

EXERCISES

CHAPTER 2

SEAN LI ¹

1. Redacted

Definition Some rules for reference.

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

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

In this document, convention is that all type judgements in a proof tree, unless stated otherwise, is derived from a single 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 $\tau \rightarrow \sigma$. 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 $x : \tau \rightarrow \tau \rightarrow \sigma$ and $y : \tau$.

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

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

3. $x y : \tau \rightarrow \sigma$ **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 $\tau \equiv \sigma \rightarrow \tau \rightarrow \alpha$ and $y : \sigma$. One can construct an infinite chain of function type by substituting τ : $\tau \equiv \sigma \rightarrow (\sigma \rightarrow (\sigma \rightarrow \dots \rightarrow \alpha) \rightarrow \alpha) \rightarrow \alpha$. 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. $x y : \tau$ **1,2 T-App**
4. $x (x y) : \tau$ **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. $y x : \tau$ **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. $\left| \begin{array}{l} x : \beta \end{array} \right.$ **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. $\lambda f : \alpha \rightarrow \beta . \lambda x : \alpha . f f x : (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ **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 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 $(y z)$, it must be of type $\alpha \rightarrow \beta \rightarrow \delta$.

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

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

12.

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

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 $y : \alpha \rightarrow \beta$ 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 .

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

Complete type derivation shown below

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

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

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

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

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

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

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

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.	$x : (\delta \rightarrow \alpha) \rightarrow \alpha \rightarrow \beta$	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.

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

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

(2.6) 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.

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

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.	$z : \alpha$	Bound
3.	$y : \beta$	$\vdash \Gamma$
4.	$(\lambda z : \alpha. y) : \alpha \rightarrow \beta$	3 T-Abst
5.	$x : (\alpha \rightarrow \beta) \rightarrow \alpha$	T-Var
6.	$x (\lambda z : \alpha. y) : \beta$	5,4 T-App

7.

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

Problem

(2.7 a) 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

(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

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

Solution.

Proof.

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

■

Problem

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

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

Solution.

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

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

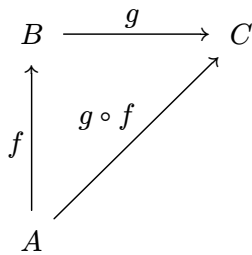
- | | | |
|----|---|------------------|
| 4. | $x z : \beta$ | 2,3 T-App |
| 5. | $y : \beta \rightarrow \delta$ | $\neg \Gamma$ |
| 6. | $y (x z) : \delta$ | 5,4 T-App |
| 7. | $\lambda z : \alpha. y (x z) : \alpha \rightarrow \delta$ | 6 T-Abst |

■

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:



Problem

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

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

Solution.

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

Problem

(2.8 b) Give a derivation in tree format

Solution.

$$\begin{array}{c}
\text{(i)} \frac{}{x : (\gamma \rightarrow \beta)} \quad \text{(ii)} \frac{}{y : (\gamma \rightarrow \beta) \rightarrow \beta} \\
\text{(iii)} \frac{}{x : (\gamma \rightarrow \beta), y : ((\gamma \rightarrow \beta) \rightarrow \beta), z : \gamma \vdash y x : \beta} \text{T-App} \\
\text{(v)} \frac{}{y : ((\gamma \rightarrow \beta) \rightarrow \beta)} \quad \text{(iv)} \frac{}{x : (\gamma \rightarrow \beta), y : ((\gamma \rightarrow \beta) \rightarrow \beta) \vdash \lambda z : \gamma. y x : \gamma \rightarrow \beta} \text{T-Abst} \\
\text{(vi)} \frac{}{x : (\gamma \rightarrow \beta), y : ((\gamma \rightarrow \beta) \rightarrow \beta) \vdash y (\lambda z : \gamma. y x) : \beta} \text{T-App} \\
\text{(vii)} \frac{}{x : (\gamma \rightarrow \beta) \vdash \lambda y : ((\gamma \rightarrow \beta) \rightarrow \beta). y (\lambda z : \gamma. y x) : ((\gamma \rightarrow \beta) \rightarrow \beta) \rightarrow \beta} \text{T-Abst} \\
\text{(viii)} \frac{}{(\lambda x : (\gamma \rightarrow \beta). y : ((\gamma \rightarrow \beta) \rightarrow \beta). y (\lambda z : \gamma. y x)) : (\gamma \rightarrow \beta) \rightarrow ((\gamma \rightarrow \beta) \rightarrow \beta) \rightarrow \beta} \text{T-Abst}
\end{array}$$

