Lambda Encoding with Comprehension

Peng Fu
Computer Science, The University of Iowa

October 30, 2013

1 Introduction

In this chapter, we will see iota-binder from a different perspective. Instead of viewing iota-binder as a type construct, we view it as a set-forming construct. So if $F[x]$ is a formula containing a free term variable x , then $\iota x.F[x]$ describes a set of terms t, which satisfies the formula, i.e. $t \in \iota x.F[x]$ iff $F[t]$. Recalled that in $\mathsf{Selfstar}$ and $\mathbf{S}$, we have $\vdash t : \iota x.T$ iff $\vdash t : [t/x]T$. If we compare $t \in \iota x.F[x]$ with $\vdash t : \iota x.T$, we observe that there is a similarity between the meta-level typing relation (denoted by ":") and the set membership notation "$\in$", which lies in the object logic. This observation is inspired from our earlier work on internalization [5]. Right now we are being informal, because it is hard to draw a connection between $F[t]$ and $\vdash t : [t/x]T$, since equating $t \in F[t]$ with $F[t]$ violates the grammatical structure of the logic. Furthermore, one can not naïvely views self type as formula. Suppose both $\iota x.T$ and T are corresponding to formulas, we know that for the formula $F[x]$, the $\iota x.F[x]$ is representing a set, not a formula, so it is again incoherent to equates $\iota x.F[x]$ with $\iota x.T$. Nonetheless, the observation above motivates us to investigate iota-binder from a pure logical perspective.

Another source of inspiration of our work in this Chapter is from Hatcher's formulation of Frege's logic [7]. Hatcher presented Frege's system [4] in modern notations, i.e. a logic with basic set-like construct and comprehension axiom. He showed how to prove all of Peano's axioms in Frege's system. Dispite Frege's system itself is inconsistent, the development of Peano's axioms, especially the derivation of induction principle is remarkable and should be emphasis over the inconsistency. In fact, later we can see the derivation of induction principle follows the spirit of Frege.

We first present Frege's System $\mathfrak{F}$ (section 2), to motivate our construction of arithmetic with lambda calculus. Then we give a formulation of second order theory of lambda calculus based on iota-binder ι and epsilon relation ϵ, we call it $\mathfrak{G}$ (section 3). There are at least three similar systems, namely, Girard's formulation of $\mathbf{HA}_2$ à la Takeuti [6], Krivine's $\mathbf{FA}_2$ [8] and Takeuti's second order logic [11]. There are two major differences between $\mathfrak{G}$ and these systems, the first one is that the domain of individuals of $\mathfrak{G}$ is lambda terms instead of primitive notion of numbers. The second one is that $\mathfrak{G}$ has (ϵ, ι)-notation, namely, set-abstraction and membership relation are explict in the object language. Thus comprehension axiom is needed in $\mathfrak{G}$. While the other systems use predication instead of membership relation, and set-abstraction is implicit at the meta-level, the comprehension axiom is admissible by performing substitution. The second difference is subtle and has significant implications, as we shall see in section 4, we can define a notion of polymorphic-dependent typing within $\mathfrak{G}$, which benefits from the facts that $\mathfrak{G}$ adimits (ϵ, ι)-notation.

We prove all of Peano's axioms in section 5. We enrich the reduction on lambda term with η-and Ω-reductions, then we are able to show that the member of the inductively defined sets such as Nat is terminating with respect to *head beta-reduction* (section 6). Finally, we show the notion of Leibniz equality in $\mathfrak{G}$ is *faithful* to the conversions in lambda calculus (section 6.2).

2 Frege's System $\mathfrak{F}$

Certain inconsistent systems and their corresponding antinomies are invaluable, because not only the antinomies can be served as criterions for maintaining consistency, but also, perhaps more importantly, they gives us examples to see how to reconstruct a large part of mathematics within these systems. Frege's system (à la Hatcher) belongs to this category. In fact, $\mathfrak{G}$ is inspired by the Fregean construction of numbers. We formalize a intuitionistic version of Frege's system $\mathfrak{F}$ in the form of sequent calculus, and then we show how to derive basic arithmetic with $\mathfrak{F}$ and see how the antinomy arises.

Definition 1 (Syntax).
Domain Terms/Set $a, b, s \ ::= \ x \mid \iota x.F$
Formula $F \ ::= \bot \mid s\epsilon s' \mid \ F_1 \to F_2 \mid \forall x.F \mid F \wedge F'$
Context $\Gamma \ ::= \ \cdot \mid \Gamma, F$

We identify four syntactical categories in $\mathfrak{F}$, namely, *domain terms*, *set* and *formula*. Note that *set* in this chapter is just a name for a syntactical category, we should not confused the notion of set in this chapter with the "set" in **ZF**. Also, the notion of set coincides with the notion of domain terms in $\mathfrak{F}$.

Definition 2 (Deduction Rules).

$$\frac{F \in \Gamma}{\Gamma \vdash F} \qquad \frac{\Gamma \vdash F_1 \quad F_1 = F_2}{\Gamma \vdash F_2} \qquad \frac{\Gamma \vdash F \quad x \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash \forall x.F}$$

$$\frac{\Gamma \vdash \forall x.F}{\Gamma \vdash [s/x]F} \qquad \frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \to F_2} \qquad \frac{\Gamma \vdash F_1 \to F_2 \quad \Gamma \vdash F_1}{\Gamma \vdash F_2}$$

$$\frac{\Gamma \vdash F_1 \wedge F_2}{\Gamma \vdash F_i} \qquad \frac{\Gamma \vdash F_1 \quad \Gamma \vdash F_2}{\Gamma \vdash F_1 \wedge F_2}$$

$F_1 = F_2$ is specified by the comprehension axiom.

