Dependently-Typed Programming with Scott Encoding

Peng Fu, Aaron Stump

Computer Science, The University of Iowa

Abstract

We introduce Selfstar, a Curry-style dependent type system featuring *self* type $\iota x.t$, together with mutually recursive definitions and $* : *$. We show how to obtain Scott-encoded datatypes and the corresponding elimination schemes with Selfstar. Examples such as numerals, vector are given to demonstrate the power of Selfstar as a dependently-typed programming language. Standard metatheorems such as type preservation are proved.

1. Introduction

Self type is originated from our previous work on $\mathbf{S}$ [5], since then self type has been studied in combination with different typing principles. In this paper, we study a system called Selfstar, which combines self type together with $* : *$ and mutually recursive definitions. In Selfstar, every type is inhabited, so Selfstar is inconsistent as a logic. The only logical feature in Selfstar is the Leibniz *convertability*, i.e. we define $t_1 =_A t_2$ to be $\Pi C : A \to *.C t_1 \to C t_2$. Note that we use "convertability" instead of "equality" to indicate one can not interpret $t_1 =_A t_2$ as a formula. If we know the inhabitant of $t_1 =_A t_2$ is normalized at the term $\lambda C.\lambda x.x$, then we can use $t_1 =_A t_2$ to cast the type $P t_1$ to $P t_2$ by applying the term $(\lambda C.\lambda x.x)P$ to the inhabitant of $P t_1$. Note that $(\lambda C.\lambda x.x)P \to_\beta \lambda x.x$, so the casting will not affect the inhabitant of $P t_1$.

Scott encoding (reported in [4]) does not suffer from the ineffeciency problem arised in Church encoding. For functional programming langauge, Scott encoding seems to be a better fit than Church encoding [9]. From the typing perspective, each Scott-encoded data contains its subdata, one would need recursive definition in order to define a type for Scott encoded data. Elimination schemes for the Scott-encoded data are derivable in Selfstar, this means programmer can write down programs that have types like $\Pi x : \mathsf{Nat}.\mathsf{add}\ x\ 0 =_{\mathsf{Nat}} x$, which increases the flexibility of type-level casting.

The main contributions of this paper are:

- We present Selfstar, which allows us to type Scott-encoded data and derive elimination schemes for Scott-encoded data. Selfstar simplifies the design of the functional programming language, since the primitive notion of inductive data and pattern matching is not needed in Selfstar.
- We prove type preservation and progress for Selfstar by applying the method we developed in the study of System $\mathbf{S}$.

1.1 Motivation

In Curry style system $\mathbf{T}$ [6] equipped with polymorphic and dependent type, one has a primitive notion of *recursor*, namely, $\mathsf{rec} : \Pi x : \mathsf{Nat}.\forall U.(\mathsf{Nat} \to U \to U) \to U \to U$ and two reductions rules: $\mathsf{rec}\ 0\ f\ v \to v$ and $\mathsf{rec}\ (\mathsf{S}n)\ f\ v \to f\ n\ (\mathsf{rec}\ n\ f\ v)$.

The recursor can be emulated with lambda terms. For example, $\mathsf{rec} := \lambda n.\lambda f.\lambda v.n\ f\ v$, with the notion of numeral $\bar{0} := \lambda s.\lambda z.z$ and $\bar{n} := \lambda s.\lambda z.s\ \overline{n-1}\ (\overline{n-1}\ s\ z)$. One can verify that the definition of rec in lambda calculus behaves the same as the one in system $\mathbf{T}$. With recursive definition, we can define $\mathsf{Nat} := \forall U.(\mathsf{Nat} \to U \to U) \to U \to U$. Note that the type of $\bar{n}$ is the same as the type of $\mathsf{rec}\ \bar{n}$.

So far the type of the recursor is *elementary*, i.e., not involving dependency. To make real use of dependent type, we ask if it is possible to obtain a type likes $\Pi x : \mathsf{Nat}.\forall U : \mathsf{Nat} \to *.(\Pi y : \mathsf{Nat}.U\ y \to U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ x$. Note that $\mathsf{S} := \lambda n.\lambda s.\lambda z.s\ n\ (n\ s\ z)$. We want to emphasis the underlying computational behavior of this type should be the same as rec, thus we want a typing relation $\mathsf{rec} : \forall U : \mathsf{Nat} \to *.\Pi x : \mathsf{Nat}.(\Pi y : \mathsf{Nat}.U\ y \to U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ x$. We also want the type of $\mathsf{rec}\ \bar{n}$, namely $(\Pi y : \mathsf{Nat}.U\ y \to U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ \bar{n}$ to be the same as the type of $\bar{n}$. So we want the following typing relation: $\bar{n} : (\Pi y : \mathsf{Nat}.U\ y \to U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ \bar{n}$ for any $\bar{n}$. We know the following self type mechanism:

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

So it is not surprising we define $\mathsf{Nat} := \iota x.(\Pi y : \mathsf{Nat}.U\ y \to U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ x$. With *selfGen*, *selfInst* and mutually recursive definition, one can verify that indeed, the type of $\bar{n}$ is the same as the type of $\mathsf{rec}\ \bar{n}$ and the type of rec is indeed $\forall U : \mathsf{Nat} \to *.\Pi x : \mathsf{Nat}.(\Pi y : \mathsf{Nat}.U\ y \to U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ x$. It is tempted to claim that rec represents the induction principle, but it is not for the following two reasons: 1. With mutually recursive definitions, the types can not be interpreted as formulas. 2. The dependent product Π is not exactly the first order quantifier $\forall$.

System $\mathbf{T}$ is closed to the functional programming language, but still a little far from a functional programmer's usual experience. In mordern functional programming language, one would want to write a plus 2 function in the following style:

```
data Nat = Zero
         | Succ Nat

plusTwo :: Nat -> Nat
plusTwo n = case n of
              Succ p -> Succ (plusTwo p)
            | Zero -> Succ (Succ Zero)
```

With Scott numerals, we can achieve the effects above. Assume Scott numerals and mutually recursive definition. We define:

```
Zero = lam s . lam z . z
Succ n =  lam s . lam z . s n
plusTwo n = case n
                lam p. Succ (plusTwo p)
                Succ (Succ Zero)
```

Of course, $\mathsf{case} := \lambda n.\lambda f.\lambda a.n\ f\ a$ and lam denotes the usual λ. One can see the differences between the two programs above are mostly superficial. Now let us first give a elementary version of type for case, which is $(\mathsf{Nat} \to U) \to U \to U$. The dependency version is $\Pi x : \mathsf{Nat}.(\Pi y : \mathsf{Nat}.U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ x$. And we define $\mathsf{Nat} := \iota x.(\Pi y : \mathsf{Nat}.U\ (\mathsf{S}y)) \to U\ \bar{0} \to U\ x$, so one can again check that $\mathsf{case}\ \bar{n}$ and $\bar{n}$ have the same type for Scott numeral $\bar{n}$ using the *selfInst* rule.

Observe that the term for rec and case (even the *iterator* in system **F**) is the term $\lambda n.\lambda f.\lambda a.n\ f\ a$. We call the type of this term *elimination scheme*. By elemination scheme is derivable in Selfstar, we mean the elimination scheme is inhabited by the term $\lambda n.\lambda f.\lambda a.n\ f\ a$ in Selfstar (modulo type annotations).

1.2 Overview

In section 2, we present Selfstar. We show how to type Scott-encoded data and its derivative. The corresponding elimination schemes for Scott-encoded data are derived. We also provide several examples (numerals and vectors) to demonstrate the power of Selfstar. In section 3, metatheorems such as type preservation are proved for Selfstar by applying the method we develped in **S** [5].

2. Dependently-typed programming with Selfstar

Selfstar use the self type mechanism to obtain inductive data, resulting a design that is simpler than most dependently-typed core languages. Intuitively, it is hard imagine how to emulate inductive datatype and pattern matching without any build-in mechanisms. But as we observe in section 1.1, Scott encoding together with mutually recursive definitions are enough to perform pattern matching on inductive data. The real difficulty lies in the typing. We want to make sure that both Scott-encoded data and definable operations on these data are typable in Selfstar. The self type allows us to type Scott data and to derive the corresponding elimination schemes. Thus operations on Scott data is typable using the elimination scheme.

2.1 System Selfstar

We give the full specification of Selfstar in this section. We use gray boxes to hightlight certain important terms and rules.

Definition 1 (Syntax)**.**
Terms $t\ ::= * \mid x \mid \lambda x.t \mid tt' \mid \mu t \mid \Pi x : t_1.t_2 \mid \iota x.t$
Closure $\mu\ ::= \{x_i \mapsto t_i\}_{i \in N}$
Value $v\ ::= * \mid \lambda x.t \mid \Pi x : t_1.t_2 \mid \iota x.t \mid \vec{\mu}(\Pi x : t_1.t_2) \mid \vec{\mu}(\iota x.t)$
Context $\Gamma\ ::= \cdot \mid \Gamma, x : t \mid \Gamma, \tilde{\mu}$

Remarks :

- If μ is $\{x_i \mapsto t_i\}_{i \in N}$, then $\tilde{\mu}$ is $\{(x_i : a_i) \mapsto t_i\}_{i \in N}$ for some term a_i.
- For $\{x_i \mapsto t_i\}_{i \in N}$, we require for any $1 \leq i \leq n$, the free variable set $\mathrm{FV}(t_i) \subseteq \mathrm{dom}(\mu) = \{x_1, ..., x_n\}$. We also do not allow any reductions and substitutions inside the closure. We call this the *locality* restriction. Without locality requirement, it is hard to establish confluence for reductions(see [1]).
- $\mathrm{FV}(\mu t) = \mathrm{FV}(t) - \mathrm{dom}(\mu)$.

$$\frac{}{\cdot \vdash \mathsf{wf}} \qquad \frac{\Gamma \vdash \mathsf{wf} \quad \{\Gamma, \tilde{\mu} \vdash t_j : a_j\}_{(t_j : a_j) \in \tilde{\mu}}}{\Gamma, \tilde{\mu} \vdash \mathsf{wf}}$$

$$\frac{\Gamma \vdash \mathsf{wf} \quad \Gamma \vdash t : *}{\Gamma, x : t \vdash \mathsf{wf}}$$

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

$$\frac{}{\Gamma \vdash * : *}\ \textit{Star} \qquad \frac{(x : t) \in \Gamma}{\Gamma \vdash x : t}\ \textit{Var}$$

$$\frac{\Gamma, x : \iota x.t \vdash t : *}{\Gamma \vdash \iota x.t : *}\ \textit{Self} \qquad \frac{\Gamma \vdash t : \iota x.t'}{\Gamma \vdash t : [t/x]t'}\ \textit{SelfInst}$$

$$\frac{\Gamma, x : t_1 \vdash t_2 : * \quad \Gamma \vdash t_1 : *}{\Gamma \vdash \Pi x : t_1.t_2 : *}\ \textit{Pi}$$

$$\frac{\Gamma \vdash t : [t/x]t' \quad \Gamma \vdash \iota x.t' : *}{\Gamma \vdash t : \iota x.t'}\ \textit{SelfGen}$$

$$\frac{\Gamma \vdash t : t_1 \quad \Gamma \vdash t_1 \cong t_2 \quad \Gamma \vdash t_2 : *}{\Gamma \vdash t : t_2}\ \textit{Conv}$$

$$\frac{\Gamma, x : t_1 \vdash t : t_2 \quad \Gamma \vdash t_1 : *}{\Gamma \vdash \lambda x.t : \Pi x : t_1.t_2}\ \textit{Lam}$$

$$\frac{\Gamma \vdash t : \Pi x : t_1.t_2 \quad \Gamma \vdash t' : t_1}{\Gamma \vdash tt' : [t'/x]t_2}\ \textit{App}$$

$$\frac{\Gamma, \tilde{\mu} \vdash t : t' \quad \{\Gamma, \tilde{\mu} \vdash t_j : a_j\}_{(t_j : a_j) \in \tilde{\mu}}}{\Gamma \vdash \mu t : \mu t'}\ \textit{Mu}$$

