Excercises

Chapter 2

Sean Li [1]

1. Reducted

Definition Some rules for reference.

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

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

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

Problem

(2.1) Type the following terms

$$x\ x\ y \quad x\ y\ y \quad x\ y\ x \quad x\ (x\ y) \quad x\ (y\ x)$$

Solution. The first term cannot be typed.

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

The third term is not typable.

Proof. Assume $x\,y\,x = (x\,y)\,x$ is typable. Therefore, $x : \tau$ where $\tau \equiv \sigma \to \tau \to \alpha$ and $y : \sigma$. One can construct an infinite chain of function type by substituting τ: $\tau \equiv \sigma \to (\sigma \to (\sigma \to \ldots \to \alpha) \to \alpha) \to \alpha$. 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. $x\,y : \tau$ **1,2 T-App**
4. $x\,(x\,y) : \tau$ **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. $y\,x : \tau$ **2,1 T-App**
4. $x\,(y\,x) : \sigma$ **1,3 T-App**

Problem

(2.2) 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

$$\text{one} \coloneqq \lambda f\ x\ .\ f\ x$$

Let f be an arbitrary function type that consumes x

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

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

Same type signatures can be given to two

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

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

Problem

(2.3) Find types for

$$K := \lambda x\, y\, .\, x$$
$$S := \lambda x\, y\, z\, .\, x\, z\, (y\, z)$$

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.	$\quad y : \beta$	**Bound**
3.	$\qquad x : \alpha$	**T-Var**
4.	$\quad \lambda y : \beta .\, x : \beta \to \alpha$	**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 $(y\, z)$, it must be of type $\alpha \to \beta \to \delta$.

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

Complete type derivation shown as below:

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

12.

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

Problem

(2.4) Type the bound variables

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

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.

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

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.	$y\, z : \beta$	**4,5 T-App**
7.	$x : \beta \to \delta$	**T-Var**
8.	$x\, (y\, z) : \delta$	**7,6 T-App**
9.	$\lambda z : \alpha .\, x\, (y\, z) : \alpha \to \delta$	**8 T-Abst**
10.	$\lambda y : \alpha \to \beta . \lambda z : \alpha .\, x\, (y\, z) : (\alpha \to \beta) \to \alpha \to \delta$	**9 T-Abst**
11.	$\lambda x : \beta \to \delta . \lambda y : \alpha \to \beta . \lambda z : \alpha .\, x\, (y\, z)$ $: (\beta \to \delta) \to (\alpha \to \beta) \to \alpha \to \delta$	**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 $x\, z : \beta$ and $z : \alpha$, therefore it is of type $y : \beta \to \alpha \to \delta$ for some δ.

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

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

Problem

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

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

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, $\tau_x \equiv (\alpha \to \tau_x) \to \tau$. 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

(2.6) 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

(2.7 a) Derive

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

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.	$f : A \to B \land g : B \to C$	**Assumption**
2.	$f : A \to B$	$1 \land E$
3.	$g : B \to C$	$1 \land E$
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

(2.7 b) Give a derivation in natural deduction of the following:

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

Using the rules

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

1.	A	**Premise**
2.	$\quad \ldots$	
3.	$\quad B$	
	$A \Rightarrow B$	$(\Rightarrow I)$

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

(2.7 c) Prove the following pre-typed term is legal using flag notation

$$\lambda z : \alpha .\, y\ (x\ z)$$

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

∎

Problem

(2.7 d) 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

(2.8 a) Pre-type the bounding variables for the following term

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

Solution.

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

Problem

(2.8 b) Give a derivation in tree format

Solution.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\text{(i)}\ \overline{\ x : (\gamma \to \beta)\ } \qquad \text{(ii)}\ \overline{\ y : (\gamma \to \beta) \to \beta\ }}{x : (\gamma \to \beta), y : ((\gamma \to \beta) \to \beta), z : \gamma \vdash y\, x : \beta}\,\text{(iii) T-App}
    }{x : (\gamma \to \beta), y : ((\gamma \to \beta) \to \beta) \vdash \lambda z : \gamma .\, y\, x : \gamma \to \beta}\,\text{(iv) T-Abst}
    \qquad \text{(v)}\ \overline{\ y : ((\gamma \to \beta) \to \beta)\ }
  }{x : (\gamma \to \beta), y : ((\gamma \to \beta) \to \beta) \vdash y\, (\lambda z\, .\gamma .\, y\, x) : \beta}\,\text{(vi) T-App}
}{
  \dfrac{x : (\gamma \to \beta) \vdash \lambda y : ((\gamma \to \beta) \to \beta).\, y\, (\lambda z : \gamma .\, y\, x) : ((\gamma \to \beta) \to \beta) \to \beta}{(\lambda x : (\gamma \to \beta).\, y : ((\gamma \to \beta) \to \beta).\, y\, (\lambda z : \gamma .\, y\, x)) : (\gamma \to \beta) \to ((\gamma \to \beta) \to \beta) \to \beta}\,\text{(viii) T-Abst}
}\,\text{(vii) T-Abst}
$$