Definition 3 (Comprehension). $s\epsilon(\iota x.F) = [s/x]F$

Comprehension axiom is essential for Fregean number construction. The definition of number, the induction principle for numbers rely on comprehension. Because the notion domain terms and set coincide, with comprehension axiom, $\mathfrak{F}$ is inconsistent. There are three major ways to avoid the antinomy, namely, Russell's type theory [12], Quine's new foundation [10] and Zermelo's **ZF** [13]. In next section, we present System $\mathfrak{G}$, which separates the notion of domain terms and set, thus is in the spirit of Quine's stratefication.

Definition 4 (Equality). $a = b := \forall z.(z\epsilon a \equiv z\epsilon b)$.

For convenient, we write $a \equiv b$ to denote $a \to b. \wedge .b \to a$. We also write $a \neq b$ for $a = b \to \bot$, $\exists a.A$ for $(\forall a.(A \to \bot)) \to \bot$. Now we can proceed to construct a naïve set theory in $\mathfrak{F}$.

Definition 5 (Naïve Set Theory).
$\Lambda := \iota x.(x = x \to \bot)$.
$\{b\} := \iota y.y = b$
$\bar{c} := \iota y.(y\epsilon c \to \bot)$.
$a \cap b := \iota z.(z\epsilon a \wedge z\epsilon b)$
$a \cup b := \iota z.(z\epsilon a \to \bot. \wedge .z\epsilon b \to \bot) \to \bot$

Theorem 1.
$\vdash \forall x.(x = x)$
$\vdash \forall x.(x\epsilon\Lambda \to \bot)$.

We can take $x\epsilon\Lambda$ as our notion of contradictory because $x\epsilon\Lambda$ implies $\bot$. We now can develop an elementary number theory in $\mathfrak{F}$.

Definition 6 (Fregean Numbers)**.**
$N := \iota x.\forall c.(\forall y.(y\epsilon c \to Sy\epsilon c)) \to 0\epsilon c \to x\epsilon c.$
$0 := \{\Lambda\}.$
$S\ a := \iota y.\exists z.(z\epsilon y. \wedge .(y \cap \overline{\{z\}})\epsilon a).$

Theorem 2.
$\vdash 0\epsilon N.$

Proof. We want to prove $\forall c.(\forall y.(y\epsilon c \to Sy\epsilon c)) \to 0\epsilon c \to 0\epsilon c$. Assume $\forall y.(y\epsilon c \to Sy\epsilon c)$ and $0\epsilon c$, we want to show $0\epsilon c$, which is obvious[1]. □

Theorem 3. $\vdash \forall y.(y\epsilon N \to Sy\epsilon N).$

Proof. Assume $y\epsilon N$, we want to show $Sy\epsilon N$. By comprehension, we want to show $\forall c.(\forall y.(y\epsilon c \to Sy\epsilon c)) \to 0\epsilon c \to (Sy)\epsilon c$. So we assume $\forall y.(y\epsilon c \to Sy\epsilon c)$ and $0\epsilon c$, we need to show $(Sy)\epsilon c$. We know that $y\epsilon N$ implies $\forall c.(\forall y.(y\epsilon c \to Sy\epsilon c)) \to 0\epsilon c \to y\epsilon c$. By modus ponens, we have $y\epsilon c$. By universal instantiation, we have $y\epsilon c \to Sy\epsilon c$. So by modus ponens, we have $Sy\epsilon c$. Thus we have the proof[2]. □

Theorem 4 (Induction Principle)**.** $\vdash \forall c.(\forall y.(y\epsilon c \to Sy\epsilon c)) \to 0\epsilon c \to \forall x.(x\epsilon N \to x\epsilon c).$

Proof. Assume $\forall y.(y\epsilon c \to Sy\epsilon c), 0\epsilon c, x\epsilon N$. We want to show $x\epsilon c$. We know $x\epsilon N$ implies $\forall c.(\forall y.(y\epsilon c \to Sy\epsilon c)) \to 0\epsilon c \to x\epsilon c$. By modus ponens, we get $x\epsilon c$[3]. □

We observe that there is an algorithmic interpretation for constructive proof of totality of certain kind of function. For example, the proof of S is total, namely, $\forall y.(y\epsilon N \to Sy\epsilon N)$, can be encoded as Church numeral's successor $\lambda n.\lambda s.\lambda z.s\ (n\ s\ z)$. This result is already known by Leivant and Krivine [9], [8]. So one should at least admit there is constructive flavor in Fregean construction of number. Of course, the system itself is inconsistent, i.e. the following formula is provable in system $\mathfrak{F}$:

Let $A := (\iota u_1.u_1 \notin u_1)\epsilon(\iota u_1.u_1 \notin u_1) = A \to \bot$. So we have $\vdash A \to \bot$ since $A \vdash A$ and $A \vdash A \to \bot$. Also, $A \to \bot \vdash A \to \bot$ implies $A \to \bot \vdash A$, thus $\vdash (A \to \bot) \to \bot$. By modus ponens, we can derive $\vdash \bot$. It is worthnoting that intuitionistic is irrelavant to prevent inconsistency.

However, as we repeatly emphasis, it is the constructive spirit that we should take upon ourself, the antinomy only serves as a criterion of the boundary.

3 System $\mathfrak{G}$

System $\mathfrak{G}$ is inspired by Frege's $\mathfrak{F}$ and the possibility of understanding the iota-binder as set-abstraction in higher order logic. System $\mathfrak{G}$ is a *simple* logical system with the (ϵ, ι)-notation.