Figure 2. Typing $\boxed{\Gamma \vdash t : t'}$

$$\frac{}{\Gamma \vdash \vec{\mu} * \rightsquigarrow *} \qquad \frac{(x_i \mapsto t_i) \in \Gamma}{\Gamma \vdash x_i \rightsquigarrow t_i}$$

$$\frac{(x_i \mapsto t_i) \in \mu \in \vec{\mu}}{\Gamma \vdash \vec{\mu} x_i \rightsquigarrow \vec{\mu} t_i} \qquad \frac{x \notin \mathrm{dom}(\vec{\mu})}{\Gamma \vdash \vec{\mu} x \rightsquigarrow x}$$

$$\frac{}{\Gamma \vdash \vec{\mu}(tt') \rightsquigarrow (\vec{\mu} t)(\vec{\mu} t')} \qquad \frac{}{\Gamma \vdash \vec{\mu}(\lambda x.t) \rightsquigarrow \lambda x.(\vec{\mu} t)}$$

$$\frac{}{\Gamma \vdash (\lambda x.t)v \rightsquigarrow [v/x]t} \qquad \frac{\Gamma \vdash t \rightsquigarrow t''}{\Gamma \vdash tt' \rightsquigarrow t''t'}$$

$$\frac{\Gamma \vdash t' \rightsquigarrow t''}{\Gamma \vdash (\lambda x.t)t' \rightsquigarrow (\lambda x.t)t''}$$

Figure 3. Executions

$$\frac{\Gamma \vdash t_1 \rightsquigarrow^* t_2}{\Gamma \vdash t_1 = t_2} \qquad \frac{}{\Gamma \vdash (\lambda x.t)t' = [t'/x]t}$$

$$\frac{\mathrm{FV}(t)\#\mathrm{dom}(\mu)}{\Gamma \vdash \mu t = t} \qquad \frac{}{\Gamma \vdash \mu(\iota x.t) = \iota x.(\mu t')}$$

$$\frac{}{\Gamma \vdash \mu(\Pi x : t_1.t_2) = \Pi x : \mu t_1.\mu t_2} \qquad \frac{\mathrm{FV}(t)\#\mathrm{dom}(\mu)}{\Gamma \vdash \mu t = t}$$

$$\frac{\Gamma \vdash t = t'' \quad \Gamma \vdash t' = t'''}{\Gamma \vdash tt' = t''t'''} \qquad \frac{\Gamma, \tilde{\mu} \vdash t = t'}{\Gamma \vdash \mu t = \mu t'}$$

$$\frac{\Gamma \vdash t = t'}{\Gamma \vdash \iota x.t = \iota x.t'} \qquad \frac{\Gamma \vdash t = t'}{\Gamma \vdash \lambda x.t = \lambda x.t'}$$

$$\frac{\Gamma \vdash t_2 = t_3 \quad \Gamma \vdash t_1 = t_2}{\Gamma \vdash t_1 = t_3} \qquad \frac{\Gamma \vdash t_2 = t_1}{\Gamma \vdash t_1 = t_2}$$

Figure 4. Equality

$$\frac{\mu \in \Gamma}{\Gamma \vdash \mu t \to_o t} \qquad \frac{\Gamma \vdash t \to_o t'}{\Gamma \vdash \lambda x.t \to_o \lambda x.t'}$$

$$\frac{\Gamma \vdash t \to_o t''}{\Gamma \vdash tt' \to_o t''t'} \qquad \frac{\Gamma \vdash t' \to_o t''}{\Gamma \vdash tt' \to_o tt''}$$

$$\frac{\Gamma \vdash t_1 \to_o t_1'}{\Gamma \vdash \Pi x : t_1.t_2 \to_o \Pi x : t_1'.t_2'} \qquad \frac{\Gamma, \tilde{\mu} \vdash t \to_o t'}{\Gamma \vdash \mu t \to_o \mu t'}$$

$$\frac{\Gamma \vdash t_2 \to_o t_2'}{\Gamma \vdash \Pi x : t_1.t_2 \to_o \Pi x : t_1'.t_2'} \qquad \frac{\Gamma \vdash t \to_o t'}{\Gamma \vdash \iota x.t \to_o \iota x.t'}$$

Figure 5. Closure Reductions

Remarks:

- $(t_i : a_i) \in \tilde{\mu}$ means $(x_i : a_i) \mapsto t_i \in \tilde{\mu}$. $\vec{\mu} t$ denotes $\mu_1 ... \mu_n t$.
- Typing does not depend on well-formness of the context, so the self type formation rule *self* is not circular in this sense. We will show: if $\Gamma \vdash \mathsf{wf}$ and $\Gamma \vdash t : t'$, then $\Gamma \vdash t' : *$ (Appendix A).
- We use call-by-value strategy for the execution.
- $\cong$ denotes $=_o \cup =$, where $=_o$ denotes the reflexive transitive and symmetry closure of $\to_o$.
- The equality rules incorpates executions to automatize a portion of equality reasoning.
- At type level, we want to have the ability to open the closure when it appear in the context. closure reduction allow us to do this, without this type level reduction, we can not prove type preservation.

2.2 Scott Encodings in Selfstar

Now let us see some concrete examples of Scott encodings in Selfstar. For convenient, we write $a \to b$ for $\Pi x : a.b$ with $x \notin \mathrm{FV}(b)$.

Definition 2 (Scott's Derivative)**.** *Let $\tilde{\mu}_d$ be the following recursive defintions:*
$(\mathsf{Nat} : *) \mapsto$
$\iota x.\Pi C : \mathsf{Nat} \to *.(\Pi n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n))) \to (C\ 0) \to C\ x$
$(\mathsf{S} : \mathsf{Nat} \to \mathsf{Nat}) \mapsto \lambda n.\lambda C.\lambda s.\lambda z.s\ n\ (n\ C\ s\ z)$
$(0 : \mathsf{Nat}) \mapsto \lambda C.\lambda s.\lambda z.z$
with $s : \Pi n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n)), z : C0, n : \mathsf{Nat}$, *we have* $\tilde{\mu}_d \vdash \mathsf{wf}$ *(using selfGen and selfInst rules).*

Definition 3 (Elimination Scheme for Scott's Derivative)**.**
$\tilde{\mu}_d \vdash \mathsf{Rec} : \Pi C : \mathsf{Nat} \to *.\Pi n : \mathsf{Nat}.((C\ n) \to C\ (\mathsf{S}\ n)) \to C\ 0 \to \Pi n : \mathsf{Nat}.C\ n$
$\mathsf{Rec} := \lambda C.\lambda s.\lambda z.\lambda n.n\ C\ s\ z$
with $s : \Pi n : \mathsf{Nat}.(C\ n) \to (C\ (\mathsf{S}\ n)), z : C0, n : \mathsf{Nat}$.

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

Definition 4 (Scott Numerals)**.** *Let $\tilde{\mu}_s$ be the following recursive defintions:*
$(\mathsf{Nat} : *) \mapsto$
$\iota x.\Pi C : \mathsf{Nat} \to *.\ (\Pi n : \mathsf{Nat}.C\ (\mathsf{S}\ n)) \to (C\ 0) \to (C\ x)$
$(\mathsf{S} : \mathsf{Nat} \to \mathsf{Nat}) \mapsto \lambda n.\lambda C.\lambda s.\lambda z.s\ n$
$(0 : \mathsf{Nat}) \mapsto \lambda C.\lambda s.\lambda z.z$
with $s : \Pi n : \mathsf{Nat}.C\ (\mathsf{S}\ n), z : C0, n : \mathsf{Nat}$, *we have* $\tilde{\mu}_s \vdash \mathsf{wf}$*(using SelfGen and SelfInst rules).*

Definition 5 (Elimination Scheme for Scott Numerals)**.**
$\tilde{\mu}_s \vdash \mathsf{Case} : \Pi C : \mathsf{Nat} \to *.\Pi n : \mathsf{Nat}.(C\ (\mathsf{S}\ n)) \to C\ 0 \to \Pi n : \mathsf{Nat}.C\ n$
$\mathsf{Case} := \lambda C.\lambda s.\lambda z.\lambda n.n\ C\ s\ z$
with $s : \Pi n : \mathsf{Nat}.(C\ (\mathsf{S}\ n)), z : C0, n : \mathsf{Nat}$*(using SelfInst rule).*

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

Definition 6 (Addition)**.** *We define μ_+:*
$(\mathsf{add} : \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}) \mapsto$
$\quad \lambda n.\lambda m.\mathsf{Case}\ (\lambda n.\mathsf{Nat})\ (\lambda p.(\mathsf{S}\ (\mathsf{add}\ p\ m)))\ m\ n$

One can check that $\tilde{\mu}_s, \mu_+ \vdash \mathsf{wf}$.

Definition 7 (Leibniz Convertability)**.**
$\mathsf{Eq} := \lambda A.\lambda x.\lambda y.\Pi C : A \to *.C\ x \to C\ y.$

Definition 8. $\tilde{\mu}_s \vdash \mathsf{addZ} : \Pi x : \mathsf{Nat}.\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ x\ 0)\ x.$

Typing: We are trying to show
$\Pi x : \mathsf{Nat}.\Pi C : \mathsf{Nat} \to *.C\ (\mathsf{add}\ x\ 0) \to C\ x$
is inhabited. We know that the type of case $(\lambda z.(\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ z\ 0)\ z))$
is $\Pi n : \mathsf{Nat}.(\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ (\mathsf{S}n)\ 0)\ \mathsf{S}n) \to (\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ 0\ 0)\ 0) \to \Pi n : \mathsf{Nat}.\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ n\ 0)\ n$.
So case $(\lambda z.(\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ z\ 0)\ z))\ p_1\ p_2 : \Pi n : \mathsf{Nat}.\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ n\ 0)\ n$,
with $p_1 : \Pi n : \mathsf{Nat}.(\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ (\mathsf{S}n)\ 0)\ \mathsf{S}n)$ and $p_2 : \mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ 0\ 0)\ 0$.
It is easy to see that $p_2 := \lambda C[: \mathsf{Nat} \to *].\lambda x[: C(\mathsf{add}\ 0\ 0)].x$.
We know $\mathsf{addZ}\ n\ (\lambda q[: \mathsf{Nat}].C(\mathsf{S}q)) : C(\mathsf{S}(\mathsf{add}\ n\ 0)) \to C(\mathsf{S}n)$.
Thus $p_1 :=$
$\lambda n[: \mathsf{Nat}].\lambda C[: \mathsf{Nat} \to *].\lambda z[: C(\mathsf{add}\ (\mathsf{S}n)\ 0)].$
$\quad (\mathsf{addZ}\ n\ (\lambda q[: \mathsf{Nat}].C(\mathsf{S}q)))\ z.$
So we arrive at the following definition:
$\mathsf{addZ} := \mathsf{case}\ (\lambda z.(\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ z\ 0)\ z))$
$\quad (\lambda n.\lambda C.\lambda z.(\mathsf{addZ}\ n\ (\lambda q.C(\mathsf{S}q)))\ z)$
$\quad (\lambda C.\lambda x.x) =_\beta$
$\quad \lambda y.y\ (\lambda z.(\mathsf{Eq}\ \mathsf{Nat}\ (\mathsf{add}\ z\ 0)\ z))$
$\quad (\lambda n.\lambda C.\lambda z.(\mathsf{addZ}\ n\ (\lambda q.C(\mathsf{S}q)))\ z)\ (\lambda C.\lambda x.x)$

Observe that addZ is a recursive function that is equivalent to $\lambda C.\lambda z.z$ for all input of Scott numerals. So it is safe to use $\mathsf{addZ}\ \bar{n}$ to convert $\mathsf{add}\ \bar{n}\ 0$ to $\bar{n}$.

