Church Encoding with Dependent Types

Peng Fu, Aaron Stump

Computer Science, The University of Iowa

Abstract. We introduce **S**, a Curry-style dependent type system featuring new type assignment rules for the *self type* $\iota x.T$, together with restrictive mutually recursive definitions and Miquel's implicit product $\forall x : T.T'$. We show how to obtain Church-encoded datatypes and the corresponding induction principles with **S**. A notion of erasure from **S** to $\mathbf{F}_\omega$ with let-bindings is defined, thus establishing strongly normalization for the terms. Type preservation is proved by combining several confluence techniques with Barendregt's method for proving subject reduction for Curry-style System **F**.

1 Introduction

It is well known that the natural numbers can be encoded as lambda terms using Church encoding [6] or Scott encoding (reported in [8]). Thus operations such as *addition*, *multiplication* can be performed by beta-reduction on lambda terms. Other inductive data structures such as trees and lists ([4], chapter 11 in [10]) can also be represented in a similar fashion.

Church-encoded data can be represented in System **F**[9], but it is rarely adopted in theorem provers based on dependent types. As summarized in Werner's [18], it is ineffecient to define certain operation on Church-encoded data, e.g. the *predecessor* function; the induction principle is not derivable within Calculus of Construction (**CC**); and $0 \neq 1$ can not be proved within **CC**. These problems provide reasons for treating inductive datatypes as primitive [15]. We want to stress that the ineffeciency to retrieve subdata is an inherent problem for Church encoding, $0 \neq 1$ is unprovable in **CC** is because the notion of the contradiction is inadequate, and the underivability of induction principle shows the limitation of **CC**.

In this paper, we explore another point of view, namely, adapt **CC** to fit Church encoding. The results of our work show that this point of view is viable and can potentially simplify the design of dependently typed languages. More specifically, we make the following technical contributions:

- We propose System **S**, which allows us to derive induction principles for Church-encoded data. We also propose to change the notion of contradiction, which enables us to derive $0 \neq 1$ in System **S**.
- We prove strong normalization of **S** by erasing it to $\mathbf{F}_\omega$ with let-bindings.
- We show how to prove type preservation by applying several confluence results and techniques.

1.1 Induction Principle

We shall take a closer look at the difficulties of deriving an induction principle for Church numerals in **CC**, then offer our solutions. In **CC** à la Curry, let $\mathsf{Nat} := \Pi X.(X \to X) \to X \to X$. One can obtain a notion of *indexed iterator* by $\mathsf{It} := \lambda x.\lambda f.\lambda a.xfa$ and $\mathsf{It} : \Pi X.\Pi x : \mathsf{Nat}.(X \to X) \to X \to X$. Thus we have $\mathsf{It}\ \bar{n} =_\beta \lambda f.\lambda a.\bar{n}\ f\ a =_\beta \lambda f.\lambda a.\underbrace{f(f(f...(f}_{n}\ a)...))$. One may want to know if we can obtain a finer version, namely, the induction principle-Ind such that:

$\mathsf{Ind} : \Pi P : \mathsf{Nat} \to *.\Pi x : \mathsf{Nat}.(\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to P\ x$

Let us try to construct such Ind. First observe the following beta-equalities and typings:

$\mathsf{Ind}\ \bar{0} =_\beta \lambda f.\lambda a.a$

$\mathsf{Ind}\ \bar{0} : (\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to P\ \bar{0}$

$\mathsf{Ind}\ \bar{n} =_\beta \lambda f.\lambda a.\underbrace{f\ \overline{n-1}(...f\ \bar{1}\ (f}_{n>0}\ \bar{0}\ a))$

$\mathsf{Ind}\ \bar{n} : (\Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y))) \to P\ \bar{0} \to P\ \bar{n}$

with $f : \Pi y : \mathsf{Nat}.(Py \to P(\mathsf{S}y)), a : P\ \bar{0}$

The equalities above suggest that $\mathsf{Ind} := \lambda x.\lambda f.\lambda a.x\ f\ a$, with a different notion of lambda numerals, i.e.

$\bar{0} := \lambda s.\lambda z.z$

$\bar{n} := \lambda s.\lambda z.s\ \overline{n-1}\ (\overline{n-1}\ s\ z)$

Each numeral corresponds to its terminating recursion. They are not exactly Church numerals because each numeral contains its subdata., similarly to Scott numerals.

Now, let us try to type these lambda numerals. It is reasonable to assign $s : \Pi y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y))$ and $z : P\ \bar{0}$. Thus we have the following typing relations:

$\bar{0} : \Pi y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ \bar{0}$

$\bar{1} : \Pi y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ \bar{1}$

$\bar{n} : \Pi y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ \bar{n}$

So we have the following definition:

$\mathsf{Nat} := \Pi P : \mathsf{Nat} \to *.\Pi y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ \bar{n}$ for any $\bar{n}$.

Two problems arise with this scheme of encoding. The first problem involves mutually recursiveness. The definiens of Nat contains Nat and $\mathsf{S}, \bar{0}$, while the type of S is $\mathsf{Nat} \to \mathsf{Nat}$ and the type of $\bar{0}$ is Nat. So the typing of Nat will be mutually recursive. Observe that the recursive occurrences of Nat are all at the type-annotated positions; i.e., the right side of the ":".

Note that the subdata of $\bar{n}$ is responsible for one recursive occurrence of Nat, namely, $\Pi y : \mathsf{Nat}$. If one never computes with the subdata, then these numerals will behave just like Church numerals. This inspires us to use Miquel's implicit product [14]. So we redefine Nat to be:

$\mathsf{Nat} := \Pi P : \mathsf{Nat} \to *.\forall y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ \bar{n}$ for any $\bar{n}$.

Here $\forall y : \mathsf{Nat}$ is the implicit product. Now our notion of numerals are exactly Church numerals instead of Scott's derivative. Even better, this definition of Nat can be erased meaningfully to $\mathbf{F}_\omega$. $\mathbf{F}_\omega$'s types do not have dependency on terms,

so $P : \mathsf{Nat} \to *$ will get erased to $P : *$ and it is known that one can also erase the implicit product[1]. The erasure of Nat will be $\Pi P : *.(P \to P) \to P \to P$, which is the definition of Nat in $\mathbf{F}_\omega$. As long as we restrict the recursive occurrences of the type to be at the erased positions, we will have a meaningful interpretation over $\mathbf{F}_\omega$.

The second problem is about quantification. We want to define a type Nat for any $\bar{n}$, but right now what we really have is one Nat for each numerals $\bar{n}$. We solve this problem by introducing a new type construct $\iota x.T$ called *self type*. Thus we define

$\mathsf{Nat} := \iota x.\Pi P : \mathsf{Nat} \to *.\forall y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ x$

We require that the self type can only be instantiated/generalized by its own subject, so we add the following two rules:

$$\frac{\Gamma \vdash t : [t/x]T}{\Gamma \vdash t : \iota x.T}\ \mathit{selfGen} \qquad \frac{\Gamma \vdash t : \iota x.T}{\Gamma \vdash t : [t/x]T}\ \mathit{selfInst}$$

We have the following inference[1]:

$$\frac{\bar{n} : \Pi P : \mathsf{Nat} \to *.\forall y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ \bar{n}}{\bar{n} : \iota x.\Pi P : \mathsf{Nat} \to *.\forall y : \mathsf{Nat}.(P\ y \to P(\mathsf{S}\ y)) \to P\ \bar{0} \to P\ x}$$

1.2 The Notion of Contradiction

In **CC** à la Curry, it is customary to use $\Pi X : *.X$ as the notion of contradiction, since an inhabitant of the type $\Pi X : *.X$ will inhabit any type, so the law of explosion is subsumed by the type $\Pi X : *.X$. However, this notion of contradiction is too strong to be useful. Let $t =_A t'$ denote $\Pi C : A \to *.Ct \to Ct'$ with $A : *, t : A, t' : A$. Then $0 =_{\mathsf{Nat}} 1$ can be expanded to $\Pi C : \mathsf{Nat} \to *.C0 \to C1$ (0 is Leibniz equals to 1). One can not derive a proof for $(\Pi C : \mathsf{Nat} \to *.C0 \to C1) \to \Pi X : *.X$, because the erasure of $(\Pi C : \mathsf{Nat} \to *.C0 \to C1) \to \Pi X : *.X$ in System **F** would be $(\Pi C : *.C \to C) \to \Pi X : *.X$, and we know that $\Pi C : *.C \to C$ is inhabited. So the inhabitation of $(\Pi C : \mathsf{Nat} \to *.C0 \to C1) \to \Pi X : *.X$ will imply the inhabitation of $\Pi X : *.X$. If we take Leibniz equality, then using $\Pi X : *.X$ as the notion of contradiction is so strong that we simply can not prove any negative results about equality.

On the other hand, an equational theory is considered inconsistent if $a = b$ for all term a and b. This motivates us to use this as the notion of contradiction. We propose to use $\Pi A : *.\Pi x : A.\Pi y : A.x =_A y$ as the notion of contradiction in **CC**. We first want to make sure it is uninhabited. The way to argue that is first assume it is inhabited by t. Since **CC** is strongly normalizing, the normal form of t must be of the form[2] $[\lambda A : *.]\lambda x[: A].\lambda y[: A].[\lambda C : A \to *].\lambda z[: C\ x].n$ for some normal term n with type $C\ y$, but we know that there is no combination

[1] The double bar means that the converse of the inference rule is also true.

[2] We use square [ ] to mean the type inside is only for typing.

of x, y, z to make a term of type $C\ y$. So the type $\Pi A : *.\Pi x : A.\Pi y : A.\Pi C : A \to *.Cx \to Cy$ is uninhabited. Then, we want to make sure this notion of contradiction is usable, namely, we want to prove the following theorem.

Theorem 1. *$0 = 1 \to \bot$ is inhabited in* **CC**, *where* $\bot := \Pi A : *.\Pi x : A.\Pi y : A.\Pi C : A \to *.Cx \to Cy$, $0 := \lambda s.\lambda z.z$, $1 := \lambda s.\lambda z.s\ z$.

Proof. Assume $\mathsf{Nat} := \Pi B : *.(B \to B) \to B \to B$. Let $\Gamma = a : (\Pi D : \mathsf{Nat} \to *.D0 \to D1), A : *, x : A, y : A, C : A \to *, c : C\ x$. We want to construct a term of type $C\ y$. Let $F := \lambda n[:\mathsf{Nat}].n\ [A]\ (\lambda q[:A].y)x$. Note that $F : \mathsf{Nat} \to A$. We know that $F0 =_\beta x$ and $F1 =_\beta y$. So we can indeed convert the type of c from Cx to $C\ (F0)$. And then we instantiate the D in $\Pi D : \mathsf{Nat} \to *.D0 \to D1$ with $\lambda x[:\mathsf{Nat}].C\ (Fx)$. So we have $C\ (F0) \to C\ (F1)$ as the type of a. So $ac : C(F1)$, which means $ac : Cy$. So we just show how to inhabit $0 = 1 \to \bot$ in $\mathbf{CC}$[3].

Once $\bot$ is derived, one can not distinguish the domain of individuals. Note that this notion of contradiction does not admit law of explosion.

1.3 Overview

In section 2, we present a novel type System **S**, which not only enables us to type Church-encoded data, but also allows us to derive the corresponding induction principles. Several examples about numerals and vectors are given to demonstrate the power of **S**.

In section 3, we show how to erase **S** to $\mathbf{F}_\omega$, thus established the consistency of **S**. Type preservation of **S** is proved by applying several techniques: Hardin's interpretation method[11], Hindley-Rosen's commutativity method[12][16], typing as rewrite relation (inspired by [13][17]) and Barendregt's method of proving subject reduction for Curry-style System $\mathbf{F}$[5] . The difficulties arise when we try to handle mutually recursive definitions, together with several non-syntax-directed rules. Although the proof method we developed is specific for our system, we believe that it can be adapted to prove type preservation for other Curry-style systems that make use of mutually recurisve definitions. Finally, we give a proof of $0 \neq 1$ in **S** in section 4, following the approach of the proof of Theorem 1.

2 Church Encoding in S

There are two differences between **S** and **CC**: the first one is the self type, and the second one is the recursive definitions. Intuitively, it seems impossible to obtain a consistent system while allowing recursive definitions. To maintain a well-defined erasure over $\mathbf{F}_\omega$, we restrict the following: there is no recursive definition at term level and the recursive occurrences of type can only be at the erased positions at type level. If we take consistency in the sense that not every type is inhabited, then the well-defineness of the erasure will imply that the type

[3] The Coq code for this example is in appendix A

$\Pi X : *.X$ in **S** will not be inhabited, resulting the consistency of **S**. Self types provide a mechanism for the type to refer to its subject. With these two features, we gain the ability to quantify over the subject in **S**, which is missing in **CC**.

2.1 System S

We use gray boxes to highlight the terms, types and rules that are not in $\mathbf{F}_\omega$ with let-bindings. We also include a full specification of $\mathbf{F}_\omega$ with let-bindings in appendix C.

Definition 1 (Syntax).
Terms t $::=$ $x \mid \lambda x.t \mid tt' \mid \mu t$
Types T $::=$
$X \mid \Pi X : \kappa.T \mid \Pi x : T_1.T_2 \mid \forall x : T_1.T_2 \mid \iota x.T \mid T\ t \mid \lambda X.T \mid \lambda x.T \mid T_1T_2 \mid \mu T$
Kinds κ $::=$ $* \mid \Pi x : T.\kappa \mid \Pi X : \kappa'.\kappa \mid \mu\kappa$
Context Γ $::=$ $\cdot \mid \Gamma, x : T \mid \Gamma, X : \kappa \mid \Gamma, \tilde{\mu}$
Closure μ $::=$ $\{x_i \mapsto t_i\}_{i\in N} \cup \{X_i \mapsto T_i\}_{i\in M}$

Remarks :

- For $\{x_i \mapsto t_i\}_{i\in N}$, we do not allow any recursive (or mutually recursive) definitions, so they are let-bindings. For $\{X_i \mapsto T_i\}_{i\in M}$, we do not allow mutually recursive type definitions, and the recursive occurrence of type can only be at the erased positions, namely, only at the right side of ":" under the binder Π and $\forall$.
- If μ is $\{x_i \mapsto t_i\}_{i\in N} \cup \{X_i \mapsto T_i\}_{i\in M}$, then $\tilde{\mu}$ is $\{(x_i : T_i) \mapsto t_i\}_{i\in N} \cup \{(X_i : \kappa_i) \mapsto T_i\}_{i\in N}$ for some type T_i, kind κ_i.
- For $\{x_i \mapsto t_i\}_{i\in N} \cup \{X_i \mapsto T_i\}_{i\in M}$, we require for any $1 \le i \le n$, the free variable set $\mathrm{FV}(t_i) = \emptyset$ and for any $1 \le i \le m$, the free term and type variable of T_i, $\mathrm{FV}(T_i) \subseteq \mathrm{dom}(\mu)$. We also do not allow any reductions and substitutions inside the closure. We call this the *locality* restriction. Locality is a very important property that will be assumed or used through out this paper. On the other hand, without locality requirement, we predict it is impossible to establish confluence (see [2]).
- $\mathrm{FV}(\mu t) = \mathrm{FV}(t) - \mathrm{dom}(\mu)$, same for $\mu T, \mu\kappa$.

$$\frac{}{\cdot \vdash \mathsf{wf}} \qquad \frac{\Gamma \vdash \mathsf{wf} \quad \Gamma \vdash T : *}{\Gamma, x : T \vdash \mathsf{wf}}$$

$$\frac{\Gamma \vdash \mathsf{wf} \quad \Gamma \vdash \kappa : \Box}{\Gamma, X : \kappa \vdash \mathsf{wf}} \qquad \frac{\Gamma \vdash \mathsf{wf} \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma, \tilde{\mu} \vdash \mathsf{wf}}$$

Fig. 1: Well-formed Context $\boxed{\Gamma \vdash \mathsf{wf}}$

$$\frac{}{\Gamma \vdash * : \Box} \qquad \frac{\Gamma, X : \kappa' \vdash \kappa : \Box \quad \Gamma \vdash \kappa' : \Box}{\Gamma \vdash \Pi X : \kappa'.\kappa : \Box}$$

$$\frac{\Gamma, x : T \vdash \kappa : \Box \quad \Gamma \vdash T : *}{\Gamma \vdash \Pi x : T.\kappa : \Box} \qquad \frac{\Gamma, \tilde{\mu} \vdash \kappa : \Box \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma \vdash \mu\kappa : \Box}$$

Fig. 2: Well-formed Kind $\boxed{\Gamma \vdash \kappa : \Box}$

$$\frac{(X : \kappa) \in \Gamma}{\Gamma \vdash X : \kappa} \qquad \frac{\Gamma \vdash T : \kappa \quad \Gamma \vdash \kappa \cong \kappa' \quad \Gamma \vdash \kappa' : \Box}{\Gamma \vdash T : \kappa'}$$

$$\frac{\Gamma \vdash T_1 : * \quad \Gamma, x : T_1 \vdash T_2 : *}{\Gamma \vdash \Pi x : T_1.T_2 : *} \qquad \frac{\Gamma, X : \kappa \vdash T : * \quad \Gamma \vdash \kappa : \Box}{\Gamma \vdash \Pi X : \kappa.T : *}$$

$$\frac{\Gamma, \tilde{\mu} \vdash T : \kappa \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma \vdash \mu T : \mu\kappa} \qquad \frac{\Gamma, x : \iota x.T \vdash T : *}{\Gamma \vdash \iota x.T : *}\ \mathit{Self}$$

$$\frac{\Gamma, X : \kappa \vdash T : \kappa' \quad \Gamma \vdash \kappa : \Box}{\Gamma \vdash \lambda X.T : \Pi X : \kappa.\kappa'} \qquad \frac{\Gamma, x : T' \vdash T : \kappa \quad \Gamma \vdash T' : *}{\Gamma \vdash \lambda x.T : \Pi x : T'.\kappa}$$

$$\frac{\Gamma \vdash S : \Pi x : T.\kappa \quad \Gamma \vdash t : T}{\Gamma \vdash St : [t/x]\kappa} \qquad \frac{\Gamma \vdash S : \Pi X : \kappa'.\kappa \quad \Gamma \vdash T : \kappa'}{\Gamma \vdash ST : [T/X]\kappa}$$

$$\frac{\Gamma, x : T_1 \vdash T_2 : * \quad \Gamma \vdash T_1 : *}{\Gamma \vdash \forall x : T_1.T_2 : *}$$