Definition 7.
Formula $F\ ::=\ X^0 \mid t\epsilon S \mid \Pi X^1.F \mid\ F_1 \to F_2 \mid \forall x.F \mid \Pi X^0.F$
Set $S\ ::= X^1 \mid \iota x.F$
Domain Terms/Pure Lambda Terms $t\ ::=\ x \mid \lambda x.t \mid tt'$
Context $\Gamma\ ::=\ \cdot \mid \Gamma, F$

X^0 is a formula variable, it represents any formula. X^1 is a set variable, it represents any set. $\iota x.F$ is the set formed by the formula F. Unlike $\mathfrak{F}$, we separate the notion of set and domain terms, the domain terms in $\mathfrak{G}$ are pure lambda terms. Set can only occur inside of a formula, they do not have their own rule and identity outside of a formula. Again, please do not confuse the set in this chapter with the set in **ZF**. $\Pi X^0.F$ is a formula formed by quantifying over formula and $\Pi X^1.F$ is formed by quantifying over set. The notation of ϵ, ι are formal parts of the language of $\mathfrak{G}$, they are called (ϵ, ι)-notation.

[1]Observe that the lambda term for the proof is Church numberal zero $\lambda s.\lambda z.z$.
[2]The lambda term for this proof is Church successor $\lambda n.\lambda s.\lambda z.s(n\ s\ z)$.
[3]The lambda term for this proof is iterator $\lambda f.\lambda a.\lambda n.n\ f\ a$.

Definition 8 (Deduction Rules)**.**

$$\frac{F \in \Gamma}{\Gamma \vdash F} \qquad \frac{\Gamma \vdash F_1 \quad F_1 =_{\beta,\iota} F_2}{\Gamma \vdash F_2}\ \textit{Conv} \qquad \frac{\Gamma \vdash F \quad x \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash \forall x.F}$$

$$\frac{\Gamma \vdash \forall x.F}{\Gamma \vdash [t/x]F} \qquad \frac{\Gamma \vdash F \quad X^i \notin \mathrm{FV}(\Gamma) \quad i = 0,1}{\Gamma \vdash \Pi X^i.F} \qquad \frac{\Gamma \vdash \Pi X^0.F}{\Gamma \vdash [F'/X^0]F}\ \textit{Inst0}$$

$$\frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \to F_2} \qquad \frac{\Gamma \vdash F_1 \to F_2 \quad \Gamma \vdash F_1}{\Gamma \vdash F_2} \qquad \frac{\Gamma \vdash \Pi X^1.F}{\Gamma \vdash [S/X^1]F}\ \textit{Inst1}$$

The rule *Inst0* allows us to instantiate X^0 with any formula, this is what the instantiation does in system **F**, while the *Inst1* rule allows us to instantiate a set variable X^1 with any set S.

Definition 9 (Axioms)**.** *$F_1 =_{\iota,\beta} F_2$ iff one the the following holds.*

1. *F_1 (or F_2) is of the form $t\epsilon(\iota x.F)$ and F_2 (or F_1) is of the form $[t/x]F$.*
2. *F_1 (or F_2) contains a term t and F_2 (or F_1) is obtained from F_1 by replacing t with its beta-equivalent term t'.*

The first axiom corresponds to the comprehension axiom. The second axiom corresponds to the axiom of extensionality [7], it also depends on beta-conversion axiom in lambda calculus. We know that beta-conversion in lambda calculus is Church-Rosser, thus not every lambda terms are considered equal. The reason we set up axioms through the *Conv* rule is that it will not affect the overall proof tree, a direct consequence is that then consistency is easy to prove, as we shall see next.

3.1 Consistency of System $\mathfrak{G}$

We have presented the whole specifications of $\mathfrak{G}$. Now we show $\mathfrak{G}$ is consistent, in the sense that not every formula is provable in $\mathfrak{G}$. To prove consistency, we will first devise a version of $\mathfrak{G}$ with proof term annotation, denoted by $\mathfrak{G}[p]$. Then a forgetful mapping from $\mathfrak{G}[p]$ to System **F** is defined. Finally, any deravable judgement in $\mathfrak{G}[p]$ can be mapped to a deravable judgement in System **F**. Thus we can conclude the proof term for $\mathfrak{G}[p]$ is strongly normalizing and not every formula in $\mathfrak{G}$ is provable.

Definition 10 (System $\mathfrak{G}[p]$)**.**
Proof Terms $p \ ::= \ a \mid \lambda a.p \mid pp'$
Proof Context $\Gamma \ ::= \ \cdot \mid a : F, \Gamma$

$$\frac{\Gamma \vdash p : F \quad x \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash p : \forall x.F} \qquad \frac{\Gamma \vdash p : F_1 \quad F_1 =_{\beta,\iota} F_2}{\Gamma \vdash p : F_2} \qquad \frac{(a : F) \in \Gamma}{\Gamma \vdash a : F}$$

$$\frac{\Gamma \vdash p : \forall x.F}{\Gamma \vdash p : [t'/x]F} \qquad \frac{\Gamma \vdash p : F \quad X^i \notin \mathrm{FV}(\Gamma) \quad i = 0,1}{\Gamma \vdash p : \Pi X^i.F} \qquad \frac{\Gamma \vdash p : \Pi X^0.F}{\Gamma \vdash p : [F'/X^0]F}$$

$$\frac{\Gamma, a : F_1 \vdash p : F_2}{\Gamma \vdash \lambda a.p : F_1 \to F_2} \qquad \frac{\Gamma \vdash p : F_1 \to F_2 \quad \Gamma \vdash p' : F_1}{\Gamma \vdash pp' : F_2} \qquad \frac{\Gamma \vdash p : \Pi X^1.F}{\Gamma \vdash p : [S/X^1]F}$$

The proof terms only annotated the introduction and elimination rules of implication. We now define the mapping to System **F**.

Definition 11.