Problem

(2.8 c) Sketch a diagram of tree structure of derivation

Solution. Trivial. See tree above

Problem

(2.8 d) Transform the derivation into flag notation

Solution.

	1.	$x : \gamma \to \beta$	**Bound**
	2.	$\quad y : (\gamma \to \beta) \to \beta$	**Bound**
	3.	$\quad\quad z : \gamma$	**Bound**
(ii)	4.	$\quad\quad\quad y : (\gamma \to \beta) \to \beta$	**T-Var**
(i)	5.	$\quad\quad\quad x : \gamma \to \beta$	**T-Var**
(iii)	6.	$\quad\quad\quad y\, x : \beta$	**4,5 T-App**
(iv)	7.	$\quad\quad \lambda z : \gamma .\, y\, x : \gamma \to \beta$	**6 T-Abst**
(v)	8.	$\quad\quad y : (\gamma \to \beta) \to \beta$	**T-Var**
(vi)	9.	$\quad\quad y\, (\lambda z : \gamma .\, y\, x) : \beta$	**8,7 T-App**
(vii)	10.	$\quad \lambda y : (\gamma \to \beta) \to \beta .\, y\, (\lambda z : \gamma .\, y\, x) : ((\gamma \to \beta) \to \beta) \to \beta$	**9 T-Abst**
(viii)	11.	$\lambda x : (\gamma \to \beta).\lambda y : (\gamma \to \beta) \to \beta .\, y\, (\lambda z : \gamma .\, y\, x) : (\gamma \to \beta) \to ((\gamma \to \beta) \to \beta) \to \beta$	**10 T-Abst**

Problem

(2.9 a) Give derivations of the following judgement

$$x : \delta \to \delta \to \alpha, y : \gamma \to \alpha, z : \alpha \to \beta \vdash$$
$$\lambda u : \delta.\lambda v : \gamma.\, z\,(y\,v) : \delta \to \gamma \to \beta$$

Solution.

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

Problem

(2.9 b) Give derivations of the following judgement

$$x : \delta \to \delta \to \alpha, y : \gamma \to \alpha, z : \alpha \to \beta \vdash$$
$$\lambda u : \delta.\lambda v : \gamma.\, z\,(x\,u\,u) : \delta \to \gamma \to \beta$$

Solution.

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

Problem

(2.10 a) Give derivation for

$$x\ z\ (y\ z)$$

Solution. Assume an context

$$\Gamma \vdash x : \alpha \to \beta \to \gamma$$
$$\Gamma \vdash y : \alpha \to \beta$$
$$\Gamma \vdash z : \alpha$$

1.	$x : \alpha \to \beta \to \gamma$	**T-Var**
2.	$y : \alpha \to \beta$	**T-Var**
3.	$z : \alpha$	**T-Var**
4.	$x\ z : \beta \to \gamma$	**1,3 T-App**
5.	$y\ z : \beta$	**2,3 T-App**
6.	$x\ z\ (y\ z) : \gamma$	**4,5 T-App**

Problem

(2.10 b) Give derivation for

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

Solution. Assume an context

$$\Gamma \vdash y : \gamma \to (\alpha \to \beta)$$
$$\Gamma \vdash z : \gamma$$

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

Problem

(2.10 c) Give derivation for

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

Solution. Assume a context

$$\Gamma \vdash x : \alpha \to \alpha \to \beta$$

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

Problem

(2.10 d) Give derivation for

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

Solution. Consider a context

$$\Gamma \vdash z : \alpha$$
$$\Gamma \vdash y : \beta \to \alpha \to \gamma$$

1.	$x : \alpha \to \beta$	**Bound**
2.	$x : \alpha \to \beta$	**T-Var**
3.	$z : \alpha$	**T-Var**
4.	$x\,z : \beta$	**2,3 T-App**
5.	$y : \beta \to \alpha \to \gamma$	**T-Var**
6.	$y\,(x\,z) : \alpha \to \gamma$	**5,4 T-App**
7.	$y\,(x\,z)\,z : \gamma$	**3,5 T-App**

8. $\lambda x : \alpha \to \beta.\, y\, (x\, z)\, z : (\alpha \to \beta) \to \gamma$ **7 T-Abst**

Problem

(2.11 a) Find an inhabitant of type and prove through derivation

$$(\alpha \to \alpha \to \gamma) \to \alpha \to \beta \to \gamma$$

Solution.