Definition 9 (Vector)**.**
Let $\tilde{\mu}_v$ be the following recursive defintions:

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

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

Definition 10 (Elimination Scheme for Vector)**.**
$\tilde{\mu}_v \vdash \mathsf{Case}(U,n) :$
$\quad \Pi C : (\Pi p : \mathsf{Nat}.\mathsf{vec}(U,p) \to *).$
$\quad (\Pi m : \mathsf{Nat}.\Pi u : U.\Pi y : \mathsf{vec}(U,m).(C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y)))$
$\quad \to C\ 0\ \mathsf{nil} \to \Pi x : \mathsf{vec}(U,n).(C\ n\ x)$
where $\mathsf{Case}(U,n) := \lambda C.\lambda s.\lambda z.\lambda x.x\ C\ s\ z$
$C : (\Pi p : \mathsf{Nat}.\mathsf{vec}(U,p) \to *), s : \Pi m : \mathsf{Nat}.\Pi u : U.\Pi y : \mathsf{vec}(U,m).(C\ (\mathsf{S}m)\ (\mathsf{cons}\ m\ u\ y)), z : C\ 0\ \mathsf{nil}, x : \mathsf{vec}(U,n)$.

Definition 11 (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 $\mathsf{app} := \lambda n_1.\lambda n_2.\lambda l_1.\lambda l_2.$
$\quad \mathsf{Case}(U,n_1)(\lambda z.\lambda q.\mathsf{vec}(U,z+n_2))$
$\quad (\lambda m.\lambda h.\lambda t.\mathsf{cons}\ (m+n_2)\ h\ (\mathsf{app}\ m\ n_2\ t\ l_2))$
$\quad l_2\ l_1$

Typing: We want to show $\mathsf{app} : \Pi n_1 : \mathsf{Nat}.\Pi n_2 : \mathsf{Nat}.\mathsf{vec}(U,n_1) \to \mathsf{vec}(U,n_2) \to \mathsf{vec}(U,n_1+n_2)$. We instantiate $C := \lambda z.(\lambda q.\mathsf{vec}(U,z+n_2))$, where q free over $\mathsf{vec}(U,y+n_2)$, in $\mathsf{Case}(U,n_1)$. By beta reductions, we get $\mathsf{Case}(U,n_1)\ (\lambda z.\lambda q.\mathsf{vec}(U,z+n_2)) : \Pi m : \mathsf{Nat}.\Pi u : U.\Pi y : \mathsf{vec}(U,m).(\mathsf{vec}(U,(\mathsf{S}m)+n_2)) \to \mathsf{vec}(U,0+n_2) \to \Pi x : \mathsf{vec}(U,n_1).\mathsf{vec}(U,n_1+n_2)$.
Also, $\lambda m.\lambda h.\lambda t.\mathsf{cons}\ (m+n_2)\ h\ (\mathsf{app}\ m\ n_2\ t\ l_2) : \Pi m : \mathsf{Nat}.\Pi h : U.\Pi t : \mathsf{vec}(U,m).(\mathsf{vec}(U,(\mathsf{S}\ (m+n_2)))$
With $l_1 : \mathsf{vec}(U,n_1), l_2 : \mathsf{vec}(U,n_2)$, we can see that it is the case.

Definition 12 (Associativity)**.**
$\tilde{\mu}_v \vdash \mathsf{assoc} : \Pi(n_1.n_2.n_3 : \mathsf{Nat}).$
$\quad \Pi(v_1 : \mathsf{vec}(U,n_1).v_2 : \mathsf{vec}(U,n_2).v_3 : \mathsf{vec}(U,n_3)).$
$\quad \mathsf{Eq}\ \mathsf{vec}(U,n_1{+}n_2{+}n_3)\ (\mathsf{app}\ n_1\ (n_2{+}n_3)\ v_1\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3))$
$\quad (\mathsf{app}\ (n_1+n_2)\ n_3\ (\mathsf{app}\ n_1\ n_2\ v_1\ v_2)\ v_3)$

3. Metatheory

The proof of type preservation for Selfstar follows the same method for proving type preservation of **S**. Selfstar is a simpler system compare to **S** in the following senses: 1. With $* : *$, we no longer have separate syntactic categories for types and kinds. 2. Polymorphism is anotated in Selfstar, which leads to an easier proof of type preservation, namely, we do not need *morph analysis* [5] for Selfstar. These simplifications lead to a simpler proof of type preservation.

In order to prove type preservation for Selfstar, we need *confluence analysis* for the type level transformation. We need to show

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

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

$$\frac{\Gamma \vdash t \to_\beta t''}{\Gamma \vdash tt' \to_\beta t''t'} \qquad \frac{\Gamma \vdash t' \to_\beta t''}{\Gamma \vdash tt' \to_\beta tt''}$$

$$\frac{\Gamma, \tilde{\mu} \vdash t \to_\beta t'}{\Gamma \vdash \mu t \to_\beta \mu t'} \qquad \frac{\Gamma \vdash t \to_\beta t'}{\Gamma \vdash \iota x.t \to_\beta \iota x.t'}$$

$$\frac{\Gamma \vdash t_2 \to_\beta t_2'}{\Gamma \vdash \Pi x : t_1.t_2 \to_\beta \Pi x : t_1'.t_2'} \qquad \frac{\Gamma \vdash t_1 \to_\beta t_1'}{\Gamma \vdash \Pi x : t_1.t_2 \to_\beta \Pi x : t_1'.t_2'}$$

Figure 6. Beta Reductions

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

$$\frac{}{\Gamma \vdash \mu(t_1 t_2) \to_\mu (\mu t_1)(\mu t_2)} \qquad \frac{}{\Gamma \vdash \mu(\iota x.t) \to_\mu \iota x.\mu t}$$

$$\frac{}{\Gamma \vdash \mu(\Pi x : t_1.t_2) \to_\mu \Pi x : \mu t_1.\mu t_2} \qquad \frac{\Gamma \vdash t \to_\mu t'}{\Gamma \vdash \lambda x.t \to_\mu \lambda x.t'}$$

$$\frac{\Gamma \vdash t' \to_\mu t''}{\Gamma \vdash tt' \to_\mu tt''} \qquad \frac{\Gamma \vdash t \to_\mu t''}{\Gamma \vdash tt' \to_\mu t''t'}$$

$$\frac{\Gamma \vdash t_1 \to_\mu t_1'}{\Gamma \vdash \Pi x : t_1 t_2 \to_\mu \Pi x : t_1.t_2'} \qquad \frac{\Gamma \vdash t \to_\mu t'}{\Gamma \vdash \iota x.t \to_\mu \iota x.t'}$$

$$\frac{\Gamma \vdash t_2 \to_\mu t_2'}{\Gamma \vdash \Pi x : t_1 t_2 \to_\mu \Pi x : t_1.t_2'} \qquad \frac{\Gamma, \tilde{\mu} \vdash t \to_\mu t'}{\Gamma \vdash \mu t \to_\mu \mu t'}$$

Figure 7. Mu Reductions

type level transformation is confluent. Thus the transformation from $\Pi x : t_1.t_2$ to $\Pi x : t_1'.t_2'$ implies t_1 can be transformed to t_1' and t_2 can be transformed to t_2'. Thus we establish the *compatibility* property for Selfstar, which is the major result in order to prove type preservation. The proofs for this section are in Appendix C, D and E. Once we prove type preservation, progress theorem is easy to prove, see Appendix B.

3.1 The Analytical System

It is combersome to directly prove the equality in Selfstar is Church-Rosser. We develop an *analytical system* and we prove that the analytical system is equivalent (theorem 16) to the equality system in Selfstar. Then we prove the analytical system is confluent, which implies the Church-Rosser of the equality in Selfstar.

The beta-reductions (figure 6) include definition substitutions and the ordinary beta-reduction in lambda calculus. The mu-reductions (figure 7) are for moving the closure inside the term structure.

Let $\to$ denote $\to_\beta \cup \to_\mu$. Let $\leftrightarrow^*$ denote $(\to \cup \to^{-1})^*$. The following lemmas show the relation between $\to$ and $=$.

Lemma 13. *If* $\Gamma \vdash t_1 \rightsquigarrow t_2$, *then* $\Gamma \vdash t_1 \to t_2$.

Proof. By induction on derivation of $\Gamma \vdash t_1 \rightsquigarrow t_2$. $\square$

Lemma 14. *If* $\Gamma \vdash t_1 = t_2$*, then* $\Gamma \vdash t_1 \leftrightarrow^* t_2$.

Proof. By induction on the derivation of $\Gamma \vdash t_1 = t_2$. □

Lemma 15. *If* $\Gamma \vdash t_1 \to t_2$*, then* $\Gamma \vdash t_1 = t_2$.

Proof. By induction on the derivation of $\Gamma \vdash t_1 \to t_2$. □

The following theorem shows that the analytic system is equivalent to the equality system.

Theorem 16. $\Gamma \vdash t_1 = t_2$ *iff* $\Gamma \vdash t_1 \leftrightarrow^* t_2$.

Proof. By lemma 14, lemma 15. □

Suppose $\to$ is confluent. By theorem 16, we know that $\Gamma \vdash \Pi x : t_1.t_2 = \Pi x : t_1'.t_2'$ implies $\Gamma \vdash \Pi x : t_1.t_2 \leftrightarrow^* \Pi x : t_1'.t_2'$. The confluence of $\to$ implies Church-Rosser of $\leftrightarrow^*$, namely, there exists a t such that $\Gamma \vdash \Pi x : t_1.t_2 \to^* t$ and $\Gamma \vdash \Pi x : t_1'.t_2' \to^* t$. By definition of $\to$, we know t must be of the form $\Pi x : t_3.t_4$, with $\Gamma \vdash t_1 \to t_3$, $\Gamma \vdash t_1' \to t_3$, $\Gamma \vdash t_2 \to t_4$ and $\Gamma \vdash t_2' \to t_4$. So by lemma 15, we have $\Gamma \vdash t_1 = t_1'$ and $\Gamma \vdash t_2 = t_2'$.

Now let us focus on the proof of the confluence of $\to$. The confluence argument is similar to the one described in [3]. We are going to use the following lemma to conclude the confluence of $\to_\beta \cup \to_\mu$.

Lemma 17 (Hardin's interpretation lemma[7])**.** *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:*

$\to_i \subseteq \twoheadrightarrow$, [1] *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 idea behinds interpretation method is that it allows us to modulo the $\to_1$-reduction, we only need to focus on proving the confluence of $\to_i$. This is essential since in our case, $\to_{\beta,\mu}$ can not be directly parallelized, namely, one can not use Tait-Martin Löf's method (reported in [2]) directly to prove the confluence of $\to_{\beta,\mu}$, the paralleled version does not enjoy diamond property. With the interpretation method, after modulo the $\to_\mu$-reduction, we introduce a new reduction $\to_{\beta\mu}$ (corresponds to $\to_i$), we can then use the parallel reduction method to prove confluence of $\to_{\beta\mu}$.