$\phi(X^0) := X$	$\phi(t\epsilon S) := \phi(S)$
$\phi(F_1 \to F_2) := \phi(F_1) \to \phi(F_2)$	$\phi(\Pi X^0.F) := \Pi X.\phi(F)$
$\phi(\Pi X^1.F) := \Pi X.\phi(F)$	$\phi(\forall x.F) := \phi(F)$
$\phi(X^1) := X$	$\phi(\iota x.F) := \phi(F)$

Note that the function ϕ can be easily extended to the proof context. It maps formula and set in $\mathfrak{G}[p]$ to types in System **F**.

Lemma 1.

1. *If* $F_1 =_{\beta,\iota} F_2$*, then* $\phi(F_1) \equiv \phi(F_2)$.
2. $\phi(F) \equiv \phi([t'/x]F)$.
3. $\phi([F'/X^0]F) \equiv [\phi(F')/X]\phi(F)$.
4. $\phi([S/X^1]F) \equiv [\phi(S)/X]\phi(F)$.

The following theorem connects System $\mathfrak{G}[p]$ with System **F**.

Theorem 5. *If* $\Gamma \vdash p : F$ *in* $\mathfrak{G}[p]$*, then* $\phi(\Gamma) \vdash p : \phi(F)$ *in* **F**.

Proof. By induction on the derivation of $\Gamma \vdash p : F$.
Case:

$$\frac{(a : F) \in \Gamma}{\Gamma \vdash a : F}$$

By $a : \phi(F) \in \phi(\Gamma)$.
Case:

$$\frac{\Gamma \vdash p : F \quad x \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash p : \forall x.F}$$

By IH, we know that $\phi(\Gamma) \vdash p : \phi(F) \equiv \phi(\forall x.F)$.
Case:

$$\frac{\Gamma \vdash p : F_1 \quad F_1 =_{\beta,\iota} F_2}{\Gamma \vdash p : F_2}$$

By lemma 1, we know that $\phi(F_1) \equiv \phi(F_2)$.
Case:

$$\frac{\Gamma \vdash p : \forall x.F}{\Gamma \vdash p : [t'/x]F}$$

By lemma 1, we know that $\phi(\forall x.F) \equiv \phi(F) \equiv \phi([t'/x]F)$.
Case:

$$\frac{\Gamma \vdash p : F \quad X^i \notin \mathrm{FV}(\Gamma) \quad i = 0, 1}{\Gamma \vdash p : \Pi X^i.F}$$

By IH, we know $\phi(\Gamma) \vdash p : \phi(F)$. And $X \notin \mathrm{FV}(\phi(\Gamma))$, thus $\phi(\Gamma) \vdash p : \Pi X.\phi(F) \equiv \phi(\Pi^i X.F)$.
Case:

$$\frac{\Gamma \vdash p : \Pi X^0.F}{\Gamma \vdash p : [F'/X^0]F}$$

By IH, we know that $\phi(\Gamma) \vdash p : \Pi X.\phi(F)$. Thus $\phi(\Gamma) \vdash p : [\phi(F')/X]\phi(F) \equiv \phi([F'/X^0]F)$. The last equality is by lemma 1.
Case:

$$\frac{\Gamma, a : F_1 \vdash p : F_2}{\Gamma \vdash \lambda a.p : F_1 \to F_2}$$
By IH, we know $\phi(\Gamma), a : \phi(F_1) \vdash p : \phi(F_2)$. Thus $\phi(\Gamma) \vdash \lambda a.p : \phi(F_1) \to \phi(F_2)$.
Case:

$$\frac{\Gamma \vdash p : F_1 \to F_2 \quad \Gamma \vdash p' : F_1}{\Gamma \vdash pp' : F_2}$$
By IH, $\phi(\Gamma) \vdash p : \phi(F_1) \to \phi(F_2)$ and $\phi(\Gamma) \vdash p' : \phi(F_1)$. Thus $\phi(\Gamma) \vdash pp' : \phi(F_2)$.
Case:

$$\frac{\Gamma \vdash p : \Pi X^1.F}{\Gamma \vdash p : [S/X^1]F}$$
By IH, we know that $\phi(\Gamma) \vdash p : \Pi X.\phi(F)$. Thus $\phi(\Gamma) \vdash p : [\phi(S)/X]\phi(F) \equiv \phi([S/X^1]F)$. The last equality is by lemma 1.

□

Theorem 5 implies that if $\Gamma \vdash p : F$ in $\mathfrak{G}[p]$, then p is strongly normalizing and that the formlua $\Pi X^0.X$ in $\mathfrak{G}$ is unprovable.

3.2 Preservation Theorem for $\mathfrak{G}[p]$

We need to establish preservation property for $\mathfrak{G}[p]$ in order to explore more unprovable formulas in $\mathfrak{G}$. The proof of preservation theorem is an adaption of Barendregt's method for proving preservation for System **F** à la Curry [2].

Definition 12 (Formula Reduction)**.**

- *$F_1 \to_\beta F_2$ if $t_1 =_\beta t_2$, $F_1 \equiv F[t_1]$ and $F_2 \equiv F[t_2]$.*
- *$F_1 \to_\iota F_2$ if $F_1 \equiv t \epsilon \iota x.F$ and $F_2 \equiv [t/x]F$.*

Note that $F[t_1]$ means the lambda term t_1 appears inside the formula F and $\to_{\beta,\iota}$ denotes $\to_\beta \cup \to_\iota$.