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

Proof.

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

$\lambda x : \alpha \to \alpha \to \gamma.\lambda y : \alpha.\lambda z : \beta.\, x\, y\, y$
$: (\alpha \to \alpha \to \gamma) \to \alpha \to \beta \to \gamma$ **8 T-Abst**

■

Problem

(2.11 b) Find an inhabitant of type and prove through derivation

$$((\alpha \to \gamma) \to \alpha) \to (\alpha \to \gamma) \to \beta \to \gamma$$

Solution.

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

Proof.

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

$\lambda x : (\alpha \to \gamma) \to \alpha.\lambda y : \alpha \to \gamma.\lambda z : \beta.\, x\ (x\ y)$
$: ((\alpha \to \gamma) \to \alpha) \to (\alpha \to \gamma) \to \beta \to \gamma$ **7 T-Abst**

■

Problem

(2.12 a) Construct a term of type

$$((\alpha \to \beta) \to \alpha) \to (\alpha \to \alpha \to \beta) \to \alpha$$

Solution.

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

Proof.

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

12.

$\lambda x : (\alpha \to \beta) \to \alpha . \lambda y : \alpha \to \alpha \to \beta .\ x\ (\lambda z : \alpha .\ y\ z\ z)$
$\quad : ((\alpha \to \beta) \to \alpha) \to (\alpha \to \alpha \to \beta) \to \alpha$ **3,9 T-App**

∎

Problem

(2.12 b) Construct a term of type

$$((\alpha \to \beta) \to \alpha) \to (\alpha \to \alpha \to \beta) \to \beta$$

Solution.

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

Proof.

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

∎

Problem

(2.13 a) Find a term of type

$$(\alpha \to \beta) \to \alpha \to \gamma$$

in context Γ

$$x : \alpha \to \beta \to \gamma \in \Gamma$$

Solution.

$$\lambda u : \alpha \to \beta . \lambda v : \alpha .\, x\, v\, (u\, v)$$

Proof.

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

■

Problem

(2.13 b) Find a term of type

$$\alpha \to (\alpha \to \beta) \to \gamma$$

in context Γ

$$x : \alpha \to \beta \to \alpha \to \gamma \in \Gamma$$

Solution.

$$\lambda u : \alpha . \lambda v : \alpha \to \beta .\, x\, u\, (v\, u)\, u$$

Proof.

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

■

Problem

(2.13 c) Find a term of type

$$(\alpha \to \gamma) \to (\beta \to \alpha) \to \gamma$$

in context Γ

$$x : (\beta \to \gamma) \to \gamma \in \Gamma$$

Solution.

$$\lambda u : \alpha \to \gamma . \lambda v : \beta \to \alpha.\, x\, (\lambda y : \beta.\, u\, (v\, y))$$

Proof.

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

11. $x\,(\lambda y : \beta.\, u\,(v\,y)) : \gamma$ **10,3 T-App**

12. $\lambda v : \beta \to \alpha.\, x\,(\lambda y : \beta.\, u\,(v\,y)) : (\beta \to \alpha) \to \gamma$ **11 T-Abst**

13.

$$\lambda u : \alpha \to \gamma.\lambda v : \beta \to \alpha.\, x\,(\lambda y : \beta.\, u\,(v\,y))$$
$$: (\alpha \to \gamma) \to (\beta \to \alpha) \to \gamma$$ **12 T-Abst**

■

Problem

(2.14) Find an inhabitant of type $\alpha \to \beta \to \gamma$ in the context Γ

$$x : (\gamma \to \beta) \to \alpha \to \gamma \in \Gamma$$

Solution.

$$\lambda u : \alpha.\lambda v : \beta.\, x\,(\lambda z : \gamma.\, v)\,u$$

Proof.

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

■

Problem

(2.15) Prove the thinning lemma via induction.

Solution.

Lemma 1. **Thinning Lemma**. Given two contexts $\Gamma' \subseteq \Gamma$,

$$\Gamma' \vdash M : \sigma \Rightarrow \Gamma \vdash M : \sigma$$

We can do structural induction over the structure of M by the finite syntatic constructors of λ terms.

