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

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

(2.2) Find types for zero, one, and two

Solution. Term for zero is

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

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. $\quad | \; x : \beta$ **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.	$\lambda f : \alpha \to \beta.x : \alpha.f f x : (\alpha \to \beta) \to \alpha \to \beta$	**7 T-Abst**

Problem

(2.3) 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

(2.4) 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

(2.5) 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

(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.

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

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

$$\frac{A \Rightarrow B \qquad A}{B}\,(\Rightarrow E) \qquad \frac{\begin{array}{lll} 1. & A & \textbf{Premise} \\ 2. & \quad \vert \ \dots & \\ 3. & \quad \vert \ B & \end{array}}{A \Rightarrow B}\,(\Rightarrow I)$$

Solution.

Proof.

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

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

Solution.

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

1.	$z : \alpha$	**Bound**
2.	$\vert\ x : \alpha \to \beta$	$\dashv \Gamma$
3.	$\vert\ z : \alpha$	**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

(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

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

Solution.

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

Problem

(2.8 b) Give a derivation in tree format

Solution.

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

Problem

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

Solution. Trivial.

Problem

(2.8 d) Transform the derivation into flag notation

Solution.

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

$\lambda x : (\gamma \to \beta).\lambda y : (\gamma \to \beta) \to \beta.y(\lambda z : \gamma.yx)$
$: (\gamma \to \beta) \to ((\gamma \to \beta) \to \beta) \to \beta$ **10 T-Abst**

Problem

(2.9 a) Give derivations of the following judgement

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

Solution.

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

Problem

(2.9 b) Give derivations of the following judgement

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

Solution.

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

Problem

(2.10 a) Give derivation for

$$xz(yz)$$

Solution. Assume an context

$$\begin{aligned}&\Gamma \vdash x : \alpha \to \beta \to \gamma\\&\Gamma \vdash y : \alpha \to \beta\\&\Gamma \vdash z : \alpha\end{aligned}$$

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

Problem

(2.10 b) Give derivation for

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

Solution. Assume an context

$$\begin{aligned}&\Gamma \vdash y : \gamma \to (\alpha \to \beta)\\&\Gamma \vdash z : \gamma\end{aligned}$$

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

Problem

(2.10 c) Give derivation for

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

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.	$xy : \alpha \to \beta$	**4,5 T-App**
7.	$xyy : \beta$	**6,5 T-App**
8.	$z(xyy) : \gamma$	**3,6 T-App**
9.	$\lambda z : \beta \to \gamma.z(xyy) : (\beta \to \gamma) \to \gamma$	**8 T-Abst**
10.	$\lambda y : \alpha.\lambda z : \beta \to \gamma.z(xyy) : \alpha \to (\beta \to \gamma) \to \gamma$	**8 T-Abst**

Problem

(2.10 d) Give derivation for

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

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

8. $\lambda x : \alpha \to \beta . y(xz)z : (\alpha \to \beta) \to \gamma$ **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.

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

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.	$xy : \alpha \to \gamma$	**4,5 T-App**
7.	$xyy : \gamma$	**6,5 T-App**
8.	$\lambda z : \beta . xyy : \beta \to \gamma$	**7 T-Abst**
9.	$\lambda y : \alpha . \lambda z : \beta . xyy : \alpha \to \beta \to \gamma$	**8 T-Abst**
10.	$\lambda x : \alpha \to \alpha \to \gamma . \lambda y : \alpha . \lambda z : \beta . xyy : (\alpha \to \alpha \to \gamma) \to \alpha \to \beta \to \gamma$	**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.

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

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

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

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

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

12.

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

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

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

$$\lambda u : \alpha \to \beta . \lambda v : \alpha . xv(uv)$$

Proof.

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

$$\lambda u : \alpha . \lambda v : \alpha \to \beta . xu(vu)u$$

Proof.

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

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

Proof.

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

11.	$x(\lambda y : \beta . u(vy)) : \gamma$	**10,3 T-App**
12.	$\lambda v : \beta \to \alpha . x(\lambda y : \beta . u(vy)) : (\beta \to \alpha) \to \gamma$	**11 T-Abst**
13.		

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

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 $\Gamma' \subseteq \Gamma$,

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

The antecedent is a judgement. Therefore, we can do structural induction over the inference rules that construct the judgement. By the nature of the inference rules, we can deduce the premise from the conclusion via structural decomposition and apply the inductive hypothesis to its immediate premise.

Case 1 : T-Var. Therefore the derivation must be of form

$$\frac{M : \sigma \in \Gamma'}{\Gamma' \vdash M : \sigma}\text{T-Var}$$

Because this rule is invertable therefore $M : \sigma \in \Gamma'$. By the definition of subcontextes, $M : \sigma \in \Gamma$. It follows from the T-Var rule that $\Gamma \vdash M : \sigma$. ■

Case 2 : T-App. Therefore the derivation must be of form

$$\frac{\Gamma' \vdash A : \tau \to \sigma \qquad \Gamma' \vdash B : \tau}{\Gamma' \vdash AB : \sigma}\text{T-App}$$

Where $M = AB$. 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 : T-Abst. Therefore M is a function type. We denote M as a abstraction term $\lambda x : \alpha.N : \alpha \to \beta$ where $x \notin \text{FR } N$. Therefore, type derivation for M must be of form

$$\frac{\Gamma', x : \alpha \vdash N : \beta}{\Gamma' \vdash \lambda x : \alpha.N : \alpha \to \beta}\text{T-Abst}$$

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

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

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

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

■

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 $x \notin \text{FR } N$. Therefore, 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 $P(\Lambda_{\mathbb{T}})$

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

■