Lemma 2. *$\to_{\beta,\iota}$ is confluent.*

Proof. We know that $\to_\beta$ and $\to_\iota$ are confluent. We also know that $\to_\beta$ commutes with $\to_\iota$, so $\to_{\beta,\iota}$ is confluent. □

Definition 13 (Morphing Relations)**.**

- *$F_1 \to_i F_2$ if $F_1 \equiv \forall x.F$ and $F_2 \equiv [t/x]F$ for some term t.*
- *$F_1 \to_g F_2$ if $F_2 \equiv \forall x.F_1$.*
- *$F_1 \to_I F_2$ if $F_1 \equiv \Pi X^0.F$ and $F_2 \equiv [F'/X^0]F$ for formula F'.*
- *$F_1 \to_G F_2$ if $F_2 \equiv \Pi X^0.F_1$.*
- *$F_1 \to_{is} F_2$ if $F_1 \equiv \Pi X^1.F$ and $F_2 \equiv [S/X^1]F$ for some set S.*
- *$F_1 \to_{gs} F_2$ if $F_2 \equiv \Pi X^1.F_1$.*

Let $\twoheadrightarrow_{gi}$ denotes the reflexive and transitive closure of $\to_{i,g,I,G,is,gs}$.

Lemma 3. *Suppose no free variable of F occurs in Γ. If $\Gamma \vdash p : F$ and $F \twoheadrightarrow_{gi} F'$, then $\Gamma \vdash p : F'$.*

Definition 14.

$E_0(\Pi X^0.F) := E_0(F) \quad E_0(X^0) := X^0 \quad E_0(F_1 \to F_2) := F_1 \to F_2$
$E_0(\Pi X^1.F) := \Pi X^1.F \quad E_0(\forall x.F) := \forall x.F \quad E_0(t\epsilon S) := t\epsilon S$

Definition 15.

$E_1(\Pi X^1.F) := E_1(F) \quad E_1(X^0) := X^0 \quad E_1(F_1 \to F_2) := F_1 \to F_2$
$E_1(t\epsilon S) := t\epsilon S \quad E_1(\forall x.F) := \forall x.F \quad E_1(\Pi X^0.F) := \Pi X^0.F$

Definition 16.

$G(\Pi X^i.F) := \Pi X^i.F \quad G(X^0) := X^0 \quad G(F_1 \to F_2) := F_1 \to F_2$
$G(\forall x.F) := G(F) \quad G(t\epsilon S) := t\epsilon S$

Lemma 4. *$E_0([F'/X^0]F) \equiv [F''/X^0]E_0(F)$ for some F''; $E_1([S/X^1]F) \equiv [S/X^1]E_1(F)$; $G([t/x]F) \equiv [t/x]G(F)$.*

Proof. Proof by induction on the structure of F. □

Lemma 5. *If $F \twoheadrightarrow_{i,g} F'$, then there exist a substitution δ with domain of term variables and codomain of terms such that $\delta G(F) \equiv G(F')$.*

Proof. It suffices to consider $F \to_{i,g} F'$. If $F' \equiv \forall x.F$, then $G(F') \equiv G(F)$. If $F \equiv \forall x.F_1$ and $F' \equiv [t/x]F_1$, then $G(F) \equiv G(F_1)$. By lemma 4, we know $G(F') \equiv G([t/x]F_1) \equiv [t/x]G(F_1)$. □

Lemma 6. *If $F \twoheadrightarrow_{I,G} F'$, then there exist a substitution δ with domain of formula variables and codomain of formulas such that $\delta E_0(F) \equiv E_0(F')$.*

Proof. It suffices to consider $F \to_{I,G} F'$. If $F' \equiv \Pi X^0.F$, then $E_0(F') \equiv E_0(F)$. If $F \equiv \Pi X^0.F_1$ and $F' \equiv [F''/X^0]F_1$, then $E_0(F) \equiv E_0(F_1)$. By lemma 4, we know $E_0(F') \equiv E_0([F''/X^0]F_1) \equiv [F_2/X^0]E_0(F_1)$ for some F_2.

□

Lemma 7. *If $F \twoheadrightarrow_{is,gs} F'$, then there exist a substitution δ with domain of set variables and codomain of sets such that $\delta E_1(F) \equiv E_1(F')$.*

Proof. It suffices to consider $F \to_{is,gs} F'$. If $F' \equiv \Pi X^1.F$, then $E_1(F') \equiv E_1(F)$. If $F \equiv \Pi X^1.F_1$ and $F' \equiv [S/X^1]F_1$, then $E_1(F) \equiv E_1(F_1)$. By lemma 4, we know $E_1(F') \equiv E_1([S/X^1]F_1) \equiv [S/X^1]E_1(F_1)$. □

Theorem 6 (Compatibility)**.** *If $(F_1 \to F_2) \to^*_{\iota,\beta,i,g,I,G,is,gs} (F_1' \to F_2')$, then there exists a mixed substitution[4] δ such that $\delta(F_1 \to F_2) \to_\beta F_1' \to F_2'$. Thus $\delta F_1 \to_\beta F_1'$ and $\delta F_2 \to_\beta F_2'$.*

Proof. By lemma 5, lemma 6 and lemma 7, we have $\delta(F_1 \to F_2) \to_{\beta,\iota} F_1' \to F_2'$ for some mix substitution δ. Since $\to_\iota$ reduction can not happen in the sequence $\delta(F_1 \to F_2) \to_{\beta,\iota} F_1' \to F_2'$, so we have $\delta(F_1 \to F_2) \to_\beta F_1' \to F_2'$. Thus $\delta F_1 \to_\beta F_1'$ and $\delta F_2 \to_\beta F_2'$.