Lemma 18. $\to_\mu$ *is confluent and terminating.*

So $\to_\mu$ correspond to $\to_1$ in the interpretation lemma. Since $\to_\mu$ is strongly normalizing and confluent, we can define a normalization function which effectively computes the mu-normal form.

Definition 19 (μ-Normal Forms)**.**
$n ::= * \mid x \mid \mu x_i \mid \lambda x.n \mid nn' \mid \Pi x : n.n' \mid \iota x.n$

Note: for the μx_i in definition 19, we assume $x_i \in \mathrm{dom}(\mu)$.

[1] $\twoheadrightarrow$ is the reflexive symmetric transitive closure of $\to$.

Definition 20 (μ normalization function)**.**

$m(*) := *$
$m(x) := x$
$m(\lambda y.t) := \lambda y.m(t)$
$m(t_1 t_2) := m(t_1)m(t_2)$
$m(\iota x.t) := \iota x.m(t)$
$m(\Pi x : t.t') := \Pi x : m(t).m(t')$.
$m(\vec{\mu} y) := y$ *if* $y \notin dom(\vec{\mu})$.
$m(\vec{\mu} y) := \mu_i y$ *if* $y \in dom(\mu_i)$.
$m(\vec{\mu}(tt')) := m(\vec{\mu}t)m(\vec{\mu}t')$
$m(\vec{\mu}(\lambda x.t)) := \lambda x.m(\vec{\mu}t)$.
$m(\vec{\mu}(\iota x.t)) := \iota x.m(\vec{\mu}t)$.
$m(\vec{\mu}(\Pi x : t.t')) := \Pi x : m(\vec{\mu}t).m(\vec{\mu}t')$.

We shall devise a new notion of reduction on mu-normal form, then show that this reduction is confluent (corresponds to $\to_i$ in the interpretation lemma and satisfying the † property), thus by the interpretation lemma, we can show $\to_{\beta,\mu}$ is confluent[2]. 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 21 (β Reduction on μ-normal Forms)**.**

$$\frac{\Gamma \vdash n \to_\beta t}{\Gamma \vdash n \to_{\beta\mu} m(t)}$$

The following lemma shows that $\to_{\beta\mu}$ corresponds to the $\to_i$ in the interpretation lemma.

Lemma 22. *If* $\Gamma \vdash t \to_\beta t'$*, then* $\Gamma \vdash m(t) \to_{\beta\mu} m(t')$.

Lemma 23. $\to_{\beta\mu}$ *is confluent.*

Theorem 24. $\to_{\beta,\mu}$ *is confluent.*

Proof. We know $\to_{\beta\mu}$ is confluent. Since $\to_\mu$ is strongly normalizing and confluent, and by lemma 22 and Hardin's interpretation lemma(lemma 17), we conclude $\to_{\beta,\mu}$ is confluent. □

3.2 Confluence Analysis

Definition 25.
$\Gamma \vdash t_1 \to_\iota t_2$ *if* $t_1 \equiv \iota x.t'$ *and* $t_2 \equiv [t/x]t'$ *for some fix term t.*

Note that $\to_\iota$ models the *selfInst* rule, $\to_\iota^{-1}$ models the *selfGen* rule. The notion of ι-reduction does not build in structure congruence, 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*. This rewriting point of view on typing is inspired by Kuan et.al. [10] and Stump et.al. [12].

Lemma 26 (Confluence)**.** $\to_\iota$ *is confluent.*

Proof. This is obvious since $\to_\iota$ is deterministic. □

The goal of this section is to show $\to_{o,\iota,\beta,\mu}$ is confluent. We make extensive use of the notion of *commutativity*, which provides a simple way to prove the confluence of a reduction system that has several confluent subreductions.

Definition 27 (Commutativity)**.** *Let* $\to_1, \to_2$ *be two notions of reduction.* $\to_1$ *(strongly) commute with* $\to_2$ *if* $a \to_1 b_1$ *and* $a \to_2 b_2$*, then there exists a c such that* $b_1 \to_2 c$ *and* $b_2 \to_1 c$.

Proposition 28 (Hindley-Rosen [8] [11])**.** *Let* $\to_1, \to_2$ *be two notions of reduction. Suppose both* $\to_1$ *and* $\to_2$ *are confluent, and* $\to_1^*$ *commutes with* $\to_2^*$*. Then* $\to_1 \cup \to_2$ *is confluent.*

[2] $\to_{\beta,\mu}$ denotes $\to_\beta \cup \to_\mu$, we will use this convention throughout the paper.

Proposition 29 (Weak Commutativity [2]). *Let $\hookrightarrow$ denote the reflexive closure of $\to$. Let $\to_1, \to_2$ be two notions of reduction. $\to_1$ weak commutes with $\to_2$ if $a \to_1 b_1$ and $a \to_2 b_2$, then there exists a c such that $b_1 \hookrightarrow_2 c$ and $b_2 \twoheadrightarrow_1 c$.*

If $\to_1$ weak commutes with $\to_2$, then $\to_1^*$ and $\to_2^*$ commute.

Lemma 30. *$\to_{\beta,\mu}$ commutes with $\to_\iota$. Thus $\to_{\beta,\mu,\iota}$ is confluent.*

Lemma 31. *$\to_o$ has diamond property, thus is confluent.*

Lemma 32. *$\to_o$ commutes with $\to_\iota$, weak commutes with $\to_\beta$, $\to_\mu$.*

Theorem 33. *$\to_{o,\iota,\beta,\mu}$ is confluent.*

Let $=_{\beta,\mu,\iota,o}$ denotes the reflexive transitive symmetry closure of $\to_o \cup \to_\iota \cup \to_\beta \cup \to_\mu$. The goal of confluence analysis is to establish the following theorem.

Theorem 34 (ι-elimination, Compatibility).
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 $\Gamma \vdash t_2 =_{\beta,\mu,o} t_2'$. $\square$

3.3 Type Preservation

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 35 (Inversion).

- *If $\Gamma \vdash \lambda x.t : t'$, then $\Gamma, x : t_1 \vdash t : t_2$ and $\Gamma \vdash \Pi x : t_1.t_2 =_{\beta,\mu,\iota,o} t'$ for some t_1, t_2.*
- *If $\Gamma \vdash t_1 t_2 : t'$, then $\Gamma \vdash t_1 : \Pi x : t_1'.t_2'$ and $\Gamma \vdash t_2 : t_1'$, $\Gamma \vdash [t_2/x]t_2' =_{\beta,\mu,\iota,o} t'$ for some t_1', t_2'.*
- *If $\Gamma \vdash x : t'$, then $x : t \in \Gamma$ and $\Gamma \vdash t =_{\beta,\mu,\iota,o} t'$ for some t.*

Lemma 36 (Substitution). *If $\Gamma_1, x : t_1, \Gamma_2 \vdash t : t_2$ and $\Gamma \vdash t' : t_1$, then $\Gamma_1, [t'/x]\Gamma_2 \vdash [t'/x]t : [t'/x]t_2$.*

Theorem 37 (Type Preservation). *If $\Gamma \vdash \mathsf{wf}$ and $\Gamma \vdash t \leadsto t'$ and $\Gamma \vdash t : t''$, then $\Gamma \vdash t' : t''$.*

Proof. We list one interesting case here.

$$\frac{\Gamma \vdash t_1' : \Pi x : t_1''.t_2'' \quad \Gamma \vdash t_2' : t_1''}{\Gamma \vdash t_1' t_2' : [t_2'/x]t_2''}$$

Suppose $\Gamma \vdash (\lambda x.t_1)v \leadsto [v/x]t_1$. Then we know $\Gamma \vdash (\lambda x.t_1)v : [t/x]t_2''$ and $\Gamma \vdash \lambda x.t_1 : \Pi x : t_1''.t_2''$ and $\Gamma \vdash v : t_1''$. By inversion on $\Gamma \vdash \lambda x.t_1 : \Pi x : t_1''.t_2''$, we have $\Gamma, x : a \vdash t_1 : b$ and $\Gamma \vdash \Pi x : a.b =_{\beta,\mu,\iota,o} \Pi x : t_1''.t_2''$. By theorem 34, we have $\Gamma \vdash a =_{\beta,\mu,o} t_1''$ and $\Gamma \vdash b =_{\beta,\mu,o} t_2''$. So we have $\Gamma, x : a \vdash t_1 : t_2''$ and $\Gamma \vdash v : a$. So by lemma 79, we have $\Gamma \vdash [v/x]t_1 : [v/x]t_2''$, as required. $\square$

4. Conclusion

We introduce Selfstar, which incorporates the self type construct together with $* : *$ and mutually recursion. Scott-encoded datatypes and the corresponding elimination schemes are derivable within Selfstar. We also demonstrate the process of proving the type preservation theorem.

References

[1] Zena M. Ariola and Jan Willem Klop. Lambda calculus with explicit recursion. *Information and Computation*, 139(2):154 – 233, 1997.

[2] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics (Studies in Logic and the Foundations of Mathematics, Volume 103). Revised Edition.* North Holland, revised edition, November 1985.

[3] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *J. ACM*, 43(2):362–397, 1996.

[4] H. B. Curry, J. R. Hindley, and J. P. Seldin. *Combinatory Logic, Volume II.* North-Holland, 1972.

[5] Peng Fu and Aaron Stump. Church encoding with dependent types. 2013. submitted, available from first Author's web page.

[6] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.

[7] 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.

[8] James Roger Hindley. *The Church-Rosser property and a result in combinatory logic.* PhD thesis, University of Newcastle upon Tyne, 1964.

[9] J.M. Jansen, R. Plasmeijer, and P. Koopman. Functional pearl: Comprehensive encoding of data types and algorithms in the lambda-calculus. *Internal report, NLDA*, 2011.

[10] 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.

[11] Barry K. Rosen. Tree-manipulating systems and church-rosser theorems. *J. ACM*, 20(1):160–187, January 1973.

[12] Aaron Stump, Garrin Kimmell, and Roba El Haj Omar. Type preservation as a confluence problem. In *RTA*, pages 345–360, 2011.

A. Well-Form Type

Lemma 38. *If $\Gamma \vdash \mathsf{wf}$ and $\Gamma \vdash t : t'$, then $\Gamma \vdash t' : *$.*

Proof. By induction on derivation of $\Gamma \vdash t : t'$. We list a few nontrivial cases.

Case:

$$\frac{\Gamma \vdash t : \iota x.t'}{\Gamma \vdash t : [t/x]t'}\ \mathit{SelfInst}$$

By IH, we have $\Gamma \vdash \iota x.t' : *$. So by inversion, we have $\Gamma, x : \iota x.t' \vdash t' : *$. So by lemma 79, we know $\Gamma \vdash [t/x]t' : *$.

Case:

$$\frac{\Gamma, x : t_1 \vdash t : t_2 \quad \Gamma \vdash t_1 : *}{\Gamma \vdash \lambda x.t : \Pi x : t_1.t_2}\ \mathit{Lam}$$

By IH, we know $\Gamma, x : t_1 \vdash t_2 : *$. Since $\Gamma \vdash t_1 : *$, by *Pi* rule, we have $\Gamma \vdash \Pi x : t_1.t_2 : *$.

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, we have $\Gamma \vdash \Pi x : t_1.t_2 : *$. By inversion on $\Gamma \vdash \Pi x : t_1.t_2 : *$, we have $\Gamma, x : t_1 \vdash t_2 : *$. So by lemma 79, we have $\Gamma \vdash [t'/x]t_2 : *$.