Fig. 3: Kinding $\boxed{\Gamma \vdash T : \kappa}$

$$\frac{\Gamma \vdash t : T_1 \quad \Gamma \vdash T_1 \cong T_2 \quad \Gamma \vdash T_2 : *}{\Gamma \vdash t : T_2}\ \mathit{Conv} \qquad \frac{(x : T) \in \Gamma}{\Gamma \vdash x : T}\ \mathit{Var}$$

$$\frac{\Gamma \vdash t : [t/x]T \quad \Gamma \vdash \iota x.T : *}{\Gamma \vdash t : \iota x.T}\ \mathit{SelfGen} \qquad \frac{\Gamma \vdash t : \iota x.T}{\Gamma \vdash t : [t/x]T}\ \mathit{SelfInst}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash T_1 : * \quad x \notin \mathrm{FV}(t)}{\Gamma \vdash t : \forall x : T_1.T_2}\ \mathit{Indx} \qquad \frac{\Gamma \vdash t : \forall x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash t : [t'/x]T_2}\ \mathit{Dex}$$

$$\frac{\Gamma, \tilde{\mu} \vdash t : T \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma \vdash \mu t : \mu T}\ \mathit{Mu} \qquad \frac{\Gamma, X : \kappa \vdash t : T \quad \Gamma \vdash \kappa : \Box}{\Gamma \vdash t : \Pi X : \kappa.T}\ \mathit{Poly}$$

$$\frac{\Gamma \vdash t : \Pi X : \kappa.T \quad \Gamma \vdash T' : \kappa}{\Gamma \vdash t : [T'/X]T}\ \mathit{Inst} \qquad \frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash T_1 : *}{\Gamma \vdash \lambda x.t : \Pi x : T_1.T_2}\ \mathit{Func}$$

$$\frac{\Gamma \vdash t : \Pi x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash tt' : [t'/x]T_2}\ \mathit{App}$$

Fig. 4: Typing $\boxed{\Gamma \vdash t : T}$

Remarks :

- $\Gamma, \tilde{\mu} \vdash \mathsf{ok}$ stands for $\{\Gamma, \tilde{\mu} \vdash t_j : T_j\}_{(t_j:T_j)\in\tilde{\mu}}$ and $\{\Gamma, \tilde{\mu} \vdash T_j : \kappa_j\}_{(T_j:\kappa_j)\in\tilde{\mu}}$.
- $(t_i : T_i) \in \tilde{\mu}$ means $(x_i : T_i) \mapsto t_i \in \tilde{\mu}$ and $(T_i : \kappa_i) \in \tilde{\mu}$ means $(X_i : \kappa_i) \mapsto T_i \in \tilde{\mu}$.
- $\cong$ is the reflexive transitive and symmetry closure of $\to_\beta \cup \to_o \cup \to_\mu$.
- Typing and kinding do not depend on well-formness of the context, so the self type formation rule *self* is not circular in this sense.

$$\frac{(x_i \mapsto t_i) \in \mu}{\Gamma \vdash \mu x_i \to_\beta \mu t_i} \qquad \frac{(x \mapsto t) \in \Gamma}{\Gamma \vdash x \to_\beta t} \qquad \frac{}{\Gamma \vdash (\lambda x.t)t' \to_\beta [t'/x]t}$$

$$\frac{(X_i \mapsto T_i) \in \mu}{\Gamma \vdash \mu X_i \to_\beta \mu T_i} \qquad \frac{(X \mapsto T) \in \Gamma}{\Gamma \vdash X \to_\beta T} \qquad \frac{}{\Gamma \vdash (\lambda x.T)t \to_\beta [t/x]T}$$

$$\frac{}{\Gamma \vdash (\lambda X.T)T' \to_\beta [T'/X]T}$$

Fig. 5: Beta Reduction

$$\frac{\mu \in \Gamma}{\Gamma \vdash \mu t \to_o t} \qquad \frac{\mu \in \Gamma}{\Gamma \vdash \mu T \to_o T}$$

$$\frac{\mathrm{dom}(\mu)\#\mathrm{FV}(t)}{\Gamma \vdash \mu t \to_\mu t} \qquad \frac{}{\Gamma \vdash \mu(\lambda x.t) \to_\mu \lambda x.\mu t}$$

$$\frac{}{\Gamma \vdash \mu(t_1 t_2) \to_\mu (\mu t_1)(\mu t_2)} \qquad \frac{}{\Gamma \vdash \mu(T\ T') \to_\mu (\mu T)(\mu T')}$$

$$\frac{\mathrm{dom}(\mu)\#\mathrm{FV}(T)}{\Gamma \vdash \mu T \to_\mu T} \qquad \frac{}{\Gamma \vdash \mu(\iota x.T) \to_\mu \iota x.\mu T}$$

$$\frac{}{\Gamma \vdash \mu(\forall x : T_1.T_2) \to_\mu \forall x : \mu T_1.\mu T_2} \qquad \frac{}{\Gamma \vdash \mu(\Pi X : \kappa.T) \to_\mu \Pi X : \mu\kappa.\mu T}$$

$$\frac{}{\Gamma \vdash \mu(T\ t) \to_\mu (\mu T)(\mu t)} \qquad \frac{}{\Gamma \vdash \mu(\lambda X.T) \to_\mu \lambda X.\mu T}$$

$$\frac{}{\Gamma \vdash \mu(\lambda x.T) \to_\mu \lambda x.\mu T} \qquad \frac{}{\Gamma \vdash \mu(\Pi x : T_1.T_2) \to_\mu \Pi x : \mu T_1.\mu T_2}$$

Fig. 6: O reduction and Mu Reduction

Remarks:

- $x_i \mapsto t_i \in \tilde{\mu}$ means $(x_i : T_i) \mapsto t_i \in \tilde{\mu}$ for some T_i and $X_i \mapsto T_i \in \tilde{\mu}$ means $(X_i : \kappa_i) \mapsto T_i \in \tilde{\mu}$ for some κ_i.
- We list only the essential reduction rules for terms and types. For the whole specification of reductions, see appendix B.
- One can see that beside beta-reduction, we have mu-reduction to move the closure inside the term or type structure.
- At the type level, we want to discard a closure when it is already in the context. o-reduction allow us to do this. Without this type level reduction, we can not prove the type preservation theorem. Note that we do not allow o-reduction at term level.

2.2 Church Encoding in S

Now let us see some concrete examples of Church encodings in **S**. For convenience, we write $T \to T'$ for $\Pi x : T.T'$ with $x \notin \mathrm{FV}(T')$.

Definition 2 (Church Numerals). *Let $\tilde{\mu}_c$ be the following recursive defintions:*
$(\mathsf{Nat} : *) \mapsto \iota x.\Pi C : (\Pi y : \mathsf{Nat}.*).(\forall n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n))) \to (C\ 0) \to (C\ x)$
$(\mathsf{S} : \mathsf{Nat} \to \mathsf{Nat}) \mapsto \lambda n.\lambda s.\lambda z.s\ (n\ s\ z)$
$(0 : \mathsf{Nat}) \mapsto \lambda s.\lambda z.z$

With $s : \forall n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n)), z : C0, n : \mathsf{Nat}$, we have $\tilde{\mu}_c \vdash \mathsf{wf}$ (using *selfGen* and *selfInst* rules). Also note that the $\tilde{\mu}_c$ satisfies the constraints on recursive definitions.

Theorem 2 (Induction Principle).
$\tilde{\mu}_c \vdash \mathsf{Ind} : \Pi C : (\Pi y : \mathsf{Nat}.*).(\forall n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n))) \to C\ 0 \to \Pi n : \mathsf{Nat}.C\ n$
where $\mathsf{Ind} := \lambda s.\lambda z.\lambda n.n\ s\ z$
with $s : \forall n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n)), z : C0, n : \mathsf{Nat}$.

Proof. Let $\Gamma = \tilde{\mu}_c, C : (\Pi y : \mathsf{Nat}.*), s : \forall n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n)), z : C0, n : \mathsf{Nat}$. Since $n : \mathsf{Nat}$, by *selfInst*, $n : \Pi C : (\Pi y : \mathsf{Nat}.*).(\forall y : \mathsf{Nat}.(C\ y) \to (C\ (\mathsf{S}\ y))) \to (C\ 0) \to (C\ n)$. Thus $n\ s\ z : C\ n$.

It is worth noting that it is really the definition of Nat and the *selfInst* rule that give us the induction principle. We also want to emphasize that this is not derivable in **CC** [7].

Definition 3 (Addition). $m + n := \mathsf{Ind}\ \mathsf{S}\ n\ m$

One can check that $\tilde{\mu}_c \vdash + : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$ by instantiating the C in the type of Ind by $\lambda y.\mathsf{Nat}$, then the type of Ind is $(\mathsf{Nat} \to \mathsf{Nat}) \to \mathsf{Nat} \to (\mathsf{Nat} \to \mathsf{Nat})$.

Definition 4 (Leibniz's Equality). $\mathsf{Eq} := \lambda A[: *].\lambda x[: A].\lambda y[: A].\Pi C : (\Pi y : A.*).Cx \to Cy$.

Note that we use $x =_A y$ to denote $\mathsf{Eq}\ A\ x\ y$. We often write $t = t'$ when the type is clear. One can check that if $\vdash A : *$ and $\vdash x, y : A$, then $\vdash x =_A y : *$.

Theorem 3. $\tilde{\mu}_c \vdash \Pi x : \mathsf{Nat}.x + 0 =_{\mathsf{Nat}} x$

Proof. We prove this by induction. We instantiate C in the type of Ind with $\lambda n.(n + 0) =_{\mathsf{Nat}} n$. So by beta reduction at type level, we have $(\forall n : \mathsf{Nat}.(n + 0 =_{\mathsf{Nat}} n) \to ((\mathsf{S}\ n)+0 =_{\mathsf{Nat}} \mathsf{S}\ n)) \to 0+0 =_{\mathsf{Nat}} 0 \to \Pi n : \mathsf{Nat}.n+0 =_{\mathsf{Nat}} n$. So for base case, we need to show $0+0 =_{\mathsf{Nat}} 0$, which is easy. For step case, we assume $n+0 =_{\mathsf{Nat}} n$ (Induction Hypothesis), and want to show $(\mathsf{S}\ n)+0 =_{\mathsf{Nat}} \mathsf{S}\ n$. Since $(\mathsf{S}\ n)+0 \to_\beta \mathsf{S}\ (n\ \mathsf{S}\ 0) =_\beta \mathsf{S}(n+0)$, by congruence on the induction hypothesis, we have $(\mathsf{S}\ n) + 0 =_{\mathsf{Nat}} \mathsf{S}\ n$. Thus $\Pi x : \mathsf{Nat}.x + 0 =_{\mathsf{Nat}} x$.

The proof above is carrying out inside **S**. It shows how to inhabit the type $\Pi x : \mathsf{Nat}.x + 0 =_{\mathsf{Nat}} x$ given $\tilde{\mu}_c$, using Ind.

For the following vector examples, we use $\mathsf{vec}(U, n)$ as a meta-level notation for a vector type of length n and the elements in the vector are of type U.

Definition 5 (Vector). *Let $\tilde{\mu}_v$ be the following recursive definitions:*
$(\mathsf{vec}(U, n) : *) \mapsto$
$\iota x\ .\Pi C :\ (\Pi p : \mathsf{Nat}.\Pi q : \mathsf{vec}(U, p).*)\ .$
$(\Pi m : \mathsf{Nat}.\Pi u : U.\forall y : \mathsf{vec}(U, m).(C\ m\ y\ \to C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y)))$
$\to C\ 0\ \mathsf{nil} \to C\ n\ \ x$

$(\mathsf{nil} : \mathsf{vec}(U, 0)) \mapsto \lambda y.\lambda x.x$
$(\mathsf{cons} : \Pi n : \mathsf{Nat}.U \to \mathsf{vec}(U, n) \to \mathsf{vec}(U, \mathsf{S}n)) \mapsto \lambda n.\lambda v.\lambda l.\lambda y.\lambda x.y\ n\ v\ (l\ y\ x)$
where $n : \mathsf{Nat}, v : U, l : \mathsf{vec}(U, n), y : \Pi m : \mathsf{Nat}.\Pi u : U.\forall y : \mathsf{vec}(U, m).(C\ m\ y \to C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y)), x : C\ 0\ \mathsf{nil}$.

Typing: It is easy to see that nil is typable to $\mathsf{vec}(U, 0)$. Now we show how cons is typable to $\Pi n : \mathsf{Nat}.U \to \mathsf{vec}(U, n) \to \mathsf{vec}(U, \mathsf{S}n)$. We can see that $l\ y\ x : C\ n\ l$. After the instantiation with l, the type of $y\ n\ v$ is $C\ n\ l \to C\ (\mathsf{S}n)\ (\mathsf{cons}\ n\ v\ l)$. So $y\ n\ v\ (l\ y\ x) : C\ (\mathsf{S}n)\ (\mathsf{cons}\ n\ v\ l)$. So $\lambda y.\lambda x.y\ n\ v\ (l\ y\ x)$: $\Pi C : (\Pi p : \mathsf{Nat}.\Pi q : \mathsf{vec}(U, p).*).(\Pi m : \mathsf{Nat}.\Pi u : U.\forall y : \mathsf{vec}(U, m).(C\ m\ y \to C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y))) \to C\ 0\ \mathsf{nil} \to C\ (\mathsf{S}n)\ (\lambda y.\lambda x.y\ n\ v\ (l\ y\ x))$. So by *selfGen*, we have $\lambda y.\lambda x.y\ n\ v\ (l\ y\ x) : \mathsf{vec}(U, \mathsf{S}n)$. Thus $\mathsf{cons} : \Pi n : \mathsf{Nat}.U \to \mathsf{vec}(U, n) \to \mathsf{vec}(U, \mathsf{S}n)$.

Definition 6 (Induction Principle for Vector).
$\tilde{\mu}_v \vdash \mathsf{Ind}(U, n) :$
$\Pi C : (\Pi p : \mathsf{Nat}.\Pi q : \mathsf{vec}(U, p).*).$
$(\Pi m : \mathsf{Nat}.\Pi u : U.\forall y : \mathsf{vec}(U, m).(C\ m\ y \to C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y)))$
$\to C\ 0\ \mathsf{nil} \to \Pi x : \mathsf{vec}(U, n).(C\ n\ x)$
where $\mathsf{Ind}(U, n) := \lambda s.\lambda z.\lambda x.x\ s\ z$
$s : \Pi C : (\Pi p : \mathsf{Nat}.\Pi q : \mathsf{vec}(U, p).*).(\Pi m : \mathsf{Nat}.\Pi u : U.\forall y : \mathsf{vec}(U, m).(C\ m\ y \to C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y))), z : C\ 0\ \mathsf{nil}, x : \mathsf{vec}(U, n)$.

Definition 7 (Append).
$\tilde{\mu}_v \vdash \mathsf{app} : \Pi n_1 : \mathsf{Nat}.\Pi n_2 : \mathsf{Nat}.\mathsf{vec}(U, n_1) \to \mathsf{vec}(U, n_2) \to \mathsf{vec}(U, n_1 + n_2)$
where $\mathsf{app} := \lambda n_1.\lambda n_2.\lambda l_1.\lambda l_2.\mathsf{Ind}(U, n_1)(\lambda n.\lambda x.\lambda v.\mathsf{cons}(n + n_2)\ x\ v)\ l_2\ l_1$.

Typing: We want to show $\mathsf{app} : \Pi n_1 : \mathsf{Nat}.\Pi n_2 : \mathsf{Nat}.\mathsf{vec}(U, n_1) \to \mathsf{vec}(U, n_2) \to \mathsf{vec}(U, n_1 + n_2)$. Observe that $\lambda n.\lambda x.\lambda v.\mathsf{cons}(n + n_2)\ x\ v : \Pi n : \mathsf{Nat}.\Pi x : U.\mathsf{vec}(U, n+n_2) \to \mathsf{vec}(U, n+n_2+1)$. We instantiate $C := \lambda y.(\lambda x.\mathsf{vec}(U, y+n_2))$, where x free over $\mathsf{vec}(U, y + n_2)$, in $\mathsf{Ind}(U, n_1)$. By beta reductions, we get $\mathsf{Ind}(U, n_1) : (\Pi m : \mathsf{Nat}.\Pi u : U.\forall y : \mathsf{vec}(U, m).(\mathsf{vec}(U, m + n_2) \to \mathsf{vec}(U, (\mathsf{S}m) + n_2)) \to \mathsf{vec}(U, 0 + n_2) \to \Pi x : \mathsf{vec}(U, n_1).\mathsf{vec}(U, n_1 + n_2)$.
So $\mathsf{Ind}(U, n_1)\ (\lambda n.\lambda x.\lambda v.\mathsf{cons}(n+n_2)\ x\ v) : \mathsf{vec}(U, 0+n_2) \to \Pi x : \mathsf{vec}(U, n_1).\mathsf{vec}(U, n_1+n_2)$. We assume $l_1 : \mathsf{vec}(U, n_1), l_2 : \mathsf{vec}(U, n_2)$. Thus $\mathsf{ID}(U, n_1)\ (\lambda n.\lambda x.\lambda v.\mathsf{cons}(n + n_2)\ x\ v)\ l_2\ l_1 : \mathsf{vec}(U, n_1 + n_2)$.