□

Lemma 8 (Inversion)**.**

- *If $\Gamma \vdash a : F$, then exist F_1 such that $F_1 \to^*_{\iota,\beta,i,g,I,G,is,gs} F$ and $(x : F_1) \in \Gamma$.*
- *If $\Gamma \vdash p_1 p_2 : F$, then exist F_1, F_2 such that $\Gamma \vdash p_1 : F_1 \to F_2$ and $\Gamma \vdash p_2 : F_1$ and $F_2 \to^*_{\iota,\beta,i,g,I,G,is,gs} F$.*
- *If $\Gamma \vdash \lambda a.p : F$, then exist F_1, F_2 such that $\Gamma, a : F_1 \vdash p : F_2$ and $F_1 \to F_2 \to^*_{\iota,\beta,i,g,I,G,is,gs} F$.*

Lemma 9 (Substitution)**.**

1. *If $\Gamma \vdash p : F$, then for any mixed substitution δ, $\delta\Gamma \vdash p : \delta F$.*

[4]A substitution that contains term, set and formula substitution

2. If $\Gamma, a : F \vdash p : F'$ and $\Gamma \vdash p' : F$, then $\Gamma \vdash [p'/a]p : F'$.

Theorem 7 (Preservation)**.** *If $\Gamma \vdash p : F$ and $p \to_\beta p'$, then $\Gamma \vdash p' : F$.*

Proof. We list one interesting case:

$$\frac{\Gamma \vdash p_1 : F_1 \to F_2 \quad \Gamma \vdash p_2 : F_1}{\Gamma \vdash p_1 p_2 : F_2}$$

Suppose $\Gamma \vdash (\lambda a.p_1)p_2 \to_\beta [p_2/a]p_1$. We know that $\Gamma \vdash \lambda a.p_1 : F_1 \to F_2$ and $\Gamma \vdash p_2 : F_1$. By inversion on $\Gamma \vdash \lambda a.p_1 : F_1 \to F_2$, we know that there exist F_1', F_2' such that $\Gamma, a : F_1' \vdash p_1 : F_2'$ and $(F_1' \to F_2') \to^*_{\iota,\beta,i,g,I,G,is,gs} (F_1 \to F_2)$. By theorem 6, we have $\delta(F_1' \to F_2') =_\beta (F_1 \to F_2)$. By Church-Rosser of $=_\beta$, we have $\delta F_1' =_\beta F_1$ and $\delta F_2' =_\beta F_2$. So by (1) of lemma 9, we have $\Gamma, a : \delta F_1' \vdash p_1 : \delta F_2'$. So $\Gamma, a : \delta F_1' \vdash p_1 : F_2$. Since $\Gamma \vdash p_2 : \delta F_1'$, by (2) of lemma 9, $\Gamma \vdash [p_2/a]p_1 : F_2$.

□

4 The Polymorphic Dependent Type System $\mathfrak{G}[t]$

In this section, we first show a polymorphic dependent type system $\mathfrak{G}[t]$. Then, we define an embedding from $\mathfrak{G}[t]$ to $\mathfrak{G}$. The embedding is invertable, thus we can transform a judgement in $\mathfrak{G}$ to a judgement in $\mathfrak{G}[t]$ and vice versa. We call this behavior *reciprocity.*

Definition 17 (Syntax)**.**
Lambda Terms $t \ := \ x \mid \lambda x.t \mid tt'$
Internal Types $U \ := X^1 \mid \iota x.Q \mid \Pi x : U.U' \mid \Delta X^1.U$
Internal Formula $Q \ := X^0 \mid t\epsilon U \mid \Pi X^0.Q \mid Q \to Q' \mid \forall x.Q \mid \Pi X^1.Q$
Internal Context $\Psi \ := \ \cdot \mid \Psi, x\epsilon U$

Besides basic set formed by formula and set variable, internal types includes dependent-type-like construct $\Pi x : U.U'$ and polymorphic-type-like construct $\Delta X^1.U$. The internal formula stay the same as formula in $\mathfrak{G}$ except replacing the notion of set by the notion of internal type. We can view ϵ relation as a kind of typing relation, thus we have the notion of internal context as a list of formula of the form $x\epsilon U$ and the following internal typing relation.

Definition 18 (Internal Typing)**.**

$$\frac{x\epsilon U \in \Psi}{\Psi \Vdash x\epsilon U} \qquad \frac{\Psi, x\epsilon U \Vdash t\epsilon U'}{\Psi \Vdash \lambda x.t\epsilon\Pi x : U.U'}$$

$$\frac{\Psi \Vdash t\epsilon U \quad X^1 \notin FV(\Psi)}{\Psi \Vdash t\epsilon\Delta X^1.U} \qquad \frac{\Psi \Vdash t\epsilon\Delta X^1.U}{\Psi \Vdash t\epsilon[U'/X]U}$$

$$\frac{\Psi \Vdash t_1\epsilon\Pi x : U'.U \quad \Psi \Vdash t_2\epsilon U'}{\Psi \Vdash t_1 t_2\epsilon[t_2/x]U}$$

The internal typing shares a lot of similarities between the usual polymorphic dependent type system. But we want to emphasis that the meaning of internal typing in $\mathfrak{G}[t]$ is different from the usual notion of typing. The internal typing relation is an internal formula in $\mathfrak{G}[t]$ (which lies in the object language), while the ususal notion of typing is a meta-level relation. The emergence of internal typing benefits from the (ϵ, ι)-notation.

Now let us relate $\mathfrak{G}[t]$ with $\mathfrak{G}$.