Case:

$$\frac{\Gamma, \tilde{\mu} \vdash t : t' \quad \{\Gamma, \tilde{\mu} \vdash t_j : a_j\}_{(t_j : a_j) \in \tilde{\mu}}}{\Gamma \vdash \mu t : \mu t'} \; Mu$$

By IH, we have $\Gamma, \tilde{\mu} \vdash t' : *$. So $\Gamma \vdash \mu t' : \mu *$, thus $\Gamma \vdash \mu t' : *$. $\square$

B. Progress

Lemma 39. *If $\cdot \vdash v : \Pi x : t_1.t_2$, then $v \equiv \lambda x.t$.*

Proof. Case analysis on v. Suppose $v \equiv *$. By inversion, $\cdot \vdash * : *$ and $\cdot \vdash * =_{\beta,\mu,\iota,o} \Pi x : t_1.t_2$, which contradicts Church-Rosser of $=_{\beta,\mu,\iota,o}$. Suppose $v \equiv \vec{\mu}(\Pi x : t_3.t_4)$. By inversion, we have $\tilde{\vec{\mu}} \vdash \Pi x : t_3.t_4 : t_a$ and $\cdot \vdash \vec{\mu} t_a \overset{\vec{\mu}(\Pi x : t_3.t_4)}{=}_{\beta,\mu,\iota,o} \Pi x : t_1.t_2$. By inversion on $\tilde{\vec{\mu}} \vdash \Pi x : t_3.t_4 : t_a$, we have $\tilde{\vec{\mu}} \vdash * \overset{\Pi x : t_3.t_4}{=}_{\beta,\mu,\iota,o} t_a$. So we have $\cdot \vdash \vec{\mu} * \overset{\vec{\mu}(\Pi x : t_3.t_4)}{=}_{\beta,\mu,\iota,o} \vec{\mu} t_a \overset{\vec{\mu}(\Pi x : t_3.t_4)}{=}_{\beta,\mu,\iota,o} \Pi x : t_1.t_2$. Again, this contradicts Church-Rosser of $=_{\beta,\mu,\iota,o}$. For other cases like: $v \equiv \Pi x : t.t', \iota x.t, \vec{\mu}(\iota x.t)$, we argue similarly. $\square$

Theorem 40 (Progress)**.** *If $\cdot \vdash t : t''$, then either $\cdot \vdash t \leadsto t'$ or t is a value.*

Proof. By induction on the derivation of $\cdot \vdash t : t''$, we list a few cases.

Case:

$$\frac{\tilde{\mu} \vdash t : t' \quad \{\tilde{\mu} \vdash t_j : a_j\}_{(t_j : a_j) \in \tilde{\mu}}}{\cdot \vdash \mu t : \mu t'} \; Mu$$

Identify t as $\dot{\vec{\mu}} t''$, where t'' does not contains any closure at head position. Case analysis on t'', if it is $*, x, \lambda x.t_a, t_a t_b$, then there exist a t' such that $\cdot \vdash t \leadsto t'$. If $t'' \equiv \Pi x : t_a.t_b, \iota x.t_a$, then it is already a value.

Case:

$$\frac{\cdot \vdash t : \Pi x : t_1.t_2 \quad \cdot \vdash t' : t_1}{\cdot \vdash tt' : [t'/x]t_2} \; App$$

Since $\cdot \vdash t : \Pi x : t_1.t_2$ and $\cdot \vdash t' : t_1$, by IH, t either steps or is a value, likewise for t'. If t can take a step, then tt' can also take a step. If t is a value, by lemma 39, t must be of the form $\lambda x.t_a$. So if t' can take a step, then tt' can also take a step. If both t' is a value, then tt' can take a step.

$\square$

C. Proofs of Section 3.1

Let $\dot{\vec{\mu}}$ denote 0 or more closures.

Lemma 41. *Let Φ denote the set of μ normal form. For any term t, $m(t) \in \Phi$.*

Proof. One way to prove this is first identify t as $\dot{\overrightarrow{\mu_1}} t'$, here $\dot{\overrightarrow{\mu_1}}$ means there are zero or more closures and t' does not contains any closure at head position. Then we can proceed by induction on the structure of t':

Base Cases: $t' = x, t' = *$, obvious.

Step Cases: If $t' = \lambda x.t''$, then $m(\dot{\overrightarrow{\mu_1}}(\lambda x.t'')) \equiv \lambda x.m(\dot{\overrightarrow{\mu_1}} t'')$. Now we can again identify t'' as $\dot{\overrightarrow{\mu_2}} t'''$, where t''' does not have any closure at head position. Since t''' is structurally smaller than $\lambda x.t''$, by IH, $m(\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}} t''') \in \Phi$, thus $m(\dot{\overrightarrow{\mu_1}}(\lambda x.t'')) \equiv \lambda x.m(\dot{\overrightarrow{\mu_1}} t'') \in \Phi$.

For $t' = t_a t_b$, $t' = \iota x.t''$, $t' = \Pi x : t_a.t_b$, we can argue similarly as above. $\square$

In order to prove lemma 22, we prove the following more general lemma instead.

Lemma 42. *If $\Gamma, \dot{\vec{\mu}} \vdash a \to_\beta b$, then $\Gamma \vdash m(\dot{\vec{\mu}} a) \to_{\beta\mu} m(\dot{\vec{\mu}} b)$.*

Proof. By induction on derivation of $\Gamma, \dot{\vec{\mu}} \vdash a \to_\beta b$.
Base Case:

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

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

Base Case:

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

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

Base Case:

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

We have $\Gamma \vdash m(\dot{\vec{\mu}}((\lambda x.t)t')) \equiv (\lambda x.m(\dot{\vec{\mu}} t))m(\dot{\vec{\mu}} t') \to_{\beta\mu} m([m(\dot{\vec{\mu}} t)/x]m(\dot{\vec{\mu}} t')) \equiv m([\dot{\vec{\mu}} t/x]\dot{\vec{\mu}} t') \equiv m(\dot{\vec{\mu}}([t'/x]t))$. The last two equalities are by lemma 44, lemma 43.

Step Case:

$$\frac{\Gamma, \dot{\vec{\mu}} \vdash t \to_\beta t'}{\Gamma, \dot{\vec{\mu}} \vdash \lambda x.t \to_\beta \lambda x.t'}$$

$\Gamma \vdash m(\dot{\vec{\mu}}(\lambda x.t)) \equiv \lambda x.m(\dot{\vec{\mu}} t) \overset{IH}{\to}_{\beta\mu} \lambda x.m(\dot{\vec{\mu}} t') \equiv m(\dot{\vec{\mu}}(\lambda x.t'))$.

Step Case:

$$\frac{\Gamma, \dot{\vec{\mu}}, \tilde{\mu} \vdash t \to_\beta t'}{\Gamma, \dot{\vec{\mu}} \vdash \mu t \to_\beta \mu t'}$$

We want to show $\Gamma \vdash m(\dot{\vec{\mu}} \mu t) \to_{\beta\mu} m(\dot{\vec{\mu}} \mu t')$. This is directly by IH.

All the other cases are similar. $\square$

Lemma 43.
$m(\vec{\mu}\vec{\mu} t) \equiv m(\vec{\mu} t)$ and $m(\vec{\mu}([t_2/x]t_1)) \equiv m([\vec{\mu} t_2/x]\vec{\mu} t_1)$

Proof. We can prove this using the same method as lemma 41, namely, identify t and then proceed by inducton. $\square$

Lemma 44. *$m(m(t)) \equiv m(t)$ and $m([m(t_1)/y]m(t_2)) \equiv m([t_1/y]t_2)$.*

Proof. The first equality is by lemma 45 and lemma 41. For the second equality, we prove it using similar method as lemma 41: We identify t_2 as $\overrightarrow{\mu_1}t_2'$, where t_2' does not contains any closure at head position. We proceed by induction on the structure of t_2':

Base Cases: $t_2' = *$, obvious. For $t_2' = x$, we use $m(m(t)) \equiv m(t)$.

Step Cases: If $t_2' = \lambda x.t_2''$, then $m(\dot{\overrightarrow{\mu_1}}(\lambda x.[t_1/y]t_2'')) \equiv$
$\lambda x.m(\dot{\overrightarrow{\mu_1}}([t_1/y]t_2'')) \equiv \lambda x.m(\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}}([t_1/y]t_2'''))$, where t_2'' is identified as $\dot{\overrightarrow{\mu_2}}t_2'''$, and t_2''' does not have any closure at head position.
Since t_2''' is structurally smaller than $\lambda x.t_2''$, by IH,
$m(\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}}([t_1/y]t_2''')) \equiv m([t_1/y](\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}}t_2''')) \equiv$
$m([m(t_1)/y]m(\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}}t_2'''))$.
Thus $\lambda x.m(\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}}([t_1/y]t_2''')) \equiv \lambda x.m([m(t_1)/y]m(\dot{\overrightarrow{\mu_1}}\dot{\overrightarrow{\mu_2}}t_2'''))$,
implying $m([t_1/y]\dot{\overrightarrow{\mu_1}}(\lambda x.t_2'')) \equiv m([m(t_1)/y]m(\lambda x.\dot{\overrightarrow{\mu_1}}t_2''))$.
Since $m([m(t_1)/y]m(\lambda x.\dot{\overrightarrow{\mu_1}}t_2'')) \equiv m([m(t_1)/y]m(\dot{\overrightarrow{\mu_1}}(\lambda x.t_2'')))$,
we conclude $m([m(t_1)/y]m(\dot{\overrightarrow{\mu_1}}(\lambda x.t_2''))) \equiv m([t_1/y]\dot{\overrightarrow{\mu_1}}(\lambda x.t_2''))$.

For $t_2' = t_a t_b$, $t_2' = \iota x.t_2''$, $t_2' = \Pi x : t_a.t_b$, we can argue similarly as above. $\square$

Lemma 45. *If $n \in \Phi$, then $m(n) \equiv n$.*

Proof. By induction on the structure of n. $\square$

Definition 46 (β Reduction on μ-normal Forms)**.**

$$\frac{\Gamma \vdash n \to_\beta t}{\Gamma \vdash n \to_{\beta\mu} m(t)}$$

Note: From this definition we can conclude:

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

$$\frac{\Gamma \vdash n' \to_{\beta\mu} n''}{\Gamma \vdash \Pi x : n.n' \to_{\beta\mu} \Pi x : n.n''} \qquad \frac{\Gamma \vdash n \to_{\beta\mu} n''}{\Gamma \vdash nn' \to_{\beta\mu} n''n'}$$

$$\frac{\Gamma \vdash n \to_{\beta\mu} n''}{\Gamma \vdash \Pi x : n.n' \to_{\beta\mu} \Pi x : n''.n'} \qquad \frac{\Gamma \vdash n \to_{\beta\mu} n'}{\Gamma \vdash \iota x.n \to_{\beta\mu} \iota x.n'}$$

The first rule follows because: Assume $\Gamma \vdash n \to_{\beta\mu} n'$, say $m(t) \equiv n'$ and $\Gamma \vdash n \to_\beta t$. Then $\Gamma \vdash \lambda x.n \to_\beta \lambda x.t$ and $m(\lambda x.t) \equiv \lambda x.m(t) \equiv \lambda x.n'$. The others follow similarly.

Lemma 47. *If $\Gamma \vdash n_1 \to_{\beta\mu} n_1'$, then $\Gamma \vdash m([n_2/x]n_1) \to_{\beta\mu} m([n_2/x]n_1')$.*

Proof. By induction on derivation of $\Gamma \vdash n_1 \to_\beta t_1$, where $m(t_1) \equiv n_1'$. We will list a few nontrivial cases. Note that the we use lemma 44 implicitly.