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

Problem

(2.9 a) Give derivations of the following judgement

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

Solution.

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

Problem

(2.9 b) Give derivations of the following judgement

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

Solution.

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

Problem

(2.10 a) Give derivation for

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

Solution. Assume an context

$$\Gamma \vdash x : \alpha \rightarrow \beta \rightarrow \gamma$$

$$\Gamma \vdash y : \alpha \rightarrow \beta$$

$$\Gamma \vdash z : \alpha$$

1. $x : \alpha \rightarrow \beta \rightarrow \gamma$ **T-Var**
2. $y : \alpha \rightarrow \beta$ **T-Var**
3. $z : \alpha$ **T-Var**
4. $x\ z : \beta \rightarrow \gamma$ **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

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

Solution. Assume an context

$$\Gamma \vdash y : \gamma \rightarrow (\alpha \rightarrow \beta)$$

$$\Gamma \vdash z : \gamma$$

- | | | |
|----|--|------------------|
| 1. | $x : (\alpha \rightarrow \beta) \rightarrow \beta$ | Bound |
| 2. | $x : (\alpha \rightarrow \beta) \rightarrow \beta$ | T-Var |
| 3. | $y : \gamma \rightarrow \alpha \rightarrow \beta$ | T-Var |
| 4. | $z : \gamma$ | T-Var |
| 5. | $y\ z : \alpha \rightarrow \beta$ | 3,4 T-App |
| 6. | <u>$x\ (y\ z) : \beta$</u> | 2,5 T-App |
| 7. | $\lambda x : (\alpha \rightarrow \beta) \rightarrow \beta. x\ (y\ z) : ((\alpha \rightarrow \beta) \rightarrow \beta) \rightarrow \beta$ | 6 T-Abst |

Problem

(2.10 c) Give derivation for

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

Solution. Assume a context

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

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

Problem

(2.10 d) Give derivation for

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

Solution. Consider a context

$$\Gamma \vdash z : \alpha$$

$$\Gamma \vdash y : \beta \rightarrow \alpha \rightarrow \gamma$$

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

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

Problem

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

$$(\alpha \rightarrow \alpha \rightarrow \gamma) \rightarrow \alpha \rightarrow \beta \rightarrow \gamma$$

Solution.

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

Proof.

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

$$\lambda x : \alpha \rightarrow \alpha \rightarrow \gamma. \lambda y : \alpha. \lambda z : \beta. x y y$$

$$: (\alpha \rightarrow \alpha \rightarrow \gamma) \rightarrow \alpha \rightarrow \beta \rightarrow \gamma \quad \mathbf{8\ T-Abst}$$

■

Problem

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

$$((\alpha \rightarrow \gamma) \rightarrow \alpha) \rightarrow (\alpha \rightarrow \gamma) \rightarrow \beta \rightarrow \gamma$$

Solution.

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

Proof.

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

■

Problem

(2.12 a) Construct a term of type

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

Solution.

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

Proof.

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

12.

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

3,9 T-App

■

Problem

(2.12 b) Construct a term of type

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

Solution.

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

Proof.

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

■

Problem

(2.13 a) Find a term of type

$$(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma$$

in context Γ

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

Solution.

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

Proof.

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

■

Problem

(2.13 b) Find a term of type

$$\alpha \rightarrow (\alpha \rightarrow \beta) \rightarrow \gamma$$

in context Γ

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

Solution.

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

Proof.