Case 1 : Variable. By the generation lemma $M : \sigma \in \Gamma'$. By the definition of subcontextes, $M : \sigma \in \Gamma$. It follows from the T-Var rule that $\Gamma \vdash M : \sigma$. ∎

Case 2 : Application. Therefore the judgement must be of form

$$\Gamma' \vdash A\ B : \sigma$$

Where $M = A\ B$. By the generation lemma, there exists a type τ such that $\Gamma' \vdash A : \tau \to \sigma$ and $\Gamma' \vdash B : \tau$. By the principle of induction the thinning lemma holds for the terms A and B. By plugging in $\Gamma \vdash A : \tau \to \sigma, B : \tau$, the T-App rule proves $\Gamma \vdash M : \sigma$. ∎

Case 3 : Abstraction. Therefore M is of a function type. We denote M as a abstraction term

$$\lambda x : \alpha.\ N : \alpha \to \beta$$

where $x \notin \mathrm{FR}\ N$. By the generation lemma, $\Gamma', x : \alpha \vdash N : \beta$.

By the principle of induction, the thinning lemma already holds for $N : \beta$. Because $\Gamma', x : \alpha \subseteq \Gamma, x : \alpha$, the thinning lemma proves $\Gamma, x : \alpha \vdash N : \beta$. By the T-Abst rule, $\Gamma \vdash \lambda x : \alpha.\ N : \alpha \to \beta$. ∎

Problem

(2.16) Prove the subterm lemma.

Solution.

Lemma 2. **Subterm Lemma**. If $M \in \Lambda_{\mathbb{T}}$ is legal, then all subterms of M are.

Let's formalize the lemma. The definition of a legal term is a term M with the existence of a context Γ and a type τ such that $\Gamma \vdash M : \tau$, in other words, M is typable. Because M is typable iff there's an appliable inference rule and each term appearing in the premise of each inference rule is an immediate subterm of M that could construct M, structural induction could be done down the typing tree as it is also an induction down the term's structure, inducting over each subterm.

Case 1 : Variable. M is the only subterm of M thus is trivially typable and legal. ■

Case 2 : Application. Therefore the derivation must be of form

$$\frac{\Gamma \vdash A : \tau \to \sigma \qquad \Gamma \vdash B : \tau}{\Gamma \vdash A\ B : \sigma}\text{T-App}$$

Where $M = A\ B$. From the generation lemma, those A and B must exist typable, thus legal. By the principle of induction, the subterm lemma holds for A and B. Denote the lemma as $P\ (\Lambda_{\mathbb{T}})$:

$$\begin{aligned} & P\ (A\ B) \land P\ (A) \land P\ (B) \land (\forall\ a \in \text{Sub}\ A, P\ (a)) \land (\forall\ b \in \text{Sub}\ B, P\ (b)) \\ = & \forall\ m \in \{A\ B, A, B\} \cup \text{Sub}\ A \cup \text{Sub}\ B, P\ (m) \\ = & \forall\ m \in \{A\ B\} \cup \text{Sub}\ A \cup \text{Sub}\ B, P\ (m) \\ = & \forall\ m \in \text{Sub}\ A\ B, P\ (m) = P\ (A\ B) = P\ (M) \end{aligned}$$

■

Case 3 : Abstraction. Therefore the derivation must be of form

$$\frac{\Gamma, x : \alpha \vdash N : \beta}{\Gamma \vdash \lambda x : \alpha.\ N : \alpha \to \beta}\text{T-Abst}$$

Where $M = \lambda x : \alpha.\ N$ and $x \notin \text{FR}\ N$. By the generation lemma, N is typable under the context $\Gamma, x : \alpha$, thus valid. By the inductive hypothese, the subterm lemma holds for N. Denote the lemma as $P\ (\Lambda_{\mathbb{T}})$

$$\begin{aligned} & P\ (\lambda x : \alpha.\ N) \land (P\ (N) \land \forall\ n \in \text{Sub}\ N, P\ (n)) \\ = & \forall\ m \in \{\lambda x : \alpha.\ N, N\} \cup \text{Sub}\ N, P\ (m) \\ = & \forall\ m \in \{\lambda x : \alpha.\ N\} \cup \text{Sub}\ N, P\ (m) = P\ (\lambda x : \alpha.\ N) = P\ (M) \end{aligned}$$

■

Problem

(2.17) Prove the uniqueness of types lemma.

Solution.

Lemma 3. Assume $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M : \tau$. Then $\sigma = \tau$.

Again do induction on the construction of M.