Base Case:

$$\frac{(y \mapsto t_1) \in \Gamma}{\Gamma \vdash y \to_\beta t_1}$$

In this case $n_1 = y$. By locality, we have $\Gamma \vdash m([n_2/x]y) \equiv y \to_{\beta\mu} m(t_1) \equiv m([n_2/x]t_1)$.

Base Case:

$$\frac{}{\Gamma \vdash (\lambda y.n)n' \to_\beta [n'/y]n}$$

$n_1 = (\lambda y.n)n'$. So $\Gamma \vdash m([n_2/x]((\lambda y.n)n'))$
$\equiv m((\lambda y.[n_2/x]n)[n_2/x]n') \equiv (\lambda y.m([n_2/x]n))m([n_2/x]n') \to_{\beta\mu}$
$m([m([n_2/x]n')/y]m([n_2/x]n)) \equiv m([[n_2/x]n'/y]([n_2/x]n)) \equiv$
$m([n_2/x]([n'/y]n))$.

Base Case:

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

$n_1 = \mu x_i$. By locality, $\Gamma \vdash m([n_2/x]\mu x_i) \equiv \mu x_i \to_{\beta\mu} m(\mu t_i) \equiv m([n_2/x](\mu t_i))$.

Step Case:

$$\frac{\Gamma \vdash n \to_\beta t'}{\Gamma \vdash \lambda y.n \to_\beta \lambda y.t'}$$

$n_1 = \lambda y.n$. By IH, we have $\Gamma \vdash m([n_2/x]n) \to_{\beta\mu} m([n_2/x]t')$.
So $\Gamma \vdash m(\lambda y.[n_2/x]n) \to_{\beta\mu} m(\lambda y.[n_2/x]t')$.

Step Case:

$$\frac{\Gamma, \tilde{\mu} \vdash t \to_\beta t'}{\Gamma \vdash \mu t \to_\beta \mu t'}$$

This case will not arise since n_1 is already in μ normal form.

The other cases are similar. $\square$

Lemma 48. *If $\Gamma \vdash n_2 \to_{\beta\mu} n_2'$, then $\Gamma \vdash m([n_2/x]n_1) \to_{\beta\mu}^* m([n_2'/x]n_1)$.*

Proof. By induction on n_1. $\square$

Definition 49 (Parallel Reductions)**.**

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

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

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

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

Lemma 50. *$\to_{\beta\mu} \subseteq \Rightarrow_{\beta\mu} \subseteq \to_{\beta\mu}^*$.*

Proof. For $\to_{\beta\mu} \subseteq \Rightarrow_{\beta\mu}$, by induction on the derivation of $\Gamma \vdash n \to_\beta t$, where $\Gamma \vdash n \to_{\beta\mu} m(t)$.

For $\Rightarrow_{\beta\mu} \subseteq \to_{\beta\mu}^*$, by induction on the derivation of $\Gamma \vdash n \Rightarrow_{\beta\mu} n'$. We show the case where(the other cases are obvious):

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

By lemma 52, we know that $\Gamma \vdash m([n_2/x]n_1) \Rightarrow_{\beta\mu} m([n_2/x]n_1)$, given $\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1', \Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2'$. Since $\to_{\beta\mu} \subseteq \Rightarrow_{\beta\mu}$, we have: if $\Gamma \vdash n_1 \to_{\beta\mu} n_1', \Gamma \vdash n_2 \to_{\beta\mu} n_2'$, then $\Gamma \vdash m([n_2/x]n_1) \to_{\beta\mu} m([n_2'/x]n_1')(\dagger)$. By IH, we have $\Gamma \vdash n_1 \to_{\beta\mu}^* n_1', \Gamma \vdash n_2 \to_{\beta\mu}^* n_2'$. By lemma 47, lemma 48 and (†), we have $\Gamma \vdash (\lambda x.n_1)n_2 \to_{\beta\mu} m([n_2/x]n_1) \to_{\beta\mu}^* m([n_2'/x]n_1')$. $\square$

Lemma 51. *If $\Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2'$, then $\Gamma \vdash m([n_2/x]n_1) \Rightarrow_{\beta\mu} m([n_2'/x]n_1)$.*

Proof. By induction on the structure of n_1.

Base Cases: $n_1 = x, n_1 = \mu x_i, n_1 = *$. Obvious.

Step Case: $n_1 = \lambda y.n$. We have $\Gamma \vdash m(\lambda y.[n_2/x]n) \equiv \lambda y.m([n_2/x]n) \stackrel{IH}{\Rightarrow}_{\beta\mu} \lambda y.m([n_2'/x]n) \equiv m(\lambda y.[n_2'/x]n)$.

Step Case: $n_1 = nn'$. We have $\Gamma \vdash m([n_2/x]n[n_2/x]n') \equiv m([n_2/x]n)m([n_2/x]n') \stackrel{IH}{\Rightarrow}_{\beta\mu} m([n_2'/x]n)m([n_2'/x]n') \equiv m([n_2'/x]n[n_2'/x]n)$.

Step Case: $n_1 = \iota x.n, \Pi x : n.n'$. Similar as above. $\square$

Lemma 52. *If $\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1'$ and $\Gamma \vdash n_2 \Rightarrow_{\beta\mu} n_2'$, then $\Gamma \vdash m([n_2/y]n_1) \Rightarrow_{\beta\mu} m([n_2'/y]n_1')$.*

Proof. We prove this by induction on the derivation of $\Gamma \vdash n_1 \Rightarrow_{\beta\mu} n_1'$.

Base Case:

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

By lemma 51.

Base Case:

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

Since $y \notin \mathrm{FV}(\mu x_i)$ and μ is local, $m([n_2/y]\mu x_i) \equiv m(\mu x_i)$, then $m(\mu x_1) \equiv \mu x_i \Rightarrow_{\beta\mu} m(\mu t_i) \equiv m(m(\mu t_i))$(lemma 44).

Base Case:

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

In this case, we assume $x \not\equiv y$, then we have $m([n_2/y]x) \equiv m(x) \equiv x \Rightarrow_{\beta\mu} m(t) \equiv m(m(t)) \equiv m([n_2/y]m(t))$.

Step Case:

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

We have $\Gamma \vdash m((\lambda x.[n_2/y]n_a)[n_2/y]n_b) \equiv$
$(\lambda x.m([n_2/y]n_a))m([n_2/y]n_b) \stackrel{IH}{\Rightarrow}_{\beta\mu}$
$m([m([n_2'/y]n_b')/x]m([n_2'/y]n_a')) \equiv m([n_2'/y]([n_b'/x]n_a'))$. The last equality is by lemma 44. 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 $\Gamma \vdash m(\lambda x.[n_2/y]n) \equiv \lambda x.m([n_2/y]n) \stackrel{IH}{\Rightarrow}_{\beta\mu}$
$\lambda x.m([n_2'/y]n') \equiv m(\lambda x.[n_2'/y]n')$

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 $\Gamma \vdash m([n_2/y]n_a[n_2/y]n_b) \equiv m([n_2/y]n_a)m([n_2/y]n_b)$
$\stackrel{IH}{\Rightarrow}_{\beta\mu} m([n_2'/y]n_a')m([n_2'/y]n_b') \equiv m([n_2'/y](n_a' n_b'))$.

The other cases are similar as above. $\square$

Lemma 53 (Diamond Property)**.** *If $\Gamma \vdash n \Rightarrow_{\beta\mu} n'$ and $\Gamma \vdash n \Rightarrow_{\beta\mu} n''$, then there exist n''' such that $\Gamma \vdash n'' \Rightarrow_{\beta\mu} n'''$ and $\Gamma \vdash n' \Rightarrow_{\beta\mu} n'''$.*

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

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

Obvious.

Base Case:

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

Obvious.

Base Case:

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

Obvious.

Step Case:

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

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 52, $\Gamma \vdash m([n_1'/x]n_2') \Rightarrow_{\beta\mu} m([n_1'''/x]n_2''')$, also $\Gamma \vdash (\lambda x.n_1'')n_2'' \Rightarrow_{\beta\mu} m([n_1'''/x]n_2''')$.

Suppose $\Gamma \vdash (\lambda x.n_1)n_2 \Rightarrow_{\beta\mu} m([n_2''/x]n_1'')$, 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 52, $\Gamma \vdash m([n_1'/x]n_2') \Rightarrow_{\beta\mu} m([n_1'''/x]n_2''')$ and $\Gamma \vdash m([n_1''/x]n_2'') \Rightarrow_{\beta\mu} m([n_1'''/x]n_2''')$.

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

Theorem 54. *$\to_\beta \cup \to_\mu$ is confluent.*

Proof. We know by diamond property of $\Rightarrow_{\beta\mu}$, $\to_{\beta\mu}$ is confluent. Since $\to_\mu$ is strongly normalizing and confluent, and by lemma

22 and Hardin's interpretation lemma(lemma 17), we conclude $\to_\beta \cup \to_\mu$ is confluent. □

D. Proofs of Section 3.2

Lemma 55. *Let $\to$ denote $\to_\beta \cup \to_\mu$, if $\Gamma \vdash t \to t'$, then $\Gamma \vdash [t_1/x]t \to [t_1/x]t'$ for any t_1.*

Proof. Obvious. □

Lemma 56. *Let $\to$ denote $\to_\beta \cup \to_\mu$, then $\to$ commutes with $\to_\iota$. i.e. if $\Gamma \vdash t_1 \to t_2$ and $\Gamma \vdash t_1 \to_\iota t_3$, then there exist t_4 such that $\Gamma \vdash t_2 \to_\iota t_4$ and $\Gamma \vdash t_3 \to t_4$.*

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 $\Gamma \vdash t_1 \equiv \iota x.t' \to t_2$. By inversion, we know that $t_2 \equiv \iota x.t''$ with $\Gamma \vdash t' \to t''$. By lemma 55, we know that $\Gamma \vdash [t/x]t' \to [t/x]t''$. Thus $t_4 \equiv [t/x]t''$ and $\Gamma \vdash \iota x.t'' \to_\iota [t/x]t''$. □

Theorem 57. *$\to \cup \to_\iota$ is confluent.*

Lemma 58. *If $\Gamma \vdash t_1 \to_o t_2$, then $\Gamma \vdash [t/x]t_1 \to_o [t/x]t_2$.*

Proof. By induction on derivaton. □

Lemma 59. *If $\Gamma \vdash t_1 \to_o t_2$, then $\Gamma \vdash [t_1/x]t \hookrightarrow_o [t_2/x]t$.*

Proof. By induction on the structure of t. □

Lemma 60. *$\to_o$ has diamond property, thus is confluent.*

Proof. Straightforward induction. □

Lemma 61. *$\to_o$ 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 58, 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 62. *$\to_o$ 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 58, 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 59, 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 by induction. □

Lemma 63. *$\to_o$ weak commutes with $\to_\mu$. i.e. if $\Gamma \vdash t \to_o t'$ and $\Gamma \vdash t \to_\mu t''$, then there exist a t_1 such that $\Gamma \vdash t'' \to_o^* t_1$ and $\Gamma \vdash t' \hookrightarrow_\mu t_1$.*

Proof. By induction on $\Gamma \vdash t \to_o t'$.

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

Suppose $\Gamma \vdash \mu t \to_\mu t$ with $dom(\mu)\#\mathrm{FV}(t)$. 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 $\Gamma \vdash (\mu t_2)(\mu t_3) \to_o^* t_2 t_3$.

For $t \equiv \iota x.t_2, \Pi x : t_2.t_3$, we can argue similarly.

The other cases are by induction. □

Theorem 64. *$\to_o \cup \to_\iota \cup \to_\beta \cup \to_\mu$ is confluent.*

E. Proofs of Section 3.3

Note: In this section we use $\stackrel{t}{=}_{\beta,\mu,\iota,o}$ to mean the same thing as $=_{\beta,\mu,\iota,o}$, but with an emphasis on the subject t.

