.x(\lambda z.x)y. \end{aligned}$$

Solution. The first term is trivially typable.

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

11.

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

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, $\Gamma' \vdash \lambda z : \alpha.x : \alpha \rightarrow \tau_x$. 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.

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

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. $x : (\alpha \rightarrow \beta) \rightarrow \alpha$ | Bound |
| 2. $\left \begin{array}{l} z : \alpha \\ \hline y : \beta \end{array} \right.$ | Bound |
| 3. $\left \begin{array}{l} z : \alpha \\ \hline y : \beta \end{array} \right.$ | $\dashv \Gamma$ |
| 4. $\left \begin{array}{l} z : \alpha \\ \hline (\lambda z : \alpha.y) : \alpha \rightarrow \beta \end{array} \right.$ | 3 T-Abst |
| 5. $\left \begin{array}{l} z : \alpha \\ \hline x : (\alpha \rightarrow \beta) \rightarrow \alpha \end{array} \right.$ | T-Var |

$$6. \quad \boxed{x(\lambda z : \alpha.y) : \beta} \quad \text{5,4 T-App}$$

7.

$$\begin{aligned} \lambda x : ((\alpha \rightarrow \beta) \rightarrow \beta).x(\lambda z : \alpha.y) \\ : (\alpha \rightarrow \beta) \rightarrow \beta \rightarrow \alpha \end{aligned} \quad \text{6 T-Abst}$$

Problem

Derive

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

Using the rules

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

Solution.

Proof.

1.	$f : A \rightarrow B \wedge g : B \rightarrow C$	Assumption
2.	$f : A \rightarrow B$	$1 \wedge E$
3.	$g : B \rightarrow C$	$1 \wedge E$
4.	$a \in A$	
5.	$f(a) \in B$	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.	$g \circ f : A \rightarrow C$	8 F-Abst
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

$$\frac{\frac{A \Rightarrow B \quad A \quad (\Rightarrow E)}{B} \quad \frac{1. \quad A \quad \text{Premise}}{2. \quad \frac{...}{3. \quad \frac{B}{A \Rightarrow B}}}{(\Rightarrow I)}}$$

Solution.

Proof.

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

■

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 β and δ .

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

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