Theorem 4 (Associativity).
$\tilde{\mu}_v \vdash \Pi(n_1.n_2.n_3 : \mathsf{Nat}).\Pi(v_1 : \mathsf{vec}(U, n_1).v_2 : \mathsf{vec}(U, n_2).v_3 : \mathsf{vec}(U, n_3)).$
$(\mathsf{app}\ n_1\ (n_2 + n_3)\ v_1\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (n_1 + n_2)\ n_3\ (\mathsf{app}\ n_1\ n_2\ v_1\ v_2)\ v_3$

Proof. Use $\mathsf{Ind}(U, n_1)$. We will not go through the proof here.

3 Metatheory

We will first outline the erasure from **S** to $\mathbf{F}_\omega$ with let-bindings, thus establishing the strong normalization property for the terms. Note that we assume $\mathbf{F}_\omega$ with nonrecursive

let-bindings is consistent, since it can be easily reduced to $\mathbf{F}_\omega$ by substituting all the defined symbols with their defining terms. Since we are dealing with a Curry-style system, type preservation is a hard technical problem. We will then outline our method to prove type preservation for **S**. The proofs in this section can be found in appendix D, E, F.

3.1 Erasure to $\mathbf{F}_\omega$

The interesting cases in the following definition are highlighted in gray boxes.

Definition 8 (Erasure).

$$
\begin{array}{ll}
F(*) := * & F(\Pi x : T.\kappa) := F(\kappa) \\
F(\Pi X : \kappa'.\kappa) := F(\kappa') \to F(\kappa) & F(\mu\kappa) := F(\kappa) \\
F(X) := X & F(\iota x.T) := F(T) \\
F(\Pi X : \kappa.T) := \Pi X : F(\kappa).F(T) & F(\Pi x : T_1.T_2) := F(T_1) \to F(T_2) \\
F(\forall x : T_1.T_2) := F(T_2) & F(T_1T_2) := F(T_1)F(T_2) \\
F(\lambda X.T) := \lambda X.F(T) & F(T\ t) := F(T) \\
F(\lambda x.T) := F(T) & F(\mu T) := F(\mu)F(T) \\
F(x) := x & F(\lambda x.t) := \lambda x.F(t) \\
F(tt') := F(t)F(t') & F(\mu t) := F(\mu)F(t) \\
F(x_i \mapsto t_i, \mu) := x_i \mapsto F(t_i), F(\mu) & F(X_i \mapsto T_i, \mu) := X_i \mapsto F(T_i), F(\mu) \\
F((X : \kappa) \mapsto T, \Gamma) := X : F(\kappa) \mapsto F(T), F(\Gamma) & F(X : \kappa, \Gamma) := X : F(\kappa), F(\Gamma) \\
F((x : T) \mapsto t, \Gamma) := x : F(T) \mapsto F(t), F(\Gamma) & F(x : T, \Gamma) := x : F(T), F(\Gamma)
\end{array}
$$

At term level, we have term μt, where μ may contain type definitions, thus we need to define erasure on terms as well. The erasure of $\iota x.T$ is $F(T)$: we simply erase all the terms at type level. With our restriction on closure, $F(\mu)$ will be an ordinary let-bindings.

Let $\mathcal{M}_s, \mathcal{T}_s, \mathcal{K}_s, \Psi$ denote the set of terms, types, kinds and closures in **S**, let $\mathcal{M}_\omega, \mathcal{T}_\omega, \mathcal{K}_\omega, \Theta$ denote the set of terms, types, kinds and let-binding in $\mathbf{F}_\omega$.

Lemma 1 (Well-definedness of Erasure).

1. *If $\kappa \in \mathcal{K}_s$, then $F(\kappa) \in \mathcal{K}_\omega$.*
2. *If $t \in \mathcal{M}_s$, then $F(t) \in \mathcal{M}_\omega$.*
3. *If $T \in \mathcal{T}_s$, then $F(T) \in \mathcal{T}_\omega$.*
4. *If $\mu \in \Psi$, then $F(\mu) \in \Theta$.*

Lemma 2. *We have following equalities: $F(\kappa) \equiv F([t/x]\kappa)$, $F([T/X]\kappa) \equiv F(\kappa)$, $F([t/x]T) \equiv F(T)$, $F([T'/X]T) \equiv [F(T')/X]F(T)$ and $F([t'/x]t) \equiv [F(t')/x]F(t)$.*

Lemma 3. *If $\Gamma \vdash \kappa \cong \kappa'$, then $F(\kappa) \equiv F(\kappa')$.*

Let $\hookrightarrow_{o,\beta,\mu}$ denote the reflexive closure of $\to_{o,\beta,\mu}$[4].

Lemma 4. *If $\Gamma \vdash T \to_{o,\beta,\mu} T'$, then $F(\Gamma) \vdash F(T) \hookrightarrow_{o,\beta,\mu} F(T')$.*

[4] $\to_{o,\beta,\mu}$ denotes $\to_o \cup \to_\beta \cup \to_\mu$. We will use this convention throughout the paper.

Proof. By induction on derivation of $\Gamma \vdash T \rightarrow_{o,\beta,\mu} T'$. We use lemma 3, lemma 2 above.

Lemma 5. *If* $\Gamma \vdash t \rightarrow_{\beta,\mu} t'$, *then* $F(\Gamma) \vdash F(t) \rightarrow_{\beta,\mu} F(t')$.

Proof. By induction on derivation.

Note that reductions inside the closure are not allowed. If one allows reduction inside closure, and then we would need to revise the lemma to: if $\Gamma \vdash t \rightarrow_{\beta,\mu} t'$, then $F(\Gamma) \vdash F(t) \hookrightarrow_{\beta,\mu} F(t')$. This means that the erasure erases some of the redex in the term, and then we would not be able to establish the strong normalization by the erasure theorem.

Theorem 5 (Erasure Theorem).

1. *If* $\Gamma \vdash T : \kappa$, *then* $F(\Gamma) \vdash F(T) : F(\kappa)$.
2. *If* $\Gamma \vdash t : T$, *then* $F(\Gamma) \vdash F(t) : F(T)$.

Lemma 5 and Theorem 5 are essential to conclude strong normalization at term level, since all the reduction information at term level is preserved by the erasure function. However, at type level, we will not have strong normalization(observe the type of Nat is diverging), yet **S** is still consistent in the sense that not every type is inhabited.

3.2 About Type Preservation Proof

In order to prove type preservation for **S**, one needs to know how to combine pieces together. The complications are due to closure-related reductions together with several non-syntactic directed rules. We can prove preservation by adopting several techniques. Informally, one first needs to establish the confluence property for type level reductions, then we need a straightforward extension of Barendregt's method for proving Curry style **F** to handle the implicit product. Barendregt's method allows typing to *quotient out* the *transformations* of $\Pi X : \kappa.T$ and $\forall x : T.T'$. Then confluence for the type level reductions will ensure that if $\Pi x : T.T'$ can be transformed to $\Pi x : T_1.T_2$, then it must be that T can be transformed to T_1 and T' can be transformed to T_2. Thus, one establishes the so called *compatibility* property for the explicit dependent types, which is the key to achieve type preservation.

We will first prove several confluence results (this process is called *confluence analysis*), then we develop a technique to quotient out the transformations of $\forall x : T.T'$ and $\Pi X : \kappa.T$ (called *morph analysis*). Finally, we establish the compatibility theorem, which is enough for proving preservation.

We want to emphasize that we give the outline of the whole process because we want to convince readers that the method itself is worth noting and can be adapted to many other Curry-style systems with mutually recursive definitions.

3.3 Confluence Analysis

First observe that the *selfGen* rule and *selfInst* rule are mutually inversed, so we can model the change of self type by the following rewriting point of view.

Definition 9.
$\Gamma \vdash T_1 \to_\iota T_2$ *if* $T_1 \equiv \iota x.T'$ *and* $T_2 \equiv [t/x]T'$ *for some fix* $t \in \mathcal{M}_s$.

Note that $\to_\iota$ models the *selfInst* rule, $\to_\iota^{-1}$ models the *selfGen* rule. Importantly, the notion of ι-reduction does not build in structure congruent, namely, we do not allow reduction rules like: if $T \to_\iota T'$, then $\lambda x.T \to_\iota \lambda x.T'$. The purpose of ι-reduction is to emulate the typing rule *selfInst* and *selfGen*. The goal of confluence analysis is to prove the following theorem.

Theorem 6 (Fundamental Theorem). $\to_{o,\iota,\beta,\mu}$ *is confluent.*

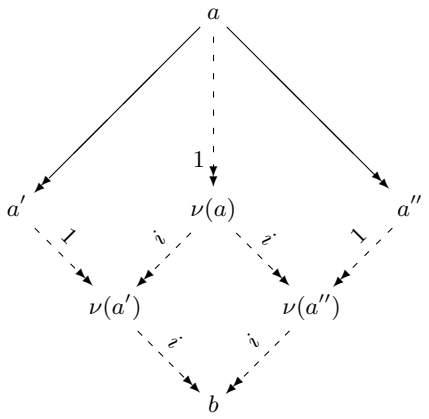
Of course, one can not prove this theorem by brute force. We will first show $\to_{\beta,\mu}$ is confluence by adopting Hardin's interpretation lemma, then we will make use of Hindley-Rossen's commutativity theorem to show $\to_o$ and $\to_\iota$ are by themself confluence and commute with $\to_{\beta,\mu}$. Thus we can conclude Theorem 6. We use $\twoheadrightarrow$ to denote the reflexive symmetric transitive closure of $\to$.

Lemma 6 (Interpretation lemma [11]). *Let* $\to$ *be* $\to_1 \cup \to_2$, $\to_1$ *being confluent and strongly normalizing. We denote by* $\nu(a)$ *the* $\to_1$*-normal form of* a. *Suppose that there is some relation* $\to_i$ *on the* $\to_1$*-normal forms satisfying:*

$\to_i \subseteq \twoheadrightarrow$, *and* $a \to_2 b$ *implies* $\nu(a) \twoheadrightarrow_i \nu(b)$ (†)

Then the confluence of $\to_i$ *implies the confluence of* $\to$.

Proof. Suppose $\to_i$ is confluent. Assume $a \twoheadrightarrow a'$ and $a \twoheadrightarrow a''$. So by (†), $\nu(a) \twoheadrightarrow_i \nu(a')$ and $\nu(a) \twoheadrightarrow_i \nu(a'')$. Note that $t \to_1^* t'$ implies $\nu(t) = \nu(t')$(By the confluence and strong normalization of $\to_1$). By the confluence of $\to_i$, there exists a b such that $\nu(a') \twoheadrightarrow_i b$ and $\nu(a'') \twoheadrightarrow_i b$. Since $\to_i, \to_1 \subseteq \twoheadrightarrow$, we get $a' \twoheadrightarrow \nu(a') \twoheadrightarrow b$ and $a'' \twoheadrightarrow \nu(a'') \twoheadrightarrow b$. Hence $\to$ is confluent.



The reason that interpretation method works while the other direct methods fail is that it allows us to quotient out the $\to_1$-reduction, we only need to focus on proving the confluence of $\to_i$. This is essential because $\to_{\beta,\mu}$ can not be directly parallelized, namely, one can not use Tait-Martin Löf's method (reported in [3]) directly to prove

the confluence of $\rightarrow_{\beta,\mu}$, because the paralleled version does not enjoy diamond property. Interestingly, after modulo the $\rightarrow_\mu$-reduction, we introduce a new reduction $\rightarrow_{\beta\mu}$ (corresponds to $\rightarrow_i$), and we can then use the parallel reduction method to prove confluence of $\rightarrow_{\beta\mu}$.

Lemma 7. *$\rightarrow_\mu$ is strongly normalizing and confluent.*

So $\rightarrow_\mu$, $\rightarrow_\beta$ correspond to $\rightarrow_1$ and $\rightarrow_2$ in the interpretation lemma. Since $\rightarrow_\mu$ is strongly normalizing and confluent, we can define a normalization function which effectively computes the mu-normal form.

Definition 10 (Mu-normal Forms).
Mu-normal Terms $n ::= x \mid \mu x_i \mid \lambda x.n \mid nn'$
Mu-normal Types $N :=$
$X \mid \mu X_i \mid \Pi x : N.N' \mid \iota x.N \mid \Pi X : K.N \mid \forall x : N.N' \mid N\ n \mid NN' \mid \lambda X.N \mid \lambda x.N$
Mu-normal Kinds $K := * \mid \Pi x : N.K \mid \Pi X : K.K'$

Note: For the μx_i and μX_i in definition 10, we assume $x_i, X_i \in \mathrm{dom}(\mu)$. Let $\boldsymbol{\mu}t$ denote $\mu_1 ... \mu_n t$ with $n \geq 1$.

Definition 11 (Normalization Function for Term).

$\nu(x) := x$ $\quad$ $\nu(\lambda y.t) := \lambda y.\nu(t)$
$\nu(t_1 t_2) := \nu(t_1)\nu(t_2)$ $\quad$ $\nu(\boldsymbol{\mu}y) := y$ *if* $y \notin \mathrm{dom}(\boldsymbol{\mu})$
$\nu(\boldsymbol{\mu}y) := \mu_i y$ *if* $y \in \mathrm{dom}(\mu_i)$ $\quad$ $\nu(\boldsymbol{\mu}(tt')) := \nu(\boldsymbol{\mu}t)\nu(\boldsymbol{\mu}t')$
$\nu(\boldsymbol{\mu}(\lambda x.t)) := \lambda x.\nu(\boldsymbol{\mu}t)$

Definition 12 (Normalization Function for Types).

$\nu(X) := X$ $\quad$ $\nu(\lambda y.T) := \lambda y.\nu(T)$
$\nu(T_1 T_2) := \nu(T_1)\nu(T_2)$ $\quad$ $\nu(\lambda X.T) := \lambda X.\nu(T)$
$\nu(Tt) := \nu(T)\nu(t)$ $\quad$ $\nu(\iota x.T) := \iota x.\nu(T)$
$\nu(\Pi X : \kappa.T) := \Pi X : \nu(\kappa).\nu(T)$ $\quad$ $\nu(\Pi x : T.T') := \Pi x : \nu(T).\nu(T')$
$\nu(\forall x : T.T') := \forall x : \nu(T).\nu(T')$ $\quad$ $\nu(\boldsymbol{\mu}X) := X$ *if* $X \notin \mathrm{dom}(\boldsymbol{\mu})$
$\nu(\boldsymbol{\mu}X) := \mu_i X$ *if* $X \in \mathrm{dom}(\mu_i)$ $\quad$ $\nu(\boldsymbol{\mu}(\iota x.T)) := \iota x.\nu(\boldsymbol{\mu}T)$
$\nu(\boldsymbol{\mu}(\Pi x : T.T')) := \Pi x : \nu(\boldsymbol{\mu}T).\nu(\boldsymbol{\mu}T')$ $\quad$ $\nu(\boldsymbol{\mu}(\forall x : T.T')) := \forall x : \nu(\boldsymbol{\mu}T).\nu(\boldsymbol{\mu}T')$
$\nu(\boldsymbol{\mu}(\Pi X : \kappa.T')) := \Pi X : \nu(\boldsymbol{\mu}\kappa).\nu(\boldsymbol{\mu}T')$ $\quad$ $\nu(\boldsymbol{\mu}(TT')) := \nu(\boldsymbol{\mu}T)\nu(\boldsymbol{\mu}T')$
$\nu(\boldsymbol{\mu}(Tt)) := \nu(\boldsymbol{\mu}T)\nu(\boldsymbol{\mu}t)$ $\quad$ $\nu(\boldsymbol{\mu}(\lambda x.T)) := \lambda x.\nu(\boldsymbol{\mu}T)$
$\nu(\boldsymbol{\mu}(\lambda X.T)) := \lambda X.\nu(\boldsymbol{\mu}T)$

Definition 13 (Normalization Function for Kinds).

$\nu(*) := *$ $\quad$ $\nu(\Pi x : T.\kappa) := \Pi x : \nu(T).\nu(\kappa)$
$\nu(\Pi X : \kappa'.\kappa) := \Pi X : \nu(\kappa').\nu(\kappa)$ $\quad$ $\nu(\boldsymbol{\mu}*) := *$
$\nu(\boldsymbol{\mu}(\Pi x : T.\kappa)) := \Pi x : \nu(\boldsymbol{\mu}T).\nu(\boldsymbol{\mu}\kappa)$ $\quad$ $\nu(\boldsymbol{\mu}(\Pi X : \kappa'.\kappa)) := \Pi X : \nu(\boldsymbol{\mu}\kappa').\nu(\boldsymbol{\mu}\kappa)$

After the definitions of mu-normalization functions, we shall devise a new notion of reduction on mu-normal form, then show that this reduction is confluent (corresponds to $\rightarrow_i$ in the interpretation lemma and satisfying the † property), thus by the interpretation lemma, we can show $\rightarrow_{\beta,\mu}$ is confluent. A natural way to define reduction on mu-normal form is that right after a beta-reduction, one immediately mu-normalizes the contractum, which can form a notion of reduction on mu-normal form.

Definition 14 (β Reduction on μ-normal Forms).

$$\frac{\Gamma \vdash n \rightarrow_\beta t}{\Gamma \vdash n \rightarrow_{\beta\mu} \nu(t)} \quad \frac{\Gamma \vdash N \rightarrow_\beta T}{\Gamma \vdash N \rightarrow_{\beta\mu} \nu(T)} \quad \frac{\Gamma \vdash K \rightarrow_\beta \kappa'}{\Gamma \vdash K \rightarrow_{\beta\mu} \nu(\kappa')}$$

The following lemma shows that $\rightarrow_{\beta\mu}$ is indeed the reduction we want to fullfill the role of $\rightarrow_i$ in the interpretation lemma.