Lemma 65. *If $\Gamma \vdash t_1 \stackrel{t}{=}_{\beta,\mu,\iota,o} t_2$ and $\Gamma \vdash t : t_1$ and $\Gamma \vdash t_2 : *$, then $\Gamma \vdash t : t_2$.*

Proof. By induction on length of $\Gamma \vdash t_1 \stackrel{t}{=}_{\beta,\mu,\iota,o} t_2$. □

Lemma 66. *If $\Gamma \vdash t_1 \stackrel{t}{=}_{\beta,\mu,\iota,o} t_2$ and $\Gamma \vdash t = t'$, then $\Gamma \vdash t_1 \stackrel{t'}{=}_{\beta,\mu,\iota,o} t_2$.*

Proof. By induction on length of $\Gamma \vdash t_1 \stackrel{t}{=}_{\beta,\mu,\iota,o} t_2$. □

Lemma 67. *$m(\mu_1\mu_2 t) \equiv m(\mu_2\mu_1 t)$, thus $\Gamma \vdash \mu_1\mu_2 t = \mu_2\mu_1 t$.*

Proof. Identify t as $\vec{\mu}t'$, where t' does not have any closure at head position. By induction on the structure of such t'. Also $\Gamma \vdash \mu_1\mu_2 t = m(\mu_1\mu_2 t) = m(\mu_2\mu_1 t) = \mu_2\mu_1 t$. □

Lemma 68. $\Gamma \vdash \mu([t/x]t') = [\mu t/x]\mu t'$

Proof. $\Gamma \vdash \mu([t/x]t') = m(\mu([t/x]t')) = m([\mu t/x]\mu t') = [\mu t/x]\mu t'$. □

Lemma 69. *If $\Gamma, \tilde{\mu} \vdash t' \stackrel{t}{=}_{\beta,\mu,\iota,o} t''$, then $\Gamma \vdash \mu t' \stackrel{\mu t}{=}_{\beta,\mu,\iota,o} \mu t''$*

Proof. By induction on length of $\Gamma, \tilde{\mu} \vdash t' \stackrel{t}{=}_{\beta,\mu,\iota,o} t''$. We list a few cases.
Case: $\Gamma, \tilde{\mu} \vdash t' = t''$.

We have $\Gamma \vdash \mu t' = \mu t''$.

Case: $\Gamma, \tilde{\mu} \vdash \iota x.t' \to_\iota [t/x]t'$.

We know $\Gamma \vdash \mu \iota x.t' = \iota x.\mu t' \to_\iota [\mu t/x]\mu t' = \mu[t/x]t'$ (the last equality is by lemma 68). □

Lemma 70 (Inversion I). *If $\Gamma \vdash \lambda x.t : t'$, then $\Gamma, x : t_1 \vdash t : t_2$ and $\Gamma \vdash \Pi x : t_1.t_2 \stackrel{\lambda x.t}{=}_{\beta,\mu,\iota,o} t'$.*

Proof. By induction on the derivation of $\Gamma \vdash \lambda x.t : t'$. □

Lemma 71 (Inversion II). *If $\Gamma \vdash t_1 t_2 : t'$, then $\Gamma \vdash t_1 : \Pi x : t_1'.t_2'$ and $\Gamma \vdash t_2 : t_1'$, $\Gamma \vdash [t_2/x]t_2' \stackrel{t_1 t_2}{=}_{\beta,\mu,\iota,o} t'$.*

Lemma 72 (Inversion III)**.** *If* $\Gamma \vdash * : t$, *then* $\Gamma \vdash * \stackrel{*}{=}_{\beta,\mu,\iota,o} t$.

Lemma 73 (Inversion IV)**.** *If* $\Gamma \vdash x : t'$, *then* $x : t \in \Gamma$ *and* $\Gamma \vdash t \stackrel{x}{=}_{\beta,\mu,\iota,o} t'$.

Lemma 74 (Inversion V)**.** *If* $\Gamma, \tilde{\mu} \vdash x_j : t'$ *and* $x_j \in dom(\mu)$, *then* $x_j : a_j \in \mu$ *and* $\Gamma, \tilde{\mu} \vdash a_j \stackrel{x_j}{=}_{\beta,\mu,\iota,o} t'$.

Lemma 75 (Inversion VI)**.** *If* $\Gamma \vdash \vec{\mu}t : t'$ *and* t *does not have a closure at head position, then* $\Gamma, \tilde{\vec{\mu}} \vdash t : t''$ *and* $\Gamma \vdash \vec{\mu}t'' \stackrel{\vec{\mu}t}{=}_{\beta,\mu,\iota,o} t'$.

Lemma 76 (Inversion VII)**.** *If* $\Gamma \vdash \iota x.t : t'$, *then* $\Gamma, x : \iota x.t \vdash t : *$ *and* $\Gamma \vdash * \stackrel{\iota x.t}{=}_{\beta,\mu,\iota,o} t'$.

Lemma 77 (Inversion VIII)**.** *If* $\Gamma \vdash \Pi x : t_1.t_2 : t'$, *then* $\Gamma, x : t_1 \vdash t_2 : *$ *and* $\Gamma \vdash t_1 : *$ *and* $\Gamma \vdash * \stackrel{\Pi x:t_1.t_2}{=}_{\beta,\mu,\iota,o} t'$.

Lemma 78. *If* $\Gamma, \tilde{\mu}, y : b \vdash t : a$, *then* $\Gamma, y : \mu b, \tilde{\mu} \vdash t : a$.

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

Lemma 79 (Substitution)**.** *If* $\Gamma_1, x : t_1, \Gamma_2 \vdash t : t_2$ *and* $\Gamma \vdash t' : t_1$, *then* $\Gamma_1, [t'/x]\Gamma_2 \vdash [t'/x]t : [t'/x]t_2$.

Proof. By induction on the derivation of $\Gamma_1, x : t_1, \Gamma_2 \vdash t : t_2$. We will show a few nontrivial cases.

Case:

$$\frac{\Gamma, y : \iota y.t \vdash t : *}{\Gamma \vdash \iota y.t : *}$$

Let $\Gamma = \Gamma_1, x : t_1, \Gamma_2$. We want to show $\Gamma_1, [t'/x]\Gamma_2 \vdash \iota y.[t'/x]t : *$. By IH, we have $\Gamma_1, [t'/x]\Gamma_2, y : \iota y.[t'/x]t \vdash [t'/x]t : *$. So it is the case.

Case:

$$\frac{\Gamma \vdash t : [t/y]t'' \quad \Gamma \vdash \iota y.t'' : *}{\Gamma \vdash t : \iota y.t''}$$

Let $\Gamma = \Gamma_1, x : t_1, \Gamma_2$. We want to show $\Gamma_1, [t'/x]\Gamma_2 \vdash [t'/x]t : \iota y.[t'/x]t''$. By IH, we have $\Gamma_1, [t'/x]\Gamma_2 \vdash [t'/x]t : [[t'/x]t/y]([t'/x]t'')$. So it is the case.

Case:

$$\frac{\Gamma \vdash t : \iota y.t''}{\Gamma \vdash t : [t/y]t''}$$

Let $\Gamma = \Gamma_1, x : t_1, \Gamma_2$. We want to show $\Gamma_1, [t'/x]\Gamma_2 \vdash [t'/x]t : [[t'/x]t/y]([t'/x]t'')$. By IH, we have $\Gamma_1, [t'/x]\Gamma_2 \vdash [t'/x]t : \iota y.[t'/x]t''$. So it is the case.

Case:

$$\frac{\Gamma, \tilde{\mu} \vdash t : t'' \quad \{\Gamma, \tilde{\mu} \vdash t_j : a_j\}_{(t_j:a_j)\in\tilde{\mu}}}{\Gamma \vdash \mu t : \mu t''}\ \textit{Mu}$$

Let $\Gamma = \Gamma_1, x : t_1, \Gamma_2$. We want to show $\Gamma_1, [t'/x]\Gamma_2 \vdash \mu[t'/x]t : \mu[t'/x]t''$. By IH, we have $\Gamma_1, [t'/x]\Gamma_2, [t'/x]\tilde{\mu} \vdash [t'/x]t : [t'/x]t''$ and $\{\Gamma_1, [t'/x]\Gamma_2, [t'/x]\tilde{\mu} \vdash t_j : [t'/x]a_j\}_{(t_j:[t'/x]a_j)\in[t'/x]\tilde{\mu}}$. $\square$

Theorem 80 (Type Preservation)**.** *If* $\Gamma \vdash \mathsf{wf}$ *and* $\Gamma \vdash t \rightsquigarrow t'$ *and* $\Gamma \vdash t : a$, *then* $\Gamma \vdash t' : a$.

Proof. By induction on the derivation of $\Gamma \vdash t : a$, We list a few nontrivial cases.

Case:

$$\frac{}{\Gamma \vdash * : *}$$

This case will not arise.

Case:

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

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

Case:

$$\frac{\Gamma \vdash t : t_1 \quad \Gamma \vdash t_1 \cong t_2 \quad \Gamma \vdash t_2 : *}{\Gamma \vdash t : t_2}$$

In this case $\Gamma \vdash t \rightsquigarrow t'$. By IH, $\Gamma \vdash t' : t_1$. Since $\Gamma \vdash t_1 \cong t_2$, we have $\Gamma \vdash t' : t_2$.

Case:

$$\frac{\Gamma \vdash t : \iota x.t''}{\Gamma \vdash t : [t/x]t''}$$

In this case $\Gamma \vdash t \rightsquigarrow t'$. By IH, $\Gamma \vdash t' : \iota x.t''$. Thus we have $\Gamma \vdash t' : [t'/x]t''$. Since $\Gamma \vdash t' = t$, we have $\Gamma \vdash t' : [t/x]t''$ by Conv rule.

Case:

$$\frac{\Gamma \vdash t : [t/x]t'' \quad \Gamma \vdash \iota x.t'' : *}{\Gamma \vdash t : \iota x.t''}$$

In this case $\Gamma \vdash t \rightsquigarrow t'$. By IH, $\Gamma \vdash t' : [t/x]t''$. Since $\Gamma \vdash [t/x]t'' = [t'/x]t''$, we have $\Gamma \vdash t' : [t'/x]t''$.Thus we have $\Gamma \vdash t' : \iota x.t''$.

Case:

$$\frac{\Gamma \vdash t_1' : \Pi x : t_1''.t_2'' \quad \Gamma \vdash t_2' : t_1''}{\Gamma \vdash t_1' t_2' : [t_2'/x]t_2''}$$

Suppose $\Gamma \vdash (\lambda x.t_1)v \rightsquigarrow [v/x]t_1$. Then we know $\Gamma \vdash (\lambda x.t_1)v : [v/x]t_2''$ and $\Gamma \vdash \lambda x.t_1 : \Pi x : t_1''.t_2''$ and $\Gamma \vdash v : t_1''$. By inversion on $\Gamma \vdash \lambda x.t_1 : \Pi x : t_1''.t_2''$, we have $\Gamma, x : a \vdash t_1 : b$ and $\Gamma \vdash \Pi x : a.b \stackrel{\lambda x.t_1}{=}_{\beta,\mu,\iota,o} \Pi x : t_1''.t_2''$. By theorem 34, we have $\Gamma \vdash a =_{\beta,\mu,o} t_1''$ and $\Gamma \vdash b =_{\beta,\mu,o} t_2''$. So we have $\Gamma, x : a \vdash t_1 : t_2''$ and $\Gamma \vdash v : a$. So by lemma 79, we have $\Gamma \vdash [v/x]t_1 : [v/x]t_2''$, as required.