Case 1 : Variable. By the generation lemma, $M : \sigma \in \Gamma$ and $M : \tau \in \Gamma$ By the definition of contexts. Because the context could only provide one judgement per variable, M must have a unique type. ■

Case 2 : Application. M must be of form $A\ B$. By the generation lemma, two judgements could be obtained where $\alpha, \beta \in \mathbb{T}$:

$$\Gamma \vdash A : \alpha \to \sigma, B : \alpha$$

$$\Gamma \vdash A : \beta \to \tau, B : \beta$$

By the inductive hypothesis, the uniqueness lemma already holds for A and B. Therefore $\alpha \equiv \beta$. Because A is a function type, it could be expanded into two forms:

$$\Gamma \vdash (\lambda x : \alpha.\ N) : \alpha \to \sigma$$

$$\Gamma \vdash (\lambda x : \beta.\ N) : \beta \to \tau$$

By the generation lemma and the equivalence $\alpha \equiv \beta$, the above further deduce to

$$\Gamma, x : \alpha \vdash N : \sigma$$

$$\Gamma, x : \alpha \vdash N : \tau$$

By the principle of induction, the uniqueness lemma holds for N, therefore $\sigma \equiv \tau$. By substituting τ for σ, the final type for M should be the same in the conclusion for both judgements. ■

Case 3 : Abstraction. Because M is a function type, two judgements could be obtained for some $\alpha, \beta \in \mathbb{T}$

$$\Gamma \vdash \lambda x : \gamma.\ N : \sigma \text{ where } \sigma \equiv \gamma \to \alpha$$

$$\Gamma \vdash \lambda x : \gamma.\ N : \tau \text{ where } \tau \equiv \gamma \to \beta$$

By applying the generation lemma

$$\Gamma, x : \gamma \vdash N : \alpha$$

$$\Gamma, x : \gamma \vdash N : \beta$$

By the principle of induction, N already conform to the lemma, thus $\alpha \equiv \beta$. Therefore, $\gamma \to \alpha \equiv \gamma \to \beta$. ■

Problem

(2.18) Prove the *Subject Reduction* lemma.

Lemma 4. If $\Gamma \vdash L : \rho$ and $L \twoheadrightarrow_{\beta} L'$, then $\Gamma \vdash L' : \rho$

Solution. Proof by generation of L': which is by establishing all possibilities of how L' was derived.

Basis : Direct Reduction.

$$L \equiv (\lambda x : \sigma.\ M)\ N \text{ and } L' \equiv M\ [x := N]$$

By the generation lemma and some destructuring we have $M : \gamma$ under $\Gamma, x : \sigma$ and $N : \sigma$ under Γ. By applying the T-App rule, $L : \gamma$. By the uniqueness of types lemma, $\gamma \equiv \rho$. Now we define $\Gamma' \equiv \emptyset$ and $\Gamma'' \equiv \Gamma$, by the substitution lemma we have $\Gamma \vdash M\ [x := N] : \gamma$. We substitue $\gamma \equiv \rho$ back and get $\Gamma \vdash M\ [x := N] : \rho$. ■

Case 1 : Left Term Reduction.

$$L \equiv M\ N \text{ and } L' \equiv M'\ N \text{ where } M \to_{\beta} M'$$

We type the terms $M : \sigma \to \rho$ and $N : \sigma$. By the inductive hypothesis, $\Gamma \vdash M' : \sigma \to \rho$. By the T-App rule, $\Gamma \vdash M'\ N : \rho$. ■

Case 2 : Left Term Reduction.

$$L \equiv M\ N \text{ and } L' \equiv M\ N' \text{ where } N \to_{\beta} N'$$

We type the terms $M : \sigma \to \rho$ and $N : \sigma$. By the inductive hypothesis, $\Gamma \vdash N' : \sigma$. By the T-App rule, $\Gamma \vdash M\ N' : \rho$. ■

Case 3 : Reduction In Body.

$$L \equiv (\lambda x : \tau.\ M) \text{ and } L' \equiv (\lambda x : \tau.\ M') \text{ where } M \to_{\beta} M'$$

Let $\sigma \in \mathbb{T}$ where $\rho \equiv \tau \to \sigma$. By the generation lemma $\Gamma, x : \tau \vdash M : \sigma$. By the inductive hypothesis, $\Gamma, x : \tau \vdash M' : \sigma$, and by the T-Abst rule, $\Gamma \vdash (\lambda x : \tau.\ M') : \tau \to \sigma$. Therefore, $\Gamma \vdash L' : \rho$ by substituting $\tau \to \sigma$ for ρ. ■

—

Completed Dec 15 2:24 am.