Lemma 8.

- *If* $\Gamma \vdash t \rightarrow_\beta t'$, *then* $\Gamma \vdash \nu(t) \rightarrow_{\beta\mu} \nu(t')$.
- *If* $\Gamma \vdash T \rightarrow_\beta T'$, *then* $\Gamma \vdash \nu(T) \rightarrow_{\beta\mu} \nu(T')$.
- *If* $\Gamma \vdash \kappa \rightarrow_\beta \kappa'$, *then* $\Gamma \vdash \nu(\kappa) \rightarrow_{\beta\mu} \nu(\kappa')$.

Lemma 9. $\rightarrow_{\beta\mu}$ *is confluent.*

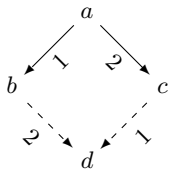
Theorem 7. $\rightarrow_{\beta,\mu}$ *is confluent.*

Proof. By lemma 7, lemma 8, lemma 9 and the interpretation lemma.

One can use the parallel reduction method to prove the confluence of $\rightarrow_{\beta\mu}$, while it is hard to directly prove the confluence of $\rightarrow_{\beta,\mu}$. Once we established the confluence of $\rightarrow_{\beta,\mu}$, it will be easier to analyze the interactions between $\rightarrow_{\beta,\mu}$ and $\rightarrow_{\iota,o}$. We will make extensive use of the notion of *commutativity*, which provides a simple way to analyze the confluence of a reduction system that has several confluent subreductons.

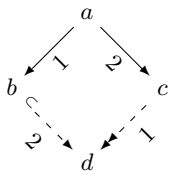
Theorem 8. $\rightarrow_\iota, \rightarrow_o$ *both are confluent.*

Definition 15 (Commutativity). *Let* $\rightarrow_1, \rightarrow_2$ *be two notions of reduction.* $\rightarrow_1$ *(strong) commutes with* $\rightarrow_2$ *if the following diagram holds.*



Proposition 1 (Hindley-Rosen[5]). *Let* $\rightarrow_1, \rightarrow_2$ *be two notions of reduction. Suppose both* $\rightarrow_1$ *and* $\rightarrow_2$ *are confluent, and* $\rightarrow_1^*$ *commutes with* $\rightarrow_2^*$. *Then* $\rightarrow_1 \cup \rightarrow_2$ *is confluent.*

Proposition 2 (Weak Commutativity[6]). *Let* $\hookrightarrow$ *denote the reflexive closure of* $\rightarrow$. *Let* $\rightarrow_1, \rightarrow_2$ *be two notions of reduction.* $\rightarrow_1$ *weak commutes with* $\rightarrow_2$ *if the following diagram holds*



If $\rightarrow_1$ *weak commutes with* $\rightarrow_2$, *then* $\rightarrow_1^*$ *and* $\rightarrow_2^*$ *commute.*

[5] The proposition is taken from Barendregt's [3] Chapter 3.3, page 64.
[6] The proposition is taken from Barendregt's [3] Chapter 3.3, page 65.

Lemma 10. *$\to_{\beta,\mu}$ commutes with $\to_\iota$. Thus $\to_{\beta,\mu,\iota}$ is confluent.*

Lemma 11. *$\to_o$ commutes with $\to_\iota$, weak commutes with $\to_\beta$ and $\to_\mu$.*

Theorem 9 (Fundamental Theorem). *$\to_{o,\iota,\beta,\mu}$ is confluent.*

The following theorem shows an application of fundamental theorem.

Theorem 10 (ι-elimination). *If $\Gamma \vdash \Pi x : T_1.T_2 =_{\beta,\mu,\iota,o} \Pi x : T_1'.T_2'$, then $\Gamma \vdash T_1 =_{\beta,\mu,o} T_1'$ and $\Gamma \vdash T_2 =_{\beta,\mu,o} T_2'$.*

Proof. If $\Gamma \vdash \Pi x : T_1.T_2 =_{\beta,\mu,\iota,o} \Pi x : T_1'.T_2'$, then by the confluence of $\to_{\beta,\mu,\iota,o}$, there exists a T such that $\Gamma \vdash \Pi x : T_1.T_2(\to_{o,\iota,\beta,\mu})^*T$ and $\Gamma \vdash \Pi x : T_1'.T_2'(\to_{o,\iota,\beta,\mu})^*T$. Since all the reductions on $\Pi x : T_1.T_2$ preserve the structure of the dependent type, one will never have a chance to use $\to_\iota$-reduction, thus $\Gamma \vdash \Pi x : T_1.T_2(\to_{o,\beta,\mu})^*T$ and $\Gamma \vdash \Pi x : T_1'.T_2'(\to_{o,\beta,\mu})^*T$. So T must be of the form $\Pi x : T_3.T_4$. And $\Gamma \vdash T_1(\to_{o,\beta,\mu})^*T_3$, $\Gamma \vdash T_1'(\to_{o,\beta,\mu})^*T_3$, $\Gamma \vdash T_2(\to_{o,\beta,\mu})^*T_4$ and $\Gamma \vdash T_2'(\to_{o,\beta,\mu})^*T_4$. Finally, we have $\Gamma \vdash T_1 =_{\beta,\mu,o} T_1'$ and $\Gamma \vdash T_2 =_{\beta,\mu,o} T_2'$.

3.4 Morph Analysis

The confluence analysis and the rewriting point of view on the nonsyntactic directed rules can not be extended to deal with polymorphism. For example, $\Pi X : \kappa.X$ can be instantiated either to T or to $T \to T$. So polymorphic instantiation as a rewrite relation is not confluent. The only known syntactic method[7] to deal with preservation proof for Curry-style System **F** is Barendregt's method[5]. We will extend his method to handle the instantiation of $\forall x : T.T'$. We call this style of analysis *morph analysis*.

Definition 16 (Morphing Relations).

- $([\Gamma], T_1) \to_i ([\Gamma], T_2)$ *if* $T_1 \equiv \Pi X : \kappa.T'$ *and* $T_2 \equiv [T/X]T'$ *for some* T *such that* $\Gamma \vdash T : \kappa$.
- $([\Gamma, X : \kappa], T_1) \to_g ([\Gamma], T_2)$ *if* $T_2 \equiv \Pi X : \kappa.T_1$ *and* $\Gamma \vdash \kappa : \Box$.
- $([\Gamma], T_1) \to_I ([\Gamma], T_2)$ *if* $T_1 \equiv \forall x : T.T'$ *and* $T_2 \equiv [t/x]T'$ *for some* t *such that* $\Gamma \vdash t : T$.
- $([\Gamma, x : T], T_1) \to_G ([\Gamma], T_2)$ *if* $T_2 \equiv \forall x : T.T_1$ *and* $\Gamma \vdash T : *$.

One can view morphing relations as a way to model nonsyntactic directed typing rules. Note that morphing relations are not intended to be viewed as rewrite relation. Instead of proving confluence for these morphing relations, we try to use substitutions to *summarize* the effects of a sequence of morphing relations. Before we do that, first we "lift" $=_{\mu,\beta,o,\iota}$ to a form of morphing relation.

Definition 17. $([\Gamma], T) =_{\mu,\beta,o,\iota} ([\Gamma], T')$ *if* $\Gamma \vdash T =_{\mu,\beta,o,\iota} T'$ *and* $\Gamma \vdash T : *$ *and* $\Gamma \vdash T' : *$.

The best way to understand the following E, G mapings is through understanding the lemma 13 and the lemma 14. They give concrete demonstrations of how to *summarize* a sequence of morphing relations.

[7] According to our knowlege.

Definition 18.

$E(\Pi X : \kappa.T) := E(T)$ $E(X) := X$ $E(\Pi x : T_1.T_2) := \Pi x : T_1.T_2$
$E(\lambda X.T) := \lambda X.T$ $E(T_1T_2) := T_1T_2$ $E(\forall x : T'.T) := \forall x : T'.T$
$E(\iota x.T) := \iota x.T$ $E(T\ t) := T\ t$ $E(\lambda x.T) := \lambda x.T$
$E(\mu T) := \mu T$

Definition 19.

$G(\Pi X : \kappa.T) := \Pi X : \kappa.T$ $G(X) := X$ $G(\Pi x : T_1.T_2) := \Pi x : T_1.T_2$
$G(\lambda X.T) := \lambda X.T$ $G(T_1T_2) := T_1T_2$ $G(\forall x : T'.T) := G(T)$
$G(\iota x.T) := \iota x.T$ $G(T\ t) := T\ t$ $G(\lambda x.T) := \lambda x.T$
$G(\mu T) := \mu T$

Lemma 12. *$E([T'/X]T) \equiv [T''/X]E(T)$ for some T''; $G([t/x]T) \equiv [t/x]G(T)$.*

Proof. By induction on the structure of T.

Lemma 13. *If $([\Gamma], T) \rightarrow^*_{i,g} ([\Gamma'], T')$, then there exists a type substitution σ such that $\sigma E(T) \equiv E(T')$.*

Proof. It suffices to consider $([\Gamma], T) \rightarrow_{i,g} ([\Gamma'], T')$. If $T' \equiv \Pi X : \kappa.T$ and $\Gamma = \Gamma', X : \kappa$, then $E(T') \equiv E(T)$. If $T \equiv \Pi X : \kappa.T_1$ and $T' \equiv [T''/X]T_1$ and $\Gamma = \Gamma'$, then $E(T) \equiv E(T_1)$. By lemma 12 above, we know $E(T') \equiv E([T''/X]T_1) \equiv [T_2/X]E(T_1)$ for some T_2.

Lemma 14. *If $([\Gamma], T) \rightarrow^*_{I,G} ([\Gamma'], T')$, then there exists a term substitution δ such that $\delta G(T) \equiv G(T')$.*

Proof. It suffices to consider $([\Gamma], T) \rightarrow_{I,G} ([\Gamma'], T')$. If $T' \equiv \forall x : T_1.T$ and $\Gamma = \Gamma', x : T_1$, then $G(T') \equiv G(T)$. If $T \equiv \forall x : T_2.T_1$ and $T' \equiv [t/x]T_1$ and $\Gamma = \Gamma'$, then $E(T) \equiv E(T_1)$. By lemma 12 above, we know $E(T') \equiv E([t/x]T_1) \equiv [t/x]E(T_1)$.

Lemma 15. *If $([\Gamma], \Pi x : T_1.T_2) \rightarrow^*_{i,g} ([\Gamma'], \Pi x : T_1'.T_2')$, then there exists a type substitution σ such that $\sigma(\Pi x : T_1.T_2) \equiv \Pi x : T_1'.T_2'$.*

Proof. By lemma 13 above.

Lemma 16. *If $([\Gamma], \Pi x : T_1.T_2) \rightarrow^*_{I,G} ([\Gamma'], \Pi x : T_1'.T_2')$, then there exists a term substitution δ such that $\delta(\Pi x : T_1.T_2) \equiv \Pi x : T_1'.T_2'$.*

Proof. By lemma 14 above.

Let $\rightarrow^*_{o,\iota,\beta,\mu,i,g,I,G}$ denote $(\rightarrow_{i,g,I,G} \cup =_{o,\iota,\beta,\mu})^*$. Let $\rightarrow_{o,\iota,\beta,\mu,i,g,I,G}$ denote $\rightarrow_{i,g,I,G} \cup =_{o,\iota,\beta,\mu}$. It is not exaggerated that the goal of confluence analysis and morph analysis is to establish the following *compatibility* theorem.

Theorem 11 (Compatibility). *If $([\Gamma], \Pi x : T_1.T_2) \rightarrow^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma'], \Pi x : T_1'.T_2')$, then there exists a mixed substitution[8] ϕ such that $([\Gamma], \phi(\Pi x : T_1.T_2)) =_{o,\iota,\beta,\mu} ([\Gamma], \Pi x : T_1'.T_2')$. Thus $\Gamma \vdash \phi T_1 =_{o,\beta,\mu} T_1'$ and $\Gamma \vdash \phi T_2 =_{o,\beta,\mu} T_2'$(theorem 10).*

Proof. By lemma 16 and lemma 15. And we make use of the fact that if $\Gamma \vdash t =_{o,\iota,\beta,\mu} t'$, then for any mixed substitution ϕ, we have $\Gamma \vdash \phi t =_{o,\iota,\beta,\mu} \phi t'$.

[8] A substitution that contains both term substitution and type substitution.

3.5 Type Preservation Result

The proof of type preservation proceeds as usual. The inversion lemma and substitution lemma are standard. Note that in the final preservation proof, we use the compatibility theorem.

Lemma 17 (Inversion).

- *If* $\Gamma \vdash x : T$, *then exist* Δ, T_1 *such that* $([\Gamma, \Delta], T_1) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$ *and* $(x : T_1) \in \Gamma$.
- *If* $\Gamma \vdash t_1 t_2 : T$, *then exist* Δ, T_1, T_2 *such that* $\Gamma, \Delta \vdash t_1 : \Pi x : T_1.T_2$ *and* $\Gamma, \Delta \vdash t_2 : T_1$ *and* $([\Gamma, \Delta], [t_2/x]T_2) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$.
- *If* $\Gamma \vdash \lambda x.t : T$, *then exist* Δ, T_1, T_2 *such that* $\Gamma, \Delta, x : T_1 \vdash t : T_2$ *and* $([\Gamma, \Delta], \Pi x : T_1.T_2) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$.
- *If* $\Gamma \vdash \mu t : T$, *then exist* Δ, T_1 *such that* $\Gamma, \Delta, \tilde{\mu} \vdash t : T_1$ *and* $([\Gamma, \Delta], \mu T_1) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$.

Lemma 18 (Substitution).

1. *If* $\Gamma \vdash t : T$, *then for any mixed substitution* ϕ *with* $\mathrm{dom}(\phi) \# \mathrm{FV}(t)$, $\phi\Gamma \vdash t : \phi T$.
2. *If* $\Gamma, x : T \vdash t : T'$ *and* $\Gamma \vdash t' : T$, *then* $\Gamma \vdash [t'/x]t : [t'/x]T'$.

Proof. By induction on derivation.

Theorem 12. *If* $\Gamma \vdash t : T$ *and* $\Gamma \vdash t \to_{\beta,\mu} t'$ *and* $\Gamma \vdash \mathsf{wf}$, *then* $\Gamma \vdash t' : T$.

Proof. We list one interesting case:

$$\frac{\Gamma \vdash t : \Pi x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash tt' : [t'/x]T_2}\ App$$

Suppose $\Gamma \vdash (\lambda x.t_1)t_2 \to_\beta [t_2/x]t_1$. We know that $\Gamma \vdash \lambda x.t_1 : \Pi x : T_1.T_2$ and $\Gamma \vdash t_2 : T_1$. By inversion on $\Gamma \vdash \lambda x.t_1 : \Pi x : T_1.T_2$, we know that there exist Δ, T_1', T_2' such that $\Gamma, \Delta, x : T_1' \vdash t_1 : T_2'$ and $([\Gamma, \Delta], \Pi x : T_1'.T_2') \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], \Pi x : T_1.T_2)$. By Theorem 11, we have $([\Gamma, \Delta], \phi(\Pi x : T_1'.T_2')) =_{o,\iota,\beta,\mu} ([\Gamma, \Delta], \Pi x : T_1.T_2)$. By Church-Rosser of $=_{o,\iota,\beta,\mu}$, we have $\Gamma, \Delta \vdash \phi T_1' =_{o,\beta,\mu} T_1$ and $\Gamma, \Delta \vdash \phi T_2' =_{o,\beta,\mu} T_2$. So by (1) of lemma 18, we have $\Gamma, \phi(\Delta), x : \phi T_1' \vdash t_1 : \phi T_2'$ with $\mathrm{dom}(\phi(\Delta)) \# (\mathrm{FV}(\phi T_1') \cup \mathrm{FV}(\phi T_2') \cup \mathrm{FV}(t_1))$. So $\Gamma, \phi(\Delta), x : T_1 \vdash t_1 : T_2$. Since $\Gamma \vdash t_2 : T_1$, by (2) of lemma 18, $\Gamma, \phi(\Delta) \vdash [t_2/x]t_1 : [t_2/x]T_2$. So we have $\Gamma \vdash [t_2/x]t_1 : [t_2/x]T_2$ (by vacuously generalized and instantiated all the variables in $\mathrm{dom}(\phi(\Delta))$).

4 $0 \neq 1$ in S

The proof of $0 \neq 1$ follows the same method as in Theorem 1, while the argument for the uninhabitation of $\bot$ needs the erasure theorem and the preservation theorem. Notice that in this section, by $a = b$, we mean $\Pi C : (\Pi x : A.*).C\ a \to C\ b$ with $a, b : A$.

Definition 20.

$\bot := \Pi A : *.\forall x : A.\forall y : A.x = y$.

Theorem 13. *There is no term t such that $\tilde{\mu}_c \vdash t : \bot$*

Proof. Suppose $\tilde{\mu}_c \vdash t : \bot$. By the erasure theorem (Theorem 5) in section 3.1, then we have $F(\tilde{\mu}_c) \vdash F(t) \equiv t : \Pi A : *.\Pi C : *.C \to C$ in $\mathbf{F}_\omega$ with let-bindings. $\Pi A : *.\Pi C : *.C \to C$ is the singleton type[9], which is inhabited by $\lambda z.z$. This means $t \to^*_{\beta,\mu} \lambda z.z$ (the term reductions of $\mathbf{F}_\omega$ with let-bindings are the same as $\mathbf{S}$) and $\tilde{\mu}_c \vdash \lambda z.z : \bot$ in $\mathbf{S}$ (by type preservation, Theorem 12). Then we would have $\tilde{\mu}_c, A : *, x : A, y : A, C : (\Pi z : A.*), z : C\ x \vdash z : C\ y$. We know this derivation is impossible since $C\ x \ncong C\ y$ by the confluence of $\cong$.

Theorem 14. $\tilde{\mu}_c \vdash 0 = 1 \to \bot$.

Proof. This proof follows the method in Theorem 1. Let $\Gamma = \tilde{\mu}_c, a : (\Pi B : (\Pi z : \mathsf{Nat}.*).B0 \to B1), A : *, x : A, y : A, C : (\Pi z : A.*), c : C\ x$. We want to construct a term of type $C\ y$. Let $F := \lambda n[:\mathsf{Nat}].n\ [\lambda p : \mathsf{Nat}.A]\ (\lambda q[:A].y)x$, note that $F : \mathsf{Nat} \to A$. We know that $F0 =_\beta x$ and $F1 =_\beta y$. So we can indeed convert the type of c from Cx to $C\ (F0)$. And then we instantiate the B in $\Pi B : (\Pi z : \mathsf{Nat}.*).B0 \to B1$ with $\lambda x[:\mathsf{Nat}].C\ (Fx)$. So we have $C\ (F0) \to C\ (F1)$ as the type of a. So $ac : C(F1)$, which means $ac : Cy$. So we just show how to inhabite $0 = 1 \to \bot$ in $\mathbf{S}$.

5 Conclusion

We propose System $\mathbf{S}$, which incorporates the self type construct together with implicit product and a restricted version of mutually recursion. Church-encoded datatypes and the corresponding induction principles are derivable within $\mathbf{S}$. With a change of notion of contradiction, we are able to show $0 \neq 1$ in both **CC** and $\mathbf{S}$. System $\mathbf{S}$ is consistent, since mutual recursiveness is only needed for the typing and the erasure of the recursive bindings becoming ordinary let-binding. We also demonstrate the process of proving the type preservation theorem, which is a hard problem for $\mathbf{S}$ due to the present of closures and the non-syntactic directed rules.

References

1. Ki Yung Ahn, Tim Sheard, Marcelo Fiore, and Andrew M Pitts. System fi. In *Typed Lambda Calculi and Applications*, pages 15–30. Springer, 2013.
2. Zena M. Ariola and Jan Willem Klop. Lambda calculus with explicit recursion. *Information and Computation*, 139(2):154 – 233, 1997.
3. Hendrik Pieter Barendregt. *The lambda calculus: Its syntax and semantics*, volume 103. North Holland, 1985.
4. Henk Barendregt. The impact of the lambda calculus in logic and computer science. *Bulletin of Symbolic Logic*, 3(2):181–215, 1997.
5. HP Barendregt. Lambda calculi with types, handbook of logic in computer science (vol. 2): background: computational structures, 1993.
6. Alonzo Church. *The Calculi of Lambda Conversion. (AM-6) (Annals of Mathematics Studies)*. Princeton University Press, Princeton, NJ, USA, 1985.
7. T. Coquand. Metamathematical investigations of a calculus of constructions. Technical Report RR-1088, INRIA, September 1989.

[9] Note that we are dealing with Curry-style $\mathbf{F}_\omega$.

8. H. B. Curry, J. R. Hindley, and J. P. Seldin. *Combinatory Logic, Volume II*. North-Holland, 1972.
9. Jean-Yves Girard. Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, 1972.
10. Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.
11. Thérèse Hardin. Confluence results for the pure strong categorical logic ccl. λ-calculi as subsystems of ccl. *Theor. Comput. Sci.*, 65(3):291–342, July 1989.
12. James Roger Hindley. *The Church-Rosser property and a result in combinatory logic*. PhD thesis, University of Newcastle upon Tyne, 1964.
13. George Kuan, David MacQueen, and Robert Bruce Findler. A rewriting semantics for type inference. In *Proceedings of the 16th European conference on Programming*, pages 426–440. Springer-Verlag, 2007.
14. Alexandre Miquel. *Le Calcul des Constructions implicite: syntaxe et sémantique*. PhD thesis, PhD thesis, Université Paris 7, 2001.
15. Christine Paulin-Mohring. Inductive definitions in the system coq rules and properties. In *Typed lambda calculi and applications*, pages 328–345. Springer, 1993.
16. Barry K. Rosen. Tree-manipulating systems and church-rosser theorems. *J. ACM*, 20(1):160–187, January 1973.
17. Aaron Stump, Garrin Kimmell, and Roba El Haj Omar. Type preservation as a confluence problem. In *RTA*, pages 345–360, 2011.
18. B. Werner. A Normalization Proof for an Impredicative Type System with Large Elimination over Integers. In B. Nordström, K. Petersson, and G. Plotkin, editors, *International Workshop on Types for Proofs and Programs (TYPES)*, pages 341–357, 1992.

A Coq Code

The following code formalizes the proof of theorem 1 in Coq.

```
Definition eq :=
  fun (A : Prop)(a b : A) => forall C : A -> Prop , C a -> C b.

Definition false :=
  forall A : Prop , forall a : A , forall b : A , eq A a b .

Definition Nat := forall A : Prop , (A -> A) -> A -> A.

Definition zero : Nat := fun (A : Prop)(s : A -> A)(z : A) => z.

Definition succ : Nat -> Nat :=
  fun (n:Nat)(A : Prop)(s : A -> A)(z : A) => s (n A s z).

Definition one : Nat := succ zero.

Theorem zeroNeqOne : eq Nat zero one -> false.
unfold false.
unfold eq.
intros u A a b C.
exact (u (fun (n:Nat) => C (n A (fun(q:A) => b) a))).
Qed.
```

B Full Sepcifications of Reductions in S

Definition 21 (Metalevel Abrieviation).
Objects $o ::= t \mid T \mid \kappa$
Reduction Context $\mathcal{C} ::=$
$\bullet \mid \lambda x.\mathcal{C} \mid \mathcal{C}t' \mid t\mathcal{C} \mid \Pi X : \kappa.\mathcal{C} \mid \Pi x : T.\mathcal{C} \mid \Pi x : \mathcal{C}.T \mid \forall x : T.\mathcal{C} \mid \forall x : \mathcal{C}.T \mid \lambda X.\mathcal{C} \mid \iota x.\mathcal{C} \mid T\mathcal{C} \mid \mathcal{C}T \mid \Pi x : \mathcal{C}.\kappa \mid \Pi X : \mathcal{C}.\kappa \mid \Pi x : \kappa.\mathcal{C} \mid \Pi X : \kappa.\mathcal{C}$

Definition 22 (Closure Reductions).

$$\frac{\mu \in \Gamma}{\Gamma \vdash \mu t \to_o t} \quad \frac{\mu \in \Gamma}{\Gamma \vdash \mu T \to_o T} \quad \frac{\mu \in \Gamma}{\Gamma \vdash \mu\kappa \to_o \kappa} \quad \frac{\Gamma, \tilde{\mu} \vdash o \to_o o'}{\Gamma \vdash \mu o \to_o \mu o'} \quad \frac{\Gamma \vdash o \to_o o'}{\Gamma \vdash \mathcal{C}[o] \to_o \mathcal{C}[o']}$$

Note: Closure reducton is defined for type level only.

Definition 23 (Beta Reductions).

$$\frac{(x \mapsto t) \in \Gamma}{\Gamma \vdash x \to_\beta t} \qquad \frac{}{\Gamma \vdash (\lambda x.t)t' \to_\beta [t'/x]t} \quad \frac{(x_i \mapsto t_i) \in \mu}{\Gamma \vdash \mu x_i \to_\beta \mu t_i}$$

$$\frac{}{\Gamma \vdash (\lambda X.T)T' \to_\beta [T'/X]T} \quad \frac{(X \mapsto T) \in \Gamma}{\Gamma \vdash X \to_\beta T} \qquad \frac{}{\Gamma \vdash (\lambda x.T)t \to_\beta [t/x]T}$$

$$\frac{(X_i \mapsto T_i) \in \mu}{\Gamma \vdash \mu X_i \to_\beta \mu T_i} \qquad \frac{\Gamma, \tilde{\mu} \vdash o \to_\beta o'}{\Gamma \vdash \mu o \to_\beta \mu o'} \qquad \frac{\Gamma \vdash o \to_\beta o'}{\Gamma \vdash \mathcal{C}[o] \to_\beta \mathcal{C}[o']}$$

Definition 24 (Mu Reductions).

$$\frac{\mathrm{dom}(\mu)\#\mathrm{FV}(t)}{\Gamma \vdash \mu t \to_\mu t} \qquad \frac{}{\Gamma \vdash \mu(\lambda x.t) \to_\mu \lambda x.\mu t}$$

$$\frac{}{\Gamma \vdash \mu(t_1 t_2) \to_\mu (\mu t_1)(\mu t_2)} \qquad \frac{}{\Gamma \vdash \mu(T\ T') \to_\mu (\mu T)(\mu T')}$$

$$\frac{\mathrm{dom}(\mu)\#\mathrm{FV}(T)}{\Gamma \vdash \mu T \to_\mu T} \qquad \frac{}{\Gamma \vdash \mu(\iota x.T) \to_\mu \iota x.\mu T}$$

$$\frac{}{\Gamma \vdash \mu(\forall x : T_1.T_2) \to_\mu \forall x : \mu T_1.\mu T_2} \quad \frac{}{\Gamma \vdash \mu(\Pi X : \kappa.T) \to_\mu \Pi X : \mu\kappa.\mu T}$$

$$\frac{}{\Gamma \vdash \mu(T\ t) \to_\mu (\mu T)(\mu t)} \qquad \frac{}{\Gamma \vdash \mu(\lambda X.T) \to_\mu \lambda X.\mu T}$$

$$\frac{}{\Gamma \vdash \mu(\lambda x.T) \to_\mu \lambda x.\mu T} \qquad \frac{}{\Gamma \vdash \mu(\Pi x : T_1.T_2) \to_\mu \Pi x : \mu T_1.\mu T_2}$$

$$\frac{\mathrm{dom}(\mu)\#\mathrm{FV}(\kappa)}{\Gamma \vdash \mu\kappa \to_\mu \kappa} \qquad \frac{}{\Gamma \vdash \mu(\Pi x : T.\kappa) \to_\mu \Pi x : \mu T.\mu\kappa}$$

$$\frac{}{\Gamma \vdash \mu(\Pi X : \kappa'.\kappa) \to_\mu \Pi X : \mu\kappa'.\mu\kappa} \quad \frac{\Gamma, \tilde{\mu} \vdash o \to_\mu o'}{\Gamma \vdash \mu o \to_\mu \mu o'}$$

$$\frac{\Gamma \vdash o \to_\mu o'}{\Gamma \vdash \mathcal{C}[o] \to_\mu \mathcal{C}[o']}$$

C Full Specifications of F_ω with Let-binding

Definition 25 (Syntax).

$$\begin{array}{lll} \textit{Terms } t & ::= & x \mid \lambda x.t \mid tt' \mid \rho t \\ \textit{Types } T & ::= & X \mid \Pi X : \kappa.T \mid T_1 \to T_2 \mid \lambda X.T \mid T_1 T_2 \mid \rho T \\ \textit{Kinds } \kappa & ::= & * \mid \kappa' \to \kappa \\ \textit{Context } \Gamma & ::= & \cdot \mid \Gamma, x : T \mid \Gamma, X : \kappa \mid \Gamma, \tilde{\rho} \\ \textit{Let-binding } \rho & ::= & \{x_i \mapsto t_i\}_{i \in N} \cup \{X_i \mapsto T_i\}_{i \in M} \end{array}$$

Note that for every $x \mapsto t, X \mapsto T \in \rho$, we require $\mathrm{FV}(t) = \emptyset$ and $\mathrm{FV}(T) = \emptyset$.

Definition 26 (Metalevel Abrieviation).

$$\begin{array}{lll} \textit{Objects } o & ::= & t \mid T \\ \textit{Reduction Context } \mathcal{C} & ::= & \bullet \mid \lambda x.\mathcal{C} \mid \mathcal{C}t' \mid t\mathcal{C} \mid \Pi X : \kappa.\mathcal{C} \mid T \to \mathcal{C} \mid \mathcal{C} \to T \mid \lambda X.\mathcal{C} \mid T\mathcal{C} \mid \mathcal{C}T \end{array}$$

Definition 27 (Well-formed Context).

$$\frac{}{\cdot \vdash \mathsf{wf}} \quad \frac{\Gamma \vdash \mathsf{wf} \quad \Gamma \vdash T : *}{\Gamma, x : T \vdash \mathsf{wf}} \quad \frac{\Gamma \vdash \mathsf{wf} \quad \Gamma, \tilde{\rho} \vdash \mathsf{ok}}{\Gamma, \tilde{\rho} \vdash \mathsf{wf}}$$

Definition 28 (Kinding).

$$\frac{(X:\kappa)\in\Gamma}{\Gamma\vdash X:\kappa} \qquad \frac{\Gamma\vdash T_1:* \quad \Gamma\vdash T_2:*}{\Gamma\vdash T_1\to T_2:*} \qquad \frac{\Gamma,\tilde{\rho}\vdash T:\kappa \quad \Gamma,\tilde{\rho}\vdash \mathsf{ok}}{\Gamma\vdash \rho T:\kappa}$$

$$\frac{\Gamma,X:\kappa\vdash T:\kappa'}{\Gamma\vdash \lambda X.T:\kappa\to\kappa'} \quad \frac{\Gamma\vdash S:\kappa'\to\kappa \quad \Gamma\vdash T:\kappa'}{\Gamma\vdash ST:\kappa} \quad \frac{\Gamma,X:\kappa\vdash T:*}{\Gamma\vdash \Pi X:\kappa.T:*}$$

Definition 29 (Typing Rules).

$$\frac{(x:T)\in\Gamma}{\Gamma\vdash x:T}\ \mathit{Var} \qquad \frac{\Gamma\vdash t:T_1 \quad \Gamma\vdash T_1\cong T_2 \quad \Gamma\vdash T_2:*}{\Gamma\vdash t:T_2}\ \mathit{Conv}$$

$$\frac{\Gamma,x:T_1\vdash t:T_2 \quad \Gamma\vdash T_1:*}{\Gamma\vdash \lambda x.t:T_1\to T_2}\ \mathit{Func} \quad \frac{\Gamma\vdash t:T_1\to T_2 \quad \Gamma\vdash t':T_1}{\Gamma\vdash tt':T_2}\ \mathit{App}$$

$$\frac{\Gamma,\tilde{\rho}\vdash t:T \quad \Gamma,\tilde{\rho}\vdash \mathsf{ok}}{\Gamma\vdash \rho t:\rho T}\ \mathit{Mu} \qquad \frac{\Gamma,X:\kappa\vdash t:T}{\Gamma\vdash t:\Pi X:\kappa.T}\ \mathit{Poly}$$

$$\frac{\Gamma\vdash t:\Pi X:\kappa.T \quad \Gamma\vdash T':\kappa}{\Gamma\vdash t:[T'/X]T}\ \mathit{Inst}$$

Remarks :

- $\Gamma,\tilde{\rho}\vdash \mathsf{ok}$ stands for $\{\Gamma,\tilde{\rho}\vdash t_j:T_j\}_{(t_j:T_j)\in\tilde{\rho}}$ and $\{\Gamma,\tilde{\rho}\vdash T_j:\kappa_j\}_{(T_j:\kappa_j)\in\tilde{\rho}}$.
- $\cong$ is the reflexive transitive and symmetry closure of $\to_\beta\cup\to_o\cup\to_\rho$.
- If ρ is $\{x_i\mapsto t_i\}_{i\in N}\cup\{X_i\mapsto T_i\}_{i\in M}$, then $\tilde{\rho}$ is $\{(x_i:T_i)\mapsto t_i\}_{i\in N}\cup\{(X_i:\kappa_i)\mapsto T_i\}_{i\in M}$.

Definition 30 (Closure Reductions).

$$\frac{\rho\in\Gamma}{\Gamma\vdash \rho T\to_o T} \quad \frac{\Gamma,\tilde{\rho}\vdash o\to_o o'}{\Gamma\vdash \rho o\to_o \rho o'} \quad \frac{\Gamma\vdash o\to_o o'}{\Gamma\vdash \mathcal{C}[o]\to_o \mathcal{C}[o']}$$

Note: Closure reduction is defined at type level and since terms will not appear at type level in $\mathbf{F}_\omega$, we will not define $\to_o$ for term.

Definition 31 (Beta Reductions).

$$\frac{(x\mapsto t)\in\Gamma}{\Gamma\vdash x\to_\beta t} \quad \frac{}{\Gamma\vdash(\lambda x.t)t'\to_\beta [t'/x]t} \quad \frac{(x_i\mapsto t_i)\in\rho}{\Gamma\vdash \rho x_i\to_\beta \rho t_i} \quad \frac{}{\Gamma\vdash(\lambda X.T)T'\to_\beta [T'/X]T}$$

$$\frac{(X\mapsto T)\in\Gamma}{\Gamma\vdash X\to_\beta T} \quad \frac{(X_i\mapsto T_i)\in\rho}{\Gamma\vdash \rho X_i\to_\beta \rho T_i} \qquad \frac{\Gamma,\tilde{\rho}\vdash o\to_\beta o'}{\Gamma\vdash \rho o\to_\beta \rho o'} \quad \frac{\Gamma\vdash o\to_\beta o'}{\Gamma\vdash \mathcal{C}[o]\to_\beta \mathcal{C}[o']}$$

Definition 32 (Rho Reductions).

$$\frac{\mathrm{dom}(\rho)\#\mathrm{FV}(t)}{\Gamma \vdash \rho t \to_\rho t} \qquad \frac{}{\Gamma \vdash \rho(\lambda x.t) \to_\rho \lambda x.\rho t} \quad \frac{}{\Gamma \vdash \rho(t_1 t_2) \to_\rho (\rho t_1)(\rho t_2)}$$

$$\frac{}{\Gamma \vdash \rho(T\ T') \to_\rho (\rho T)(\rho T')} \quad \frac{\mathrm{dom}(\rho)\#\mathrm{FV}(T)}{\Gamma \vdash \rho T \to_\rho T} \qquad \frac{}{\Gamma \vdash \rho(\Pi X : \kappa.T) \to_\rho \Pi X : \rho\kappa.\rho T}$$

$$\frac{}{\Gamma \vdash \rho(\lambda X.T) \to_\rho \lambda X.\rho T} \quad \frac{\Gamma, \tilde{\rho} \vdash o \to_\rho o'}{\Gamma \vdash \rho o \to_\rho \rho o'} \qquad \frac{}{\Gamma \vdash \rho(T_1 \to T_2) \to_\rho \rho T_1 \to \rho T_2}$$

$$\frac{\Gamma \vdash o \to_\rho o'}{\Gamma \vdash \mathcal{C}[o] \to_\rho \mathcal{C}[o']}$$

D Proofs for Section 3.1

Lemma 19. *If $\kappa \in \mathcal{K}_s$, then $F(\kappa) \in \mathcal{K}_\omega$.*

Proof. We prove this by induction on the structure of κ.

Lemma 20.

1. *If $t \in \mathcal{M}_s$, then $F(t) \in \mathcal{M}_\omega$.*
2. *If $T \in \mathcal{T}_s$, then $F(T) \in \mathcal{T}_\omega$.*
3. *If $\mu \in \Psi$, then $F(\mu) \in \Theta$.*

Proof. We prove (1),(2) by induction on struture of t, T, we prove (3) by induction on the length of μ.

1. **Base Cases**: $t = x$. Obvious.
2. **Base Cases**: $T = X$. Obvious.
3. **Base Cases**: Suppose $\mu = x \mapsto t$, then by (1), we know $F(t) \in \mathcal{M}_\omega$. So $F(x \mapsto t) \in \Theta$. Suppose $\mu = X \mapsto T$, then by (2), we know $F(T) \in \mathcal{T}_\omega$. So $F(X \mapsto T) \in \Theta$.

1. **Step Cases**: $t = \mu t'$. Then $F(t) = F(\mu)F(t')$. By (3), we know that $F(\mu) \in \Theta$. By IH, we know $F(t') \in \mathcal{M}_\omega$. So $F(\mu)F(t') \in \mathcal{M}_\omega$.
2. **Step Cases**: $T = \mu T'$. Then $F(T) = F(\mu)F(T')$. By (3), we know that $F(\mu) \in \Theta$. By IH, we know $F(T') \in \mathcal{T}_\omega$. So $F(\mu)F(T') \in \mathcal{T}_\omega$.
3. **Step Cases**: Suppose $\mu = x \mapsto t, \mu'$, then by IH, we know $F(\mu') \in \Theta$. By (1), we know that $F(t) \in \mathcal{M}_\omega$. So $F(x \mapsto t, \mu') \in \Theta$. Suppose $\mu = X \mapsto T, \mu'$, then by (2), we know $F(T) \in \mathcal{T}_\omega$. By IH, we know $F(\mu') \in \Theta$, So $F(X \mapsto T, \mu') \in \Theta$.

Lemma 21. *$F(\kappa) \equiv F([t/x]\kappa)$, $F([T/X]\kappa) \equiv F(\kappa)$, $F([t/x]T) \equiv F(T)$ and $F([T'/X]T) \equiv [F(T')/X]F(T)$ and $F([t'/x]t) \equiv [F(t')/x]F(t)$.*

Proof. The first two equalities can be proved by induction on structure of κ(we will use local property). For the third and fourth ones, by induction on structure of T. Of course, one would worry about cases like $\Pi X : \kappa.T$ and μT, it turns out that the local property of μ will help us through the second case; for the first case, we rely on $F(\kappa)$ is a well-defined kind in $\mathbf{F}_\omega$.

Lemma 22. *If $\Gamma \vdash \kappa \cong \kappa'$, then $F(\kappa) \equiv F(\kappa')$.*

Proof. It suffices to show if $\Gamma \vdash \kappa \to_{o,\beta,\mu} \kappa'$, then $F(\kappa) \equiv F(\kappa')$. So we proceed by induction on derivation of $\Gamma \vdash \kappa \to_{o,\beta,\mu} \kappa'$.

Lemma 23. *If $\Gamma \vdash T \to_{o,\beta,\mu} T'$, then $F(\Gamma) \vdash F(T) \hookrightarrow_{o,\beta,\mu} F(T')$.*

Proof. By induction on derivation of $\Gamma \vdash T \to_{o,\beta,\mu} T'$. We use lemma 22, lemma 21 above.

Theorem 15.

1. *If $\Gamma \vdash T : \kappa$, then $F(\Gamma) \vdash F(T) : F(\kappa)$.*
2. *If $\Gamma \vdash t : T$, then $F(\Gamma) \vdash F(t) : F(T)$.*

Proof. We prove the conjuction of (1) and (2). By induction on derivation.

Base Cases:

$$\frac{(X : \kappa) \in \Gamma}{\Gamma \vdash X : \kappa} \quad \frac{(x : T) \in \Gamma}{\Gamma \vdash x : T}$$

Obvious.

Step Case:

$$\frac{\Gamma, X : \kappa \vdash T : * \quad \Gamma \vdash \kappa : \Box}{\Gamma \vdash \Pi X : \kappa.T : *}$$

By IH(1), we know $F(\Gamma), X : F(\kappa) \vdash F(T) : F(*)$. So $F(\Gamma) \vdash \Pi X : F(\kappa).F(T) : *$.

Step Case:

$$\frac{\Gamma, \tilde{\mu} \vdash T : \kappa \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma \vdash \mu T : \mu\kappa}$$

Recall that $\Gamma, \tilde{\mu} \vdash \mathsf{ok}$ stands for $\{\Gamma, \tilde{\mu} \vdash t_j : T_j\}_{(t_j:T_j)\in\tilde{\mu}}$ and $\{\Gamma, \tilde{\mu} \vdash T_j : \kappa_j\}_{(T_j:\kappa_j)\in\tilde{\mu}}$. By IH(1) and (2), we know $F(\Gamma), F(\tilde{\mu}) \vdash F(T) : F(\kappa)$ and $\{F(\Gamma), F(\tilde{\mu}) \vdash F(t_j) : F(T_j)\}_{(F(t_j):F(T_j))\in F(\tilde{\mu})}$ and $\{F(\Gamma), F(\tilde{\mu}) \vdash F(T_j) : F(\kappa_j)\}_{(F(T_j):F(\kappa_j))\in F(\tilde{\mu})}$. So we have $F(\Gamma) \vdash F(\mu T) \equiv F(\mu)F(T) : F(\kappa)$ by the kinding rule in $\mathbf{F}_\omega$. Note that $F(\mu) \equiv |F(\tilde{\mu})|$, where $|\cdot|$ deletes all the type annotations.

Step Case:

$$\frac{\Gamma, x : \iota x.T \vdash T : *}{\Gamma \vdash \iota x.T : *}$$

By IH(1), we know $F(\Gamma), x : F(T) \vdash F(T) : *$. Since $x \notin \mathrm{FV}(F(T))$, we have $F(\Gamma) \vdash F(T) : *$.

Step Case:

$$\frac{\Gamma, X : \kappa \vdash T : \kappa' \quad \Gamma \vdash \kappa : \Box}{\Gamma \vdash \lambda X.T : \Pi X : \kappa.\kappa'}$$

By IH(1), we know $F(\Gamma), X : F(\kappa) \vdash F(T) : F(\kappa')$. So $F(\Gamma) \vdash \lambda X.F(T) : F(\kappa) \to F(\kappa')$.

Step Case:

$$\frac{\Gamma, x : T' \vdash T : \kappa \quad \Gamma \vdash T' : *}{\Gamma \vdash \lambda x.T : \Pi x : T'.\kappa}$$

By IH(1), we have $F(\Gamma), x : F(T') \vdash F(T) : F(\kappa)$. Since $x \notin \mathrm{FV}(F(T))$, we know $F(\Gamma) \vdash F(T) : F(\kappa)$.

Step Case:

$$\frac{\Gamma \vdash S : \Pi x : T.\kappa \quad \Gamma \vdash t : T}{\Gamma \vdash St : [t/x]\kappa}$$

By IH(1), we have $F(\Gamma) \vdash F(S) : F(\kappa)$. Then we use the fact that $F(\kappa) \equiv F([t/x]\kappa)$.

Step Case:

$$\frac{\Gamma \vdash S : \Pi X : \kappa'.\kappa \quad \Gamma \vdash T : \kappa'}{\Gamma \vdash ST : [T/X]\kappa}$$

By IH(1), we have $F(\Gamma) \vdash F(S) : F(\kappa') \to F(\kappa)$ and $F(\Gamma) \vdash F(T) : F(\kappa')$. So $F(\Gamma) \vdash F(S)F(T) : F(\kappa)$. Then we use the fact that $F([T/X]\kappa) \equiv F(\kappa)$.

Step Case:

$$\frac{\Gamma, x : T_1 \vdash T_2 : * \quad \Gamma \vdash T_1 : *}{\Gamma \vdash \forall x : T_1.T_2 : *}$$

By IH(1), we know $F(\Gamma), x : F(T_1) \vdash F(T_2) : *$. Since $x \notin \mathrm{FV}(F(T_2))$, we have $F(\Gamma) \vdash F(T_2) : *$.

Step Case:

$$\frac{\Gamma \vdash t : T_1 \quad \Gamma \vdash T_1 \cong T_2 \quad \Gamma \vdash T_2 : *}{\Gamma \vdash t : T_2}\ \mathit{Conv}$$

By lemma 23 and IH(2).

Step Case:

$$\frac{\Gamma \vdash t : [t/x]T \quad \Gamma \vdash \iota x.T : *}{\Gamma \vdash t : \iota x.T}\ \mathit{SelfGen}$$

By IH(2) and lemma 21, we know $F(\Gamma) \vdash F(t) : F(T)$.

Step Case:

$$\frac{\Gamma \vdash t : \iota x.T}{\Gamma \vdash t : [t/x]T}\ \mathit{SelfInst}$$

By IH(2) and lemma 21, we know $F(\Gamma) \vdash F(t) : F(T)$.

Step Case:

$$\frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash T_1 : * \quad x \notin \mathrm{FV}(t)}{\Gamma \vdash t : \forall x : T_1.T_2}\ \mathit{Indx}$$

By IH(2), we know $F(\Gamma), x : F(T_1) \vdash F(t) : F(T_2)$. Since $x \notin \mathrm{FV}(t)$, thus $x \notin \mathrm{FV}(F(t))$, so we get $F(\Gamma) \vdash F(t) : F(T_2)$.

Step Case:

$$\frac{\Gamma \vdash t : \forall x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash t : [t'/x]T_2}\ \mathit{Dex}$$

By IH(2) and lemma 21, we know $F(\Gamma) \vdash F(t) : F(T_2)$.

Step Case:

$$\frac{\Gamma, \tilde{\mu} \vdash t : T \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma \vdash \mu t : \mu T}\ \mathit{Mu}$$

Recall that $\Gamma, \tilde{\mu} \vdash \mathsf{ok}$ stands for $\{\Gamma, \tilde{\mu} \vdash t_j : T_j\}_{(t_j:T_j)\in\tilde{\mu}}$ and $\{\Gamma, \tilde{\mu} \vdash T_j : \kappa_j\}_{(T_j:\kappa_j)\in\tilde{\mu}}$. By (1), IH(2), we know $F(\Gamma), F(\tilde{\mu}) \vdash F(t) : F(T)$, $\{F(\Gamma), F(\tilde{\mu}) \vdash F(t_j) : F(T_j)\}_{(F(t_j):F(T_j))\in F(\tilde{\mu})}$ and $\{F(\Gamma), F(\tilde{\mu}) \vdash F(T_j) : F(\kappa_j)\}_{(F(T_j):F(\kappa_j))\in F(\tilde{\mu})}$. So we have $F(\Gamma) \vdash F(\mu)F(t) \equiv F(\mu t) : F(\mu T) \equiv F(\mu)F(T)$ by the rule in $\mathbf{F}_\omega$. Note that $F(\mu) \equiv |F(\tilde{\mu})|$, where $|\cdot|$ deletes all the type annotations.

Step Case:

$$\frac{\Gamma, X : \kappa \vdash t : T \quad \Gamma \vdash \kappa : \Box}{\Gamma \vdash t : \Pi X : \kappa.T}\ \mathit{Poly}$$

By IH(2), we know $F(\Gamma), X : F(\kappa) \vdash F(t) : F(T)$, so $F(\Gamma) \vdash F(t) : \Pi X : F(\kappa).F(T)$.

Step Case:

$$\frac{\Gamma \vdash t : \Pi X : \kappa.T \quad \Gamma \vdash T' : \kappa}{\Gamma \vdash t : [T'/X]T}\ \mathit{Inst}$$

By IH(2) and (1), we know $F(\Gamma) \vdash F(t) : \Pi X : F(\kappa).F(T)$ and $F(\Gamma) \vdash F(T') : F(\kappa)$. So $F(\Gamma) \vdash F(t) : [F(T')/X]F(T) \equiv F([T'/X]T)$(lemma 21).

Step Case:

$$\frac{\Gamma, x : T_1 \vdash t : T_2 \quad \Gamma \vdash T_1 : *}{\Gamma \vdash \lambda x.t : \Pi x : T_1.T_2}\ \mathit{Func}$$

By IH(2) and (1), we know $F(\Gamma), x : F(T_1) \vdash F(t) : F(T_2)$ and $F(\Gamma) \vdash F(T_1) : *$. So $F(\Gamma) \vdash \lambda x.F(t) \equiv F(\lambda x.t) : F(T_1) \to F(T_2) \equiv F(\Pi x : T_1.T_2)$.

Step Case:

$$\frac{\Gamma \vdash t : \Pi x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash tt' : [t'/x]T_2} \; App$$

By IH(2), we have $F(\Gamma) \vdash F(t) : F(T_1) \to F(T_2)$ and $F(\Gamma) \vdash F(t') : F(T_1)$. So $F(\Gamma) \vdash F(tt') \equiv F(t)F(t') : F(T_2) \equiv F([t'/x]T_2)$(lemma 21).

E Proofs for Section 3.3

Let $\boldsymbol{\mu}t$ denote $\mu_1 ... \mu_n t$ where $n \geq 1$, we write $\dot{\boldsymbol{\mu}}t$ for $n \geq 0$.

Lemma 24. *$\nu(\boldsymbol{\mu}\boldsymbol{\mu}o) \equiv \nu(\boldsymbol{\mu}o)$, $\nu(\boldsymbol{\mu}([o_2/x]o_1)) \equiv \nu([\boldsymbol{\mu}o_2/x]\boldsymbol{\mu}o_1)$ and $\nu(\boldsymbol{\mu}([o_2/X]o_1)) \equiv \nu([\boldsymbol{\mu}o_2/X]\boldsymbol{\mu}o_1)$*

Lemma 25. *$\nu(\nu(o)) \equiv \nu(o)$, $\nu([\nu(o_1)/y]\nu(o_2)) \equiv \nu([o_1/y]o_2)$ and $\nu([\nu(o_1)/X]\nu(o_2)) \equiv \nu([o_1/X]o_2)$.*

In order to prove lemma 8, we prove a generalized version.

Lemma 26.

- *If $\Gamma, \dot{\boldsymbol{\mu}} \vdash t \to_\beta t'$, then $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}t) \to_{\beta\mu} \nu(\dot{\boldsymbol{\mu}}t')$.*
- *If $\Gamma, \dot{\boldsymbol{\mu}} \vdash T \to_\beta T'$, then $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}T) \to_{\beta\mu} \nu(\dot{\boldsymbol{\mu}}T')$.*
- *If $\Gamma, \dot{\boldsymbol{\mu}} \vdash \kappa \to_\beta \kappa'$, then $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}\kappa) \to_{\beta\mu} \nu(\dot{\boldsymbol{\mu}}\kappa')$.*

Proof. By induction on derivation of $\Gamma, \dot{\boldsymbol{\mu}} \vdash o \to_\beta o'$.
Base Case:

$$\frac{(x \mapsto t) \in \Gamma, \dot{\boldsymbol{\mu}}}{\Gamma, \dot{\boldsymbol{\mu}} \vdash x \to_\beta t}$$

If $x \mapsto t \in \dot{\boldsymbol{\mu}}$, then $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}x) \equiv \mu x \to_{\beta\mu} \nu(\mu t) \equiv \nu(\dot{\boldsymbol{\mu}}t)$. Techincally, the last equality need to be justified, we can justify that by locality of μ. If $x \mapsto t \in \Gamma$, then $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}x) \equiv x \to_{\beta\mu} \nu(t) \equiv \nu(\dot{\boldsymbol{\mu}}t)$.

Base Case:

$$\frac{(x_i \mapsto t_i) \in \mu}{\Gamma, \dot{\boldsymbol{\mu}} \vdash \mu x_i \to_\beta \mu t_i}$$

We have $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}\mu x_i) \equiv \mu x_i \to_{\beta\mu} \nu(\mu t_i) \equiv \nu(\dot{\boldsymbol{\mu}}\mu t_i)$.