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

■

Problem

(2.13 c) Find a term of type

$$(\alpha \rightarrow \gamma) \rightarrow (\beta \rightarrow \alpha) \rightarrow \gamma$$

in context Γ

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

Solution.

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

Proof.

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

11.	$\frac{x (\lambda y : \beta. u (v y)) : \gamma}{\lambda v : \beta \rightarrow \alpha. x (\lambda y : \beta. u (v y)) : (\beta \rightarrow \alpha) \rightarrow \gamma}$	10,3 T-App
12.		11 T-Abst
13.		
	$\lambda u : \alpha \rightarrow \gamma. \lambda v : \beta \rightarrow \alpha. x (\lambda y : \beta. u (v y))$	
	$: (\alpha \rightarrow \gamma) \rightarrow (\beta \rightarrow \alpha) \rightarrow \gamma$	12 T-Abst

■

Problem

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

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

Solution.

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

Proof.

1.	$u : \alpha$	Bound
2.	$v : \beta$	Bound
3.	$x : (\gamma \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma$	T-Var
4.	$z : \gamma$	Bound
5.	$v : \beta$	T-Var
6.	$\lambda z : \gamma. v : \gamma \rightarrow \beta$	5 T-Abst
7.	$x (\lambda z : \gamma. v) : \alpha \rightarrow \gamma$	3,6 T-App
8.	$u : \alpha$	T-Var
9.	$x (\lambda z : \gamma. v) u : \gamma$	7,8 T-App
10.	$\lambda v : \beta. x (\lambda z : \gamma. v) u : \beta \rightarrow \gamma$	9 T-Abst
11.	$\lambda u : \alpha. \lambda v : \beta. x (\lambda z : \gamma. v) u : \alpha \rightarrow \beta \rightarrow \gamma$	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 syntactic 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 $\Gamma' \vdash A : \tau \rightarrow \sigma$ and $\Gamma' \vdash B : \tau$. By the principle of induction the thinning lemma holds for the terms A and B . By plugging in $\Gamma \vdash A : \tau \rightarrow \sigma, B : \tau$, 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

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

where $x \notin \text{FR } N$. 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, $\Gamma \vdash \lambda x : \alpha. N : \alpha \rightarrow \beta$. ■

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 applicable 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

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

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 $P(\Lambda_{\mathbb{T}})$:

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

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

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

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 hypothesis, the subterm lemma holds for N . Denote the lemma as $P(\Lambda_{\mathbb{T}})$

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

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 $\alpha, \beta \in \mathbb{T}$:

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

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

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:

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

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

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 $\alpha, \beta \in \mathbb{T}$

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

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

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, $\gamma \rightarrow \alpha \equiv \gamma \rightarrow \beta$. ■

Problem

(2.18) Prove the *Subject Reduction* lemma.

If $\Gamma \vdash L : \rho$ and $L \xrightarrow[\beta]{\rightarrow} L'$, 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 substitute $\gamma \equiv \rho$ back and get $\Gamma \vdash M [x := N] : \rho$. ■

Case 1 : Left Term Reduction.

$$L \equiv M N \text{ and } L' \equiv M' N \text{ where } M \xrightarrow[\beta]{\rightarrow} M'$$

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

Case 2 : Left Term Reduction.

$$L \equiv M N \text{ and } L' \equiv M N' \text{ where } N \xrightarrow[\beta]{\rightarrow} N'$$

We type the terms $M : \sigma \rightarrow \rho$ 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.

$$L \equiv (\lambda x : \tau. M) \text{ and } L' \equiv (\lambda x : \tau. M') \text{ where } M \xrightarrow[\beta]{\rightarrow} M'$$

Let $\sigma \in \mathbb{T}$ where $\rho \equiv \tau \rightarrow \sigma$. 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, $\Gamma \vdash (\lambda x : \tau. M') : \tau \rightarrow \sigma$. Therefore, $\Gamma \vdash L' : \rho$ by substituting $\tau \rightarrow \sigma$ for ρ . ■

—

Completed Dec 15 2:24 am.