Definition 19. $[\![\cdot]\!]$ *is an* ***embedding*** *from internal types in* $\mathfrak{G}[t]$ *to sets in* $\mathfrak{G}$*, internal formulas in* $\mathfrak{G}[t]$ *to formulas in* $\mathfrak{G}$*.*
$[\![X^1]\!] := X^1$
$[\![\iota x.Q]\!] := \iota x.[\![Q]\!]$
$[\![\Pi x : U'.U]\!] := \iota f.\forall x.(x\epsilon[\![U']\!] \to f\ x\epsilon[\![U]\!])$*, where f is fresh.*
$[\![\Delta X^1.U]\!] := \iota x.(\Pi X^1.x\epsilon[\![U]\!])$*, where x is fresh.*
$[\![X^0]\!] := X^0$
$[\![t\epsilon U]\!] := t\epsilon[\![U]\!]$
$[\![Q \to Q']\!] := [\![Q]\!] \to [\![Q]\!]$
$[\![\Pi X^i.Q]\!] := \Pi X^i.[\![Q]\!]$.
$[\![\forall x.Q]\!] := \forall x.[\![Q]\!]$.
$[\![x\epsilon U, \Psi]\!] := x\epsilon[\![U]\!], [\![\Psi]\!]$

Lemma 10. $[t'/x][\![U]\!] = [\![[t'/x]U]\!]$ *and* $[[\![U']\!]/X^1][\![U]\!] = [\![[U'/X^1]U]\!]$.

Proof. By induction on structure of U. ☐

Theorem 8. *If* $\Psi \Vdash t\epsilon U$*, then* $[\![\Psi]\!] \vdash t\epsilon[\![U]\!]$.

Proof. By induction on the derivation of $\Psi \Vdash t\epsilon U$.
Case:

$$\frac{x\epsilon U \in \Psi}{\Psi \Vdash x\epsilon U}$$
$[\![\Psi]\!] \vdash x\epsilon[\![U]\!]$, since $x\epsilon[\![U]\!] \in [\![\Psi]\!]$.
Case:

$$\frac{\Psi, x\epsilon U \Vdash t\epsilon U'}{\Psi \Vdash \lambda x.t\epsilon\Pi x : U.U'}$$
By induction, we have $[\![\Psi]\!], x\epsilon[\![U]\!] \vdash t\epsilon[\![U']\!]$. So $[\![\Psi]\!] \vdash x\epsilon[\![U]\!] \to t\epsilon[\![U']\!]$, then by $\forall$-intro rule, we have $[\![\Psi]\!] \vdash \forall x.(x\epsilon[\![U]\!] \to t\epsilon[\![U']\!])$. By comprehension rule and beta-reduction, we get $[\![\Psi]\!] \vdash \lambda x.t\epsilon\iota f.\forall x.(x\epsilon[\![U]\!] \to f\ x\epsilon[\![U']\!])$. We know that $[\![\Pi x : U.U']\!] := \iota f.\forall x.(x\epsilon[\![U]\!] \to f\ x\epsilon[\![U']\!])$.
Case:

$$\frac{\Psi \Vdash t\epsilon\Pi x : U'.U \quad \Psi \Vdash t'\epsilon U'}{\Psi \Vdash tt'\epsilon[t'/x]U}$$
By induction, we have $[\![\Psi]\!] \vdash t\epsilon\iota f.\forall x.(x\epsilon[\![U']\!] \to f\ x\epsilon[\![U]\!])$ and $[\![\Psi]\!] \vdash t'\epsilon[\![U']\!]$. By comprehension, we have $[\![\Psi]\!] \vdash \forall x.(x\epsilon[\![U']\!] \to t\ x\epsilon[\![U]\!])$. Instantiate x with t', we have $[\![\Psi]\!] \vdash t'\epsilon[\![U']\!] \to t\ t'\epsilon[t'/x][\![U]\!]$. So by modus ponens, we have $[\![\Psi]\!] \vdash tt'\epsilon[t'/x][\![U]\!]$. By lemma 10, we know that $[t'/x][\![U]\!] = [\![[t'/x]U]\!]$. So $[\![\Psi]\!] \vdash tt'\epsilon[\![[t'/x]U]\!]$.
Case:

$$\frac{\Psi \Vdash t\epsilon U \quad X^1 \notin FV(\Psi)}{\Psi \Vdash t\epsilon\Delta X^1.U}$$
By induction, one has $[\![\Psi]\!] \vdash t\epsilon[\![U]\!]$. So one has $[\![\Psi]\!] \vdash \Pi X^1.t\epsilon[\![U]\!]$. So by comprehension, one has $[\![\Psi]\!] \vdash t\epsilon\iota x.\Pi X^1.x\epsilon[\![U]\!]$.
Case:

$$\frac{\Psi \Vdash t\epsilon\Delta X^1.U}{\Psi \Vdash t\epsilon[U'/X]U}$$
By induction, one has $[\![\Psi]\!] \vdash t\epsilon\iota x.\Pi X^1.x\epsilon[\![U]\!]$. By comprehension, we have $[\![\Psi]\!] \vdash \Pi X^1.t\epsilon[\![U]\!]$. So by instantiation, we have $[\![\Psi]\!] \vdash t\epsilon[[\![U']\!]/X^1][\![U]\!]$. Since by lemma 10, we know $[[\![U']\!]/X^1][\![U]\!] = [\![[U'/X^1]U]\!]$.
☐

We now define the inverse of $[\![\cdot]\!]$.

Definition 20.
$[\![\cdot]\!]^{-1}$ is a maping from the sets in $\mathfrak{G}$ to the internal types in $\mathfrak{G}[t]$, from the formulas in $\mathfrak{G}$ to the internal formulas $\mathfrak{G}[t]$.
$[\![X^1]\!]^{-1} := X^1$
$[\![\iota f.\forall x.(x\epsilon S' \to f\ x\epsilon S)]\!]^{-1} := \Pi x : [\![S']\!]^{-1}.[\![S]\!]^{-1}$, *where f is fresh.*
$[\![\iota x.(\Pi X^1.x\epsilon S)]\!]^{-1} := \Delta X^1.[\![S]\!]^{-1}$, *where x is fresh.*
$[\![\iota x.T]\!]^{-1} := \iota x.[\![T]\!]^{-1}$
$[\![X^0]\!]^{-1} := X^0$
$[\![t\epsilon S]\!]^{-1} := t\epsilon[\![S]\!]^{-1}$
$[\![T \to T']\!]^{-1} := [\![T]\!]^{-1} \to [\![T]\!]^{-1}$
$[\![\Pi X^i.T]\!]^{-1} := \Pi X^i.[\![T]\!]^{-1}$.
$[\![\forall x.T]\!]^{-1} := \forall x.[\![T]\!]^{-1}$.
$[\![x\epsilon S, \Gamma]\!]^{-1} := x\epsilon[\![S]\!]^{-1}, [\![\Gamma]\!]^{-1}$

Lemma 11. *$[\![[\![S]\!]^{-1}]\!] = S$ and $[\![[\![U]\!]]\!]^{-1} = U$.*

Proof. By induction. □

By lemma 11, if we have $\Gamma \vdash t\epsilon S$ in $\mathfrak{G}$, we can go to $\mathfrak{G}[t]$ by $[\![\Gamma]\!]^{-1} \Vdash t\epsilon[\![S]\!]^{-1}$. Then, after a few deductions in $\mathfrak{G}[t]$, we can use theorem 8 to go back to $\mathfrak{G}$.

5 Proving Peano's Axioms

In this section, we show how to prove all of Peano's axioms in $\mathfrak{G}$. First, let us see the definition of natural number.

Definition 21 (Scott Numerals)**.**
$\mathsf{Nat} := \iota x.\Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to x\epsilon C$
$\mathsf{S}\ := \lambda n.\lambda s.\lambda z.s\ n$
$0\ := \lambda s.\lambda z.z$

We define the set of numerals Nat, and the $\mathsf{S}, 0$ are Scott encoded numerals.

Theorem 9 (Peano's Axiom 1)**.** *$\vdash 0\epsilon\mathsf{Nat}$.*

Proof. By comprehension, we want to show $\vdash \Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to 0\epsilon C$, which is obvious[5]. □

Definition 22 (Leibniz Equality)**.** *$x = y\ :=\ \Pi C^1.x\epsilon C \to y\epsilon C$.*

Theorem 10 (Peano Axiom 2-4)**.**

1. *$\forall x.x = x$.*
2. *$\forall x.\forall y.x = y \to y = x$.*
3. *$\forall x.\forall y.\forall z.x = y \to y = z \to x = z$.*

Proof. We only prove 2, the others are easy. Assume $\Pi C^1.x\epsilon C \to y\epsilon C(1)$, we want to show $y\epsilon A \to x\epsilon A$ for any A^1. Instantiate C in (1) with $\iota z.(z\epsilon A \to x\epsilon A)$. By comprehension, we get $(x\epsilon A \to x\epsilon A) \to (y\epsilon A \to x\epsilon A)$. And we know that $x\epsilon A \to x\epsilon A$ is derivable in our system, so by modus ponens we get $y\epsilon A \to x\epsilon A$. □

Lemma 12. *$\cdot \vdash \forall a.\forall b.\Pi P^1.(a\epsilon P \to a = b \to b\epsilon P)$.*

[5]Note the proof terms for the theorem is Church numeral 0.

Proof. By modus ponens. $\square$

Theorem 11 (Peano's Axiom 5)**.** $\cdot \vdash \forall a.\forall b.(a\epsilon\mathsf{Nat} \to a = b \to b\epsilon\mathsf{Nat})$.

Proof. Let $P := \iota x.x\epsilon\mathsf{Nat}$ for lemma 12. $\square$

Theorem 12 (Peano's Axiom 6)**.** $\cdot \vdash \forall m.(m\epsilon\mathsf{Nat} \to \mathsf{S}m\epsilon\mathsf{Nat})$.

Proof. Assume $m\epsilon\mathsf{Nat}$. We want to show $\mathsf{S}m\epsilon\mathsf{Nat}$. By comprehension, we just need to show $\Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to \mathsf{S}m\epsilon C$. By Intros, we want to derive $m\epsilon\mathsf{Nat}, \forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C), 0\epsilon C \vdash \mathsf{S}m\epsilon C$. Since $m\epsilon\mathsf{Nat}$, we know that $\Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to m\epsilon C$. By Modus Ponens, we have $m\epsilon C$. We know that $m\epsilon\mathsf{Nat}, \forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C), 0\epsilon C \vdash (m\epsilon C) \to (\mathsf{S}m)\epsilon C$. Thus we derive $m\epsilon\mathsf{Nat}, \forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C), 0\epsilon C \vdash \mathsf{S}m\epsilon C$, which is what we want[6]. $\square$

Theorem 13 (Induction Principle)**.**
$\vdash \Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to \forall m.(m\epsilon\mathsf{Nat} \to m\epsilon C)$

Proof. Assume $\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C), 0\epsilon C$ and $m\epsilon\mathsf{Nat}$. We need to show that $m\epsilon C$. Since $m\epsilon\mathsf{Nat}$ implies that $\Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to m\epsilon C$. So by instantiation and modus ponens we get $m\epsilon C$. [7]

$\square$

In order to proceed to prove Peano's axiom 7, we need to define a notion of contradiction in $\mathfrak{G}$.

Definition 23 (Notion of Contradiction)**.** $\bot := \forall x.\forall y.(x = y)$.

Theorem 14 (Consistency (Meta)[8])**.** $\bot$ *is uninhabited in* $\mathfrak{G}[p]$.

Proof. Suppose $\bot$ is inhabited, that is, there is a proof term p such that $\cdot \vdash p : \forall x.\forall y.\Pi C^1.x\epsilon C \to y\epsilon C$. By theorem 5 and theorem 7, we know that p must normalized at some normal proof term p' such that $\cdot \vdash p' : \forall x.\forall y.\Pi C^1.x\epsilon C \to y\epsilon C$. We know that p' must of the form $\lambda a.