Base Case:

$$\frac{}{\Gamma, \dot{\boldsymbol{\mu}} \vdash (\lambda x.t)t' \to_\beta [t'/x]t}$$

We have $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}((\lambda x.t)t')) \equiv (\lambda x.\nu(\dot{\boldsymbol{\mu}}t))\nu(\dot{\boldsymbol{\mu}}t') \to_{\beta\mu} \nu([\nu(\dot{\boldsymbol{\mu}}t)/x]\nu(\dot{\boldsymbol{\mu}}t')) \equiv \nu([\dot{\boldsymbol{\mu}}t/x]\dot{\boldsymbol{\mu}}t') \equiv \nu(\dot{\boldsymbol{\mu}}([t'/x]t))$. The last two equalities are by lemma 25, lemma 24.

Step Case:

$$\frac{\Gamma, \dot{\boldsymbol{\mu}} \vdash o \to_{\beta} o'}{\Gamma, \dot{\boldsymbol{\mu}} \vdash \mathcal{C}[o] \to_{\beta} \mathcal{C}[o']}$$

$\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}(\mathcal{C}[o])) \equiv \mathcal{C}[\nu(\dot{\boldsymbol{\mu}}o)] \overset{IH}{\to}_{\beta\mu} \mathcal{C}[\nu(\dot{\boldsymbol{\mu}}o')] \equiv \nu(\dot{\boldsymbol{\mu}}(\mathcal{C}[o']))$.

Step Case:

$$\frac{\Gamma, \dot{\boldsymbol{\mu}}, \tilde{\mu} \vdash o \to_{\beta} o'}{\Gamma, \dot{\boldsymbol{\mu}} \vdash \mu o \to_{\beta} \mu o'}$$

We want to show $\Gamma \vdash \nu(\dot{\boldsymbol{\mu}}\mu o) \to_{\beta\mu} \nu(\dot{\boldsymbol{\mu}}\mu o')$. This is directly by IH.

All the other cases are similar.

We will use Tait-Martin Löf's parallel reduction method to prove lemma 9. Recall the definition of the $\to_{\beta\mu}$:

Definition 33 (β Reduction on μ-normal Forms).

$$\frac{\Gamma \vdash n \to_{\beta} t}{\Gamma \vdash n \to_{\beta\mu} \nu(t)} \quad \frac{\Gamma \vdash N \to_{\beta} T}{\Gamma \vdash N \to_{\beta\mu} \nu(T)} \quad \frac{\Gamma \vdash K \to_{\beta} \kappa'}{\Gamma \vdash K \to_{\beta\mu} \nu(\kappa')}$$

Now let us define the notion of parallel reduction w.r.t. $\to_{\beta\mu}$.

Definition 34 (Parallel Reductions).

$$\frac{}{\Gamma \vdash n \Rightarrow_{\beta\mu} n} \qquad \frac{(x \mapsto t) \in \Gamma}{\Gamma \vdash x \Rightarrow_{\beta\mu} \nu(t)}$$

$$\frac{\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1' \quad \Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2'}{\Gamma \vdash (\lambda x.n_1)n_2 \Rightarrow_{\beta\mu} \nu([n_2'/x]n_1')} \qquad \frac{(x_i \mapsto t_i) \in \mu}{\Gamma \vdash \mu x_i \Rightarrow_{\beta\mu} \nu(\mu t_i)}$$

$$\frac{\Gamma \vdash n \Rightarrow_{\beta\mu} n'}{\Gamma \vdash \lambda x.n \Rightarrow_{\beta\mu} \lambda x.n'} \qquad \frac{\Gamma \vdash n \Rightarrow_{\beta\mu} n'' \quad \Gamma \vdash n' \Rightarrow_{\beta\mu} n'''}{\Gamma \vdash nn' \Rightarrow_{\beta\mu} n''n'''}$$

$$\frac{\Gamma \vdash N \Rightarrow_{\beta\mu} N'}{\Gamma \vdash \iota x.N \Rightarrow_{\beta\mu} \iota x.N'} \qquad \frac{\Gamma \vdash N' \Rightarrow_{\beta\mu} N''' \quad \Gamma \vdash N \Rightarrow_{\beta\mu} N''}{\Gamma \vdash \Pi x : N.N' \Rightarrow_{\beta\mu} \Pi x : N''.N'''}$$

$$\frac{\Gamma \vdash N \Rightarrow_{\beta\mu} N'}{\Gamma \vdash \lambda x.N \Rightarrow_{\beta\mu} \lambda x.N'} \qquad \frac{\Gamma \vdash N \Rightarrow_{\beta\mu} N'}{\Gamma \vdash \lambda X.N \Rightarrow_{\beta\mu} \lambda X.N'}$$

$$\frac{}{\Gamma \vdash N \Rightarrow_{\beta\mu} N} \qquad \frac{(X \mapsto T) \in \Gamma}{\Gamma \vdash X \Rightarrow_{\beta\mu} \nu(T)}$$

$$\frac{\Gamma \vdash N_1 \Rightarrow_{\beta\mu} N_1' \quad \Gamma \vdash N_2 \Rightarrow_{\beta\mu} N_2'}{\Gamma \vdash (\lambda X.N_1)N_2 \Rightarrow_{\beta\mu} \nu([N_2'/X]N_1')} \quad \frac{(X_i \mapsto T_i) \in \mu}{\Gamma \vdash \mu X_i \Rightarrow_{\beta\mu} \nu(\mu T_i)}$$

$$\frac{\Gamma \vdash N_1 \Rightarrow_{\beta\mu} N_1' \quad \Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2'}{\Gamma \vdash (\lambda x.N_1)n_2 \Rightarrow_{\beta\mu} \nu([n_2'/x]N_1')} \quad \frac{\Gamma \vdash N' \Rightarrow_{\beta\mu} N''' \quad \Gamma \vdash N \Rightarrow_{\beta\mu} N''}{\Gamma \vdash \forall x : N.N' \Rightarrow_{\beta\mu} \forall x : N''.N'''}$$

$$\frac{\Gamma \vdash N \Rightarrow_{\beta\mu} N' \quad \Gamma \vdash n \Rightarrow_{\beta\mu} n'}{\Gamma \vdash Nn \Rightarrow_{\beta\mu} N'n'} \qquad \frac{\Gamma \vdash N \Rightarrow_{\beta\mu} N'' \quad \Gamma \vdash N' \Rightarrow_{\beta\mu} N'''}{\Gamma \vdash NN' \Rightarrow_{\beta\mu} N''N'''}$$

$$\frac{\Gamma \vdash N' \Rightarrow_{\beta\mu} N''' \quad \Gamma \vdash K \Rightarrow_{\beta\mu} K'}{\Gamma \vdash \Pi x : K.N' \Rightarrow_{\beta\mu} \Pi x : K'.N'''} \quad \frac{}{\Gamma \vdash K \Rightarrow_{\beta\mu} K}$$

$$\frac{\Gamma \vdash N \Rightarrow_{\beta\mu} N' \quad \Gamma \vdash K \Rightarrow_{\beta\mu} K'}{\Gamma \vdash \Pi x : N.K \Rightarrow_{\beta\mu} \Pi x : N'.K'} \quad \frac{\Gamma \vdash K \Rightarrow_{\beta\mu} K'' \quad \Gamma \vdash K' \Rightarrow_{\beta\mu} K'''}{\Gamma \vdash \Pi X : K.K' \Rightarrow_{\beta\mu} \Pi X : K''.K'''}$$

Lemma 27. $\rightarrow_{\beta\mu} \subseteq \Rightarrow_{\beta\mu} \subseteq \rightarrow_{\beta\mu}^*$.

Lemma 28. *If* $\Gamma \vdash o_2 \Rightarrow_{\beta\mu} o_2'$, *then* $\Gamma \vdash \nu([o_2/x]o_1) \Rightarrow_{\beta\mu} \nu([o_2'/x]o_1)$ *and* $\Gamma \vdash \nu([o_2/X]o_1) \Rightarrow_{\beta\mu} \nu([o_2'/X]o_1)$.

Proof. By induction on the structure of o_1.

Base Cases: $o_1 = x, \mu x_i, X, \mu X_i, *$. Obvious.

Step Case: $o_1 = \lambda y.n$. We have $\Gamma \vdash \nu(\lambda y.[o_2/x]n) \equiv \lambda y.\nu([o_2/x]n) \overset{IH}{\Rightarrow}_{\beta\mu} \lambda y.\nu([o_2'/x]n) \equiv \nu(\lambda y.[o_2'/x]n)$.

Step Case: $o_1 = nn'$. We have $\Gamma \vdash \nu([o_2/x]n[o_2/x]n') \equiv \nu([o_2/x]n)\nu([o_2/x]n') \overset{IH}{\Rightarrow}_{\beta\mu} \nu([o_2'/x]n)\nu([o_2'/x]n') \equiv \nu([o_2'/x]n[o_2'/x]n)$.

The other cases are similar.

Lemma 29. *If* $\Gamma \vdash o_1 \Rightarrow_{\beta\mu} o_1'$ *and* $\Gamma \vdash o_2 \Rightarrow_{\beta\mu} o_2'$, *then* $\Gamma \vdash \nu([o_2/y]o_1) \Rightarrow_{\beta\mu} \nu([o_2'/y]o_1')$ *and* $\Gamma \vdash \nu([o_2/Y]o_1) \Rightarrow_{\beta\mu} \nu([o_2'/Y]o_1')$.

Proof. We prove this by induction on the derivation of $\Gamma \vdash o_1 \Rightarrow_{\beta\mu} o_1'$.

Base Case:

$$\frac{}{\Gamma \vdash n \Rightarrow_{\beta\mu} n}$$

$$\frac{}{\Gamma \vdash N \Rightarrow_{\beta\mu} N}$$

$$\frac{}{\Gamma \vdash K \Rightarrow_{\beta\mu} K}$$

By lemma 28.

Base Case:

$$\frac{x_i \mapsto t_i \in \mu}{\Gamma \vdash \mu x_i \Rightarrow_{\beta\mu} \nu(\mu t_i)}$$

Since $y \notin \mathrm{FV}(\mu x_i)$ and μ is local, we have $\nu([n_2/y]\mu x_i) \equiv \nu(\mu x_i)$ and $\nu(\mu x_1) \equiv \mu x_i \Rightarrow_{\beta\mu} \nu(\mu t_i) \equiv \nu(\nu(\mu t_i))$(lemma 25).

Base Case:

$$\frac{(x \mapsto t) \in \Gamma}{\Gamma \vdash x \Rightarrow_{\beta\mu} \nu(t)}$$

In this case, x is unsubstitutable. So this situation will not arise.

Step Case:

$$\frac{\Gamma \vdash n_a \Rightarrow_{\beta\mu} n_a' \quad \Gamma \vdash n_b \Rightarrow_{\beta\mu} n_b'}{\Gamma \vdash (\lambda x.n_a)n_b \Rightarrow_{\beta\mu} \nu([n_a'/x]n_b')}$$

We have $\Gamma \vdash \nu((\lambda x.[n_2/y]n_a)[n_2/y]n_b) \equiv (\lambda x.\nu([n_2/y]n_a))\nu([n_2/y]n_b)$
$\overset{IH}{\Rightarrow}_{\beta\mu} \nu([\nu([n_2'/y]n_b')/x]\nu([n_2'/y]n_a')) \equiv \nu([n_2'/y]([n_b'/x]n_a'))$. The last equality is by lemma 25. Here we first apply induction hypothesis to reduce, then apply $\Rightarrow_{\beta\mu}$.

Step Case:

$$\frac{\Gamma \vdash n \Rightarrow_{\beta\mu} n'}{\Gamma \vdash \lambda x.n \Rightarrow_{\beta\mu} \lambda x.n'}$$

We have $\Gamma \vdash \nu(\lambda x.[n_2/y]n) \equiv \lambda x.\nu([n_2/y]n) \overset{IH}{\Rightarrow}_{\beta\mu} \lambda x.\nu([n_2'/y]n') \equiv \nu(\lambda x.[n_2'/y]n')$

Step Case:

$$\frac{\Gamma \vdash n_a \Rightarrow_{\beta\mu} n_a' \quad \Gamma \vdash n_b \Rightarrow_{\beta\mu} n_b'}{\Gamma \vdash n_a n_b \Rightarrow_{\beta\mu} n_a' n_b'}$$

We have $\Gamma \vdash \nu([n_2/y]n_a[n_2/y]n_b) \equiv \nu([n_2/y]n_a)\nu([n_2/y]n_b)$
$\overset{IH}{\Rightarrow}_{\beta\mu} \nu([n_2'/y]n_a')\nu([n_2'/y]n_b') \equiv \nu([n_2'/y](n_a'n_b'))$.

The other cases are similar as above.

Lemma 30 (Diamond Property). *If $\Gamma \vdash o \Rightarrow_{\beta\mu} o'$ and $\Gamma \vdash o \Rightarrow_{\beta\mu} o''$, then there exists o''' such that $\Gamma \vdash o'' \Rightarrow_{\beta\mu} o'''$ and $\Gamma \vdash o' \Rightarrow_{\beta\mu} o'''$.*

Proof. By induction on the derivation of $\Gamma \vdash o \Rightarrow_{\beta\mu} o'$.
Base Case:

$$\frac{}{\Gamma \vdash n \Rightarrow_{\beta\mu} n}$$

Obvious.

Base Case:

$$\frac{(x \mapsto t) \in \Gamma}{\Gamma \vdash x \Rightarrow_{\beta\mu} \nu(t)}$$

Obvious.

Base Case:

$$\frac{}{\Gamma \vdash \mu x_i \Rightarrow_{\beta\mu} \nu(\mu t_i)}$$

Obvious.

Step Case:

$$\frac{\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1' \quad \Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2'}{\Gamma \vdash (\lambda x.n_1)n_2 \Rightarrow_{\beta\mu} \nu([n_2'/x]n_1')}$$

Suppose $\Gamma \vdash (\lambda x.n_1)n_2 \Rightarrow_{\beta\mu} (\lambda x.n_1'')n_2''$, where $\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1''$ and $\Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2''$. By IH, there exist n_1''', n_2''' such that $\Gamma \vdash n_1'' \Rightarrow_{\beta\mu} n_1'''$ and $\Gamma \vdash n_1' \Rightarrow_{\beta\mu} n_1'''$ and $\Gamma \vdash n_2' \Rightarrow_{\beta\mu} n_2'''$ and $\Gamma \vdash n_2' \Rightarrow_{\beta\mu} n_2'''$. By lemma 29, $\Gamma \vdash \nu([n_1'/x]n_2') \Rightarrow_{\beta\mu} \nu([n_1'''/x]n_2''')$, also $\Gamma \vdash (\lambda x.n_1'')n_2'' \Rightarrow_{\beta\mu} \nu([n_1'''/x]n_2''')$.

Suppose $\Gamma \vdash (\lambda x.n_1)n_2 \Rightarrow_{\beta\mu} \nu([n_2''/x]n_1'')$, where $\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1''$ and $\Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2''$. By IH, there exist n_1''', n_2''' such that $\Gamma \vdash n_1'' \Rightarrow_{\beta\mu} n_1'''$ and $\Gamma \vdash n_1' \Rightarrow_{\beta\mu} n_1'''$ and $\Gamma \vdash n_2' \Rightarrow_{\beta\mu} n_2'''$ and $\Gamma \vdash n_2' \Rightarrow_{\beta\mu} n_2'''$. By lemma 29, $\Gamma \vdash \nu([n_1'/x]n_2') \Rightarrow_{\beta\mu} \nu([n_1'''/x]n_2''')$ and $\Gamma \vdash \nu([n_1''/x]n_2'') \Rightarrow_{\beta\mu} \nu([n_1'''/x]n_2''')$.

The other cases are either similar to the one above or easy.

By lemma 30 and lemma 27, we conclude the confluence of $\rightarrow_{\beta\mu}$.

Lemma 31. *$\rightarrow_\iota$ is confluent.*

Proof. This is obvious since $\rightarrow_\iota$ is deterministic.

Lemma 32. *If $\Gamma \vdash o \to_{\beta,\mu} o'$, then $\Gamma \vdash [o_1/x]o \to_{\beta,\mu} [o_1/x]o'$ and $\Gamma \vdash [o_1/X]o \to_{\beta,\mu} [o_1/X]o'$ for any o_1.*

Proof. Obvious.

Lemma 33. *$\to_{\beta,\mu}$ commutes with $\to_\iota$. i.e. if $\Gamma \vdash T_1 \to_{\beta,\mu} T_2$ and $\Gamma \vdash T_1 \to_\iota T_3$, then there exists T_4 such that $\Gamma \vdash T_2 \to_\iota T_4$ and $\Gamma \vdash T_3 \to_{\beta,\mu} T_4$.*

Proof. Since $\Gamma \vdash T_1 \to_\iota T_3$, we know that $T_1 \equiv \iota x.T'$ and $T_3 \equiv [t/x]T'$. We also have $\Gamma \vdash T_1 \equiv \iota x.T' \to_{\beta,\mu} T_2$. By inversion, we know that $T_2 \equiv \iota x.T''$ with $\Gamma \vdash T' \to_{\beta,\mu} T''$. By lemma 32, we know that $\Gamma \vdash [t/x]T' \to_{\beta,\mu} [t/x]T''$. Thus $T_4 \equiv [t/x]T''$ and $\Gamma \vdash \iota x.T'' \to_\iota [t/x]T''$.

Lemma 34. *If $\Gamma \vdash o_1 \to_o o_2$, then $\Gamma \vdash [o/x]o_1 \to_o [o/x]o_2$ and $\Gamma \vdash [o/X]o_1 \to_o [o/X]o_2$.*

Proof. By induction on derivaton.

Lemma 35. *If $\Gamma \vdash o_1 \to_o o_2$, then $\Gamma \vdash [o_1/x]o \hookrightarrow_o [o_2/x]o$ and $\Gamma \vdash [o_1/X]o \hookrightarrow_o [o_2/X]o$.*

Proof. By induction on the structure of o.

Lemma 36. *$\to_o$ has diamond property, thus is confluent.*

Proof. Straightforward induction.

Lemma 37. *$\to_o$ commutes with $\to_\iota$.*

Proof. Suppose $\Gamma \vdash \iota x.T' \to_\iota [t/x]T'$ and $\Gamma \vdash \iota x.T' \to_o \iota x.T''$ with $\Gamma \vdash T' \to_o T''$. Then by lemma 34, we have $\Gamma \vdash [t/x]T' \to_o [t/x]T''$. We also have $\Gamma \vdash \iota x.T'' \to_\iota [t/x]T''$.

Lemma 38. *$\to_o$ weak commutes with $\to_\beta$.*

Proof. By induction on $\to_o$.

Case: $\Gamma \vdash \mu t \to_o t$, where $\mu \in \Gamma$.

If $\Gamma \vdash \mu x_i \to_\beta \mu t_i$, where $x_i \mapsto t_i \in \mu$, then $\Gamma \vdash \mu x_i \to_o x_i$. So we have $\Gamma \vdash \mu t_i \to_o t_i$ and $\Gamma \vdash x_i \to_\beta t_i$ since $\mu \in \Gamma$.

If $\Gamma \vdash \mu t \to_\beta \mu t'$, with $\Gamma \vdash t \to_\beta t'$. So we have $\Gamma \vdash t \to_\beta t'$ and $\Gamma \vdash \mu t' \to_o t'$.

Case: $\Gamma \vdash (\lambda x.t_1)t_2 \to_o (\lambda x.t_1')t_2$, where $\Gamma \vdash t_1 \to_o t_1'$.

Suppose $\Gamma \vdash (\lambda x.t_1)t_2 \to_\beta [t_2/x]t_1$. By lemma 34, we know that $\Gamma \vdash [t_2/x]t_1 \to_o [t_2/x]t_1'$. And we also have $\Gamma \vdash (\lambda x.t_1')t_2 \to_\beta [t_2/x]t_1'$.

Case: $\Gamma \vdash (\lambda x.t_1)t_2 \to_o (\lambda x.t_1)t_2'$, where $\Gamma \vdash t_2 \to_o t_2'$.

Suppose $\Gamma \vdash (\lambda x.t_1)t_2 \to_\beta [t_2/x]t_1$. By lemma 35, we know that $\Gamma \vdash [t_2/x]t_1 \hookrightarrow_o [t_2'/x]t_1$. And we also have $\Gamma \vdash (\lambda x.t_1)t_2' \to_\beta [t_2'/x]t_1$.

The other cases are similar.

Lemma 39. *$\to_o$ weak commutes with $\to_\mu$. i.e. if $\Gamma \vdash o \to_o o'$ and $\Gamma \vdash o \to_\mu o''$, then there exists a o_1 such that $\Gamma \vdash o'' \to_o^* o_1$ and $\Gamma \vdash o' \hookrightarrow_\mu o_1$.*

Proof. By induction on $\Gamma \vdash o \to_o o'$.

Case: $\Gamma \vdash \mu t \to_o t$, where $\mu \in \Gamma$.

Suppose $\Gamma \vdash \mu t \to_\mu t$ with $\mathrm{dom}(\mu)\#\mathrm{FV}(t)$. This case is obvious.

Suppose $t \equiv \lambda x.t_2$ and $\Gamma \vdash \mu(\lambda x.t_2) \to_\mu \lambda x.\mu t_2$. Then $\Gamma \vdash \lambda x.t_2 \hookrightarrow_\mu \lambda x.t_2$ and $\Gamma \vdash \lambda x.\mu t_2 \to_o \lambda x.t_2$.

Suppose $t \equiv t_2 t_3$ and $\Gamma \vdash \mu(t_2 t_3) \to_\mu (\mu t_2)(\mu t_3)$. Then $\Gamma \vdash t_2 t_3 \hookrightarrow_\mu t_2 t_3$ and $\Gamma \vdash (\mu t_2)(\mu t_3) \to_o^* t_2 t_3$.

The other cases are similar.

F Proofs for Section 3.5

Lemma 40. *Let $([\Gamma, \Delta], T_1) \to_{o,\iota,\beta,\mu,i,g,I,G}{}^* ([\Gamma], T_2)$. If $\Gamma, \Delta \vdash t : T_1$ with $\mathrm{dom}(\Delta)\#\mathrm{FV}(t)$, then $\Gamma \vdash t : T_2$.*

Note: We write $\overset{t}{\to}_{\beta,\mu,\iota,o,i,g,I,G}$ to mean the same thing as $\to^*_{\beta,\mu,\iota,o,i,g,I,G}$ with an emphasis on the subject t.

Lemma 41. *If $([\Gamma], T_1) \overset{t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma'], T_2)$ and $\Gamma \vdash t =_{\beta,\mu} t'$, then $([\Gamma], T_1) \overset{t'}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma'], T_2)$.*

Proof. By induction on the length of $([\Gamma], T_1) \overset{t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma], T_2)$. Note that this lemma is **not** subject expansion, do not get confused.

Lemma 42. *If $\Gamma, \tilde{\mu}, y : T' \vdash t : T$, then $\Gamma, y : \mu T', \tilde{\mu} \vdash t : T$.*

Proof. By induction on the derivation of $\Gamma, \tilde{\mu}, y : T' \vdash t : T$.

Note: If $\Delta = x : T, ..., X : \kappa$, then $\mu\Delta := x : \mu T, ..., X : \mu\kappa$.

Lemma 43. *If $([\Gamma, \tilde{\mu}, \Delta], T) \overset{t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \tilde{\mu}, \Delta'], T')$ for some Δ, Δ' and $\Gamma, \mu\Delta, \tilde{\mu} \vdash \mathsf{ok}$, then $([\Gamma, \mu\Delta], \mu T) \overset{\mu t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \mu\Delta'], \mu T')$.*

Proof. By induction on the length of $([\Gamma, \tilde{\mu}, \Delta], T) \overset{t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \tilde{\mu}, \Delta'], T')$. We list a few cases.

Case: $([\Gamma, \tilde{\mu}, \Delta], T) =_{\beta,\mu,o} ([\Gamma, \tilde{\mu}, \Delta], T')$.

Use lemma 42, we have $([\Gamma, \mu\Delta], \mu T) =_{\beta,\mu,o} ([\Gamma, \mu\Delta'], \mu T')$.

Case: $([\Gamma, \tilde{\mu}, \Delta], \iota x.T) =_\iota ([\Gamma, \tilde{\mu}, \Delta], [t/x]T)$.

We know $([\Gamma, \mu\Delta], \mu\iota x.T) =_\mu ([\Gamma, \mu\Delta], \iota x.\mu T) =_\iota ([\Gamma, \mu\Delta], [\mu t/x]\mu T) =_\mu ([\Gamma, \mu\Delta], \mu[t/x]T)$.

Case: $([\Gamma, \tilde{\mu}, \Delta], \Delta X : \kappa.T) \to_i ([\Gamma, \tilde{\mu}, \Delta], [T'/X]T)$ with $\Gamma, \tilde{\mu}, \Delta \vdash T' : \kappa$.

We know $([\Gamma, \mu\Delta], \mu(\Delta X : \kappa.T)) =_\mu ([\Gamma, \mu\Delta], \Delta X : \mu\kappa.\mu T) \to_i ([\Gamma, \mu\Delta], [\mu T'/X]\mu T) =_\mu ([\Gamma, \mu\Delta], \mu([T'/X]T))$ with $\Gamma, \mu\Delta \vdash \mu T' : \mu\kappa$.

Case: $([\Gamma, \tilde{\mu}, \Delta, X : \kappa], T) \to_g ([\Gamma, \tilde{\mu}, \Delta], \Delta X : \kappa.T)$ with $\Gamma, \tilde{\mu}, \Delta \vdash \kappa : \Box$.

We know $([\Gamma, \mu\Delta, X : \mu\kappa], \mu T) \to_g ([\Gamma, \mu\Delta], \Delta X : \mu\kappa.\mu T) =_\mu ([\Gamma, \mu\Delta], \mu(\Delta X : \kappa.T))$ with $\Gamma, \mu\Delta \vdash \mu\kappa : \Box$.

Case: $([\Gamma, \tilde{\mu}, \Delta], \forall x : T'.T) \to_I ([\Gamma, \tilde{\mu}, \Delta], [t/x]T)$ with $\Gamma, \tilde{\mu}, \Delta \vdash t : T'$.

We know $([\Gamma, \mu\Delta], \mu(\forall x : T'.T)) =_\mu ([\Gamma, \mu\Delta], \forall x : \mu T'.\mu T) \to_I ([\Gamma, \mu\Delta], [\mu t/x]\mu T) =_\mu ([\Gamma, \mu\Delta], \mu[t/x]T)$ with $\Gamma, \mu\Delta \vdash \mu t : \mu T'$.

Case: $([\Gamma, \tilde{\mu}, \Delta, x : T'], T) \to_G ([\Gamma, \tilde{\mu}, \Delta], \forall x : T'.T)$ with $\Gamma, \tilde{\mu}, \Delta \vdash T' : *$.

We know $([\Gamma, \mu\Delta, x : \mu T'], \mu T) \to_G ([\Gamma, \mu\Delta], \forall x : \mu T'.\mu T) =_\mu ([\Gamma, \mu\Delta], \mu(\forall x : T'.T))$ with $\Gamma, \mu\Delta \vdash \mu T' : *$.

Lemma 44 (Inversion I). *If $\Gamma \vdash x : T$, then exist Δ, T_1 such that $([\Gamma, \Delta], T_1) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$ and $(x : T_1) \in \Gamma$.*

Lemma 45 (Inversion II). *If $\Gamma \vdash t_1 t_2 : T$, then exist Δ, T_1, T_2 such that $\Gamma, \Delta \vdash t_1 : \Pi x : T_1.T_2$ and $\Gamma, \Delta \vdash t_2 : T_1$ and $([\Gamma, \Delta], [t_2/x]T_2) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$.*

Lemma 46 (Inversion III). *If $\Gamma \vdash \lambda x.t : T$, then exist Δ, T_1, T_2 such that $\Gamma, \Delta, x : T_1 \vdash t : T_2$ and $([\Gamma, \Delta], \Pi x : T_1.T_2) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$.*

Lemma 47 (Inversion IV). *If $\Gamma \vdash \mu t : T$, then exist Δ, T_1 such that $\Gamma, \Delta, \tilde{\mu} \vdash t : T_1$ and $([\Gamma, \Delta], \mu T_1) \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], T)$.*

Lemma 48 (Substitution).

1. *If $\Gamma \vdash t : T$, then for any mixed substitution ϕ with $\mathrm{dom}(\phi) \# \mathrm{FV}(t)$, $\phi\Gamma \vdash t : \phi T$.*
2. *If $\Gamma, x : T \vdash t : T'$ and $\Gamma \vdash t' : T$, then $\Gamma \vdash [t'/x]t : [t'/x]T'$.*

Proof. By induction on derivation.

Theorem 16. *If $\Gamma \vdash t : T$ and $\Gamma \vdash t \to_{\beta,\mu} t'$ and $\Gamma \vdash \mathsf{wf}$, then $\Gamma \vdash t' : T$.*

Proof. By induction on derivation of $\Gamma \vdash t : T$. We list a few interesting cases.

Case:

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

If $\Gamma \vdash x \to_\beta t'$, this means $(x : T) \mapsto t' \in \Gamma$ and $\Gamma \vdash t' : T$ since $\Gamma \vdash \mathsf{wf}$.

Case:

$$\frac{\Gamma \vdash t : \Pi x : T_1.T_2 \quad \Gamma \vdash t' : T_1}{\Gamma \vdash tt' : [t'/x]T_2}\ App$$

Suppose $\Gamma \vdash (\lambda x.t_1)t_2 \to_\beta [t_2/x]t_1$. We know that $\Gamma \vdash \lambda x.t_1 : \Pi x : T_1.T_2$ and $\Gamma \vdash t_2 : T_1$. By inversion on $\Gamma \vdash \lambda x.t_1 : \Pi x : T_1.T_2$, we know that there exist Δ, T_1', T_2' such that $\Gamma, \Delta, x : T_1' \vdash t_1 : T_2'$ and $([\Gamma, \Delta], \Pi x : T_1'.T_2') \to^*_{o,\iota,\beta,\mu,i,g,I,G} ([\Gamma], \Pi x : T_1.T_2)$. By Theorem 11, we have $([\Gamma, \Delta], \phi(\Pi x : T_1'.T_2')) =_{o,\iota,\beta,\mu} ([\Gamma, \Delta], \Pi x : T_1.T_2)$. By Church-Rosser of $=_{o,\iota,\beta,\mu}$, we have $\Gamma, \Delta \vdash \phi T_1' =_{o,\beta,\mu} T_1$ and $\Gamma, \Delta \vdash \phi T_2' =_{o,\beta,\mu} T_2$. So by (1) of lemma 48, we have $\Gamma, \phi(\Delta), x : \phi T_1' \vdash t_1 : \phi T_2'$ with $\mathrm{dom}(\phi(\Delta)) \# (\mathrm{FV}(\phi T_1') \cup \mathrm{FV}(\phi T_2') \cup \mathrm{FV}(t_1))$. So $\Gamma, \phi(\Delta), x : T_1 \vdash t_1 : T_2$. Since $\Gamma \vdash t_2 : T_1$, by (2) of lemma 48, $\Gamma, \phi(\Delta) \vdash [t_2/x]t_1 : [t_2/x]T_2$. So we have $\Gamma \vdash [t_2/x]t_1 : [t_2/x]T_2$.

Case:

$$\frac{\Gamma, \tilde{\mu} \vdash t : T \quad \Gamma, \tilde{\mu} \vdash \mathsf{ok}}{\Gamma \vdash \mu t : \mu T}$$

Suppose $\Gamma \vdash \mu x_j \to_\beta \mu t_j$, where $x_j \mapsto t_j \in \mu$. We have $\Gamma, \tilde{\mu} \vdash x_j : T$. By inversion, $\Gamma, \tilde{\mu}, \Delta \vdash x_j : T_j$ and $([\Gamma, \tilde{\mu}, \Delta], T_j) \overset{x_j}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \tilde{\mu}], T)$. Since $\Gamma, \tilde{\mu}, \Delta \vdash x_j =_\beta t_j$ and by lemma 41, we get $([\Gamma, \tilde{\mu}, \Delta], T_j) \overset{t_j}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \tilde{\mu}], T)$. Since $\Gamma, \tilde{\mu}, \Delta \vdash t_j : T_j$, by lemma 40, $\Gamma, \tilde{\mu} \vdash t_j : T$. Thus we have $\Gamma \vdash \mu t_j : \mu T$.

Suppose $\Gamma \vdash \mu t \to_\mu t$, where $\mathsf{FV}(t) \# \mathrm{dom}(\mu)$. We have $\Gamma, \tilde{\mu} \vdash t : T$ and $(\mathrm{FV}(t) \cup \mathrm{FV}(T)) \# \mathrm{dom}(\mu)$. So we have $([\Gamma], \mu T) =_\mu ([\Gamma], T)$. We also know that $\Gamma \vdash t : T$, so $\Gamma \vdash t : \mu T$(lemma 40).

Suppose $\Gamma \vdash \mu\lambda x.t \to_\mu \lambda x.\mu t$. We have $\Gamma, \tilde{\mu} \vdash \lambda x.t : T$ and $\Gamma, \tilde{\mu}, \Delta, x : T_1 \vdash t : T_2$ and $([\Gamma, \tilde{\mu}, \Delta], \Pi x : T_1.T_2) \overset{\lambda x.t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \tilde{\mu}], T)$(by inversion). Thus we have $\Gamma, \mu\Delta, x : \mu T_1 \vdash \mu t : \mu T_2$(lemma 42) and $([\Gamma, \mu\Delta], \mu(\Pi x : T_1.T_2)) \overset{\mu\lambda x.t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma], \mu T)$(lemma 43). By lemma 41, $([\Gamma, \mu\Delta], (\Pi x : \mu T_1.\mu T_2)) \overset{\lambda x.\mu t}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma], \mu T)$. Also, $\Gamma, \mu\Delta \vdash \lambda x.\mu t : \Pi x : (\mu T_1).(\mu T_2)$. So by lemma 40, $\Gamma \vdash \lambda x.\mu t : \mu T$.

Suppose $\Gamma \vdash \mu(t_1' t_2') \to_\mu (\mu t_1')(\mu t_2')$. We have $\Gamma, \tilde{\mu} \vdash t_1' t_2' : T$ and $\Gamma, \tilde{\mu}, \Delta \vdash t_1' : \Pi x : T_1.T_2$ and $\Gamma, \tilde{\mu}, \Delta \vdash t_2' : T_1$ and $([\Gamma, \tilde{\mu}, \Delta], [t_2'/x]T_2) \overset{t_1' t_2'}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma, \tilde{\mu}], T)$(by inversion). Thus we have $\Gamma, \mu\Delta \vdash \mu t_1' : \mu(\Pi x : T_1.T_2)$ and $\Gamma, \mu\Delta \vdash \mu t_2' : \mu T_1$ and $([\Gamma, \mu\Delta], \mu([t_2'/x]T_2)) \overset{\mu(t_1' t_2')}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma], \mu T)$(lemma 43). By lemma 41, we have $([\Gamma, \mu\Delta], \mu([t_2'/x]T_2)) \overset{(\mu t_1')(\mu t_2')}{\to}_{\beta,\mu,\iota,o,i,g,I,G} ([\Gamma], \mu T)$. So $\Gamma, \mu\Delta \vdash (\mu t_1')(\mu t_2') : [\mu t_2'/x]\mu T_2$ and then $\Gamma \vdash (\mu t_1')(\mu t_2') : \mu T$(lemma 40).