Suppose $\Gamma \vdash t_1 t_2 \rightsquigarrow t_1' t_2$, where $\Gamma \vdash t_1 \rightsquigarrow t_1'$. We know $\Gamma \vdash t_1 t_2 : [t_2/x]t_2''$ and $\Gamma \vdash t_1 : \Pi x : t_1''.t_2''$ and $\Gamma \vdash t_2 : t_1''$. By IH, we know $\Gamma \vdash t_1' : \Pi x : t_1''.t_2''$. So $\Gamma \vdash t_1' t_2 : [t_2/x]t_2''$.

Suppose $\Gamma \vdash (\lambda x.t_1)t_2 \rightsquigarrow (\lambda x.t_1)t_2'$, where $\Gamma \vdash t_2 \rightsquigarrow t_2'$. We know $\Gamma \vdash (\lambda x.t_1)t_2 : [t_2/x]t_2''$ and $\Gamma \vdash \lambda x.t_1 : \Pi x : t_1''.t_2''$ and $\Gamma \vdash t_2 : t_1''$. By IH, we know $\Gamma \vdash t_2' : t_1''$. So $\Gamma \vdash (\lambda x.t_1)t_2' : [t_2'/x]t_2''$. And we know $\Gamma \vdash [t_2/x]t_2'' = [t_2'/x]t_2''$.

Case:

$$\frac{\Gamma, \tilde{\mu} \vdash t : t' \quad \{\Gamma, \tilde{\mu} \vdash t_j : a_j\}_{(t_j : a_j) \in \tilde{\mu}}}{\Gamma \vdash \mu t : \mu t'}$$

Suppose $\Gamma \vdash \mu x_j \rightsquigarrow \mu t_j$, where $x_j \mapsto t_j \in \mu$. We have $\Gamma, \tilde{\mu} \vdash x_j : t'$. By inversion, $\Gamma, \tilde{\mu} \vdash x_j : a_j$ and $\Gamma, \tilde{\mu} \vdash a_j \stackrel{x_j}{=}_{\beta,\mu,\iota,o} t'$. Since $\Gamma, \tilde{\mu} \vdash x_j = t_j$ and by lemma 66, we get $\Gamma, \tilde{\mu} \vdash a_j \stackrel{t_j}{=}_{\beta,\mu,\iota,o} t'$. Since $\Gamma, \tilde{\mu} \vdash t_j : a_j$, by lemma 65, $\Gamma, \tilde{\mu} \vdash t_j : t'$. Thus we have $\Gamma \vdash \mu t_j : \mu t'$.

Suppose $\Gamma \vdash \mu\vec{\mu}x_j \rightsquigarrow \mu\vec{\mu}t_j$, where $x_j \mapsto t_j \in \mu_j$. By inversion on $\Gamma, \tilde{\mu} \vdash \vec{\mu}x_j : t'$, we have $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash x_j : t_a$ and $\Gamma, \tilde{\mu} \vdash \vec{\mu}t_a \stackrel{\vec{\mu}x_j}{=}_{\beta,\mu,\iota,o} t'$. By inversion on $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash x_j : t_a$, we have $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash x_j : b$, where $(x_j : b) \in \tilde{\mu} \cup \tilde{\vec{\mu}}$ and $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash b \stackrel{x_j}{=}_{\beta,\mu,\iota,o} t_a$. So $\Gamma \vdash \mu\vec{\mu}b \stackrel{\mu\vec{\mu}x_j}{=}_{\beta,\mu,\iota,o} \mu\vec{\mu}t_a \stackrel{\mu\vec{\mu}x_j}{=}_{\beta,\mu,\iota,o} \mu t'$. Since $\Gamma \vdash \mu\vec{\mu}t_j : \mu\vec{\mu}b$ and $\Gamma \vdash \mu\vec{\mu}x_j = \mu\vec{\mu}t_j$, so $\Gamma \vdash \mu\vec{\mu}t_j : \mu t'$.

Suppose $\Gamma \vdash \mu* \rightsquigarrow *$. We have $\Gamma, \tilde{\mu} \vdash * : t''$. We have $\Gamma, \tilde{\mu} \vdash * \stackrel{*}{=}_{\beta,\mu,\iota,o} t''$(by inversion). Thus we have $\Gamma \vdash \mu* \stackrel{\mu*}{=}_{\beta,\mu,\iota,o} \mu t''$(lemma 69). We also know that $\Gamma \vdash * : *$ and $\Gamma \vdash \mu* = *$. So we have $\Gamma \vdash * \stackrel{\mu*}{=}_{\beta,\mu,\iota,o} \mu t''$. Thus $\Gamma \vdash * \stackrel{*}{=}_{\beta,\mu,\iota,o} \mu t''$. So $\Gamma \vdash * : \mu t''$(lemma 65).

Suppose $\Gamma \vdash \mu\vec{\mu}* \rightsquigarrow *$. We argue similarly.

Suppose $\Gamma \vdash \mu x \rightsquigarrow x$, where $x \notin dom(\mu)$. We have $\Gamma, \tilde{\mu} \vdash x : t''$. We have $\Gamma, \tilde{\mu} \vdash a \stackrel{x}{=}_{\beta,\mu,\iota,o} t''$, where $x : a \in \Gamma$(by inversion). Thus we have $\Gamma \vdash \mu a \stackrel{\mu x}{=}_{\beta,\mu,\iota,o} \mu t''$(lemma 69). We also know that $\Gamma \vdash x : a$ and $\Gamma \vdash \mu x = x$ and $\Gamma \vdash \mu a = a$. Thus $\Gamma \vdash a \stackrel{x}{=}_{\beta,\mu,\iota,o} \mu t''$. So $\Gamma \vdash x : \mu t''$(lemma 65).

Suppose $\Gamma \vdash \mu\vec{\mu}x \rightsquigarrow x$, where $x \notin dom(\mu) \cup dom(\vec{\mu})$. By inversion on $\Gamma, \tilde{\mu} \vdash \vec{\mu}x : t'$, we have $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash x : t_a$, where $\Gamma, \tilde{\mu} \vdash \vec{\mu}t_a \stackrel{\vec{\mu}x}{=}_{\beta,\mu,\iota,o} t'$. By inversion on $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash x : t_a$, we have $x : b \in \Gamma$ and $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash b \stackrel{x}{=}_{\beta,\mu,\iota,o} t_a$. So $\Gamma, \mu \vdash \vec{\mu}b \stackrel{\vec{\mu}x}{=}_{\beta,\mu,\iota,o} \vec{\mu}t_a \stackrel{\vec{\mu}x}{=}_{\beta,\mu,\iota,o} t'$. So $\Gamma \vdash b \stackrel{x}{=}_{\beta,\mu,\iota,o} \mu t'$. Thus $\Gamma \vdash x : \mu t'$.

Suppose $\Gamma \vdash \mu\lambda x.t \rightsquigarrow \lambda x.\mu t$. We have $\Gamma, \tilde{\mu} \vdash \lambda x.t : t''$ and $\Gamma, \tilde{\mu}, x : t_1'' \vdash t : t_2''$ and $\Gamma, \tilde{\mu} \vdash \Pi x : t_1''.t_2'' \stackrel{\lambda x.t}{=}_{\beta,\mu,\iota,o} t''$(by inversion). Thus we have $\Gamma, x : \mu t_1'' \vdash \mu t : \mu t_2''$(lemma 78) and $\Gamma \vdash \mu(\Pi x : t_1''.t_2'') \stackrel{\mu\lambda x.t}{=}_{\beta,\mu,\iota,o} \mu t''$(lemma 69). By lemma 66, $\Gamma \vdash (\Pi x : \mu t_1''.\mu t_2'') \stackrel{\lambda x.\mu t}{=}_{\beta,\mu,\iota,o} \mu t''$. Also, $\Gamma \vdash \lambda x.\mu t : \Pi x : (\mu t_1'').(\mu t_2'')$. So by lemma 65, $\Gamma \vdash \lambda x.\mu t : \mu t''$.

Suppose $\Gamma \vdash \mu\vec{\mu}\lambda x.t \rightsquigarrow \lambda x.\mu\vec{\mu}t$. By inversion on $\Gamma, \tilde{\mu} \vdash \vec{\mu}(\lambda x.t) : t'$, we have $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash \lambda x.t : t_a$, where $\Gamma, \tilde{\mu} \vdash \vec{\mu}t_a \stackrel{\vec{\mu}(\lambda x.t)}{=}_{\beta,\mu,\iota,o} t'$. By inversion on $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash \lambda x.t : t_a$, then we have $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}}, x : t_1'' \vdash t : t_2''$ and $\Gamma, \tilde{\mu}, \tilde{\vec{\mu}} \vdash \Pi x : t_1''.t_2'' \stackrel{\lambda x.t}{=}_{\beta,\mu,\iota,o} t_a$. So $\Gamma, \tilde{\mu} \vdash \Pi x : \vec{\mu}t_1''.\vec{\mu}t_2'' \stackrel{\vec{\mu}(\lambda x.t)}{=}_{\beta,\mu,\iota,o} \vec{\mu}t_a \stackrel{\vec{\mu}(\lambda x.t)}{=}_{\beta,\mu,\iota,o} t'$. Thus $\Gamma \vdash \Pi x : \mu\vec{\mu}t_1''.\mu\vec{\mu}t_2'' \stackrel{\mu\vec{\mu}(\lambda x.t)}{=}_{\beta,\mu,\iota,o} \mu t'$. Since $\Gamma \vdash \lambda x.\mu\vec{\mu}t : \Pi x : \mu\vec{\mu}t_1''.\mu\vec{\mu}t_2''$, we have $\Gamma \vdash \lambda x.\mu\vec{\mu}t : \mu t'$.

Suppose $\Gamma \vdash \mu(t_1't_2') \rightsquigarrow (\mu t_1')(\mu t_2')$. We have $\Gamma, \tilde{\mu} \vdash t_1't_2' : t''$. We have $\Gamma, \tilde{\mu} \vdash t_1' : \Pi x : t_1''.t_2''$ and $\Gamma, \tilde{\mu} \vdash t_2' : t_1''$ and $\Gamma, \tilde{\mu} \vdash [t_2'/x]t_2'' \stackrel{t_1't_2'}{=}_{\beta,\mu,\iota,o} t''$(by inversion). Thus we have $\Gamma \vdash \mu t_1' : \mu(\Pi x : t_1''.t_2'')$ and $\Gamma \vdash \mu t_2' : \mu t_1''$ and $\Gamma \vdash \mu([t_2'/x]t_2'') \stackrel{\mu(t_1't_2')}{=}_{\beta,\mu,\iota,o} \mu t''$(lemma 69). By lemma 66, we have $\Gamma \vdash [\mu t_2'/x]\mu t_2'' \stackrel{(\mu t_1')(\mu t_2')}{=}_{\beta,\mu,\iota,o} \mu t''$. So $\Gamma \vdash (\mu t_1')(\mu t_2') : [\mu t_2'/x]\mu t_2''$ and then $\Gamma \vdash (\mu t_1')(\mu t_2') : \mu t''$(lemma 65).

Suppose $\Gamma \vdash \mu\vec{\mu}(t_1't_2') \rightsquigarrow (\mu\vec{\mu}t_1')(\mu\vec{\mu}t_2')$, we argue similar as the case for $\Gamma \vdash \mu\vec{\mu}\lambda x.t \rightsquigarrow \lambda x.\mu\vec{\mu}t$.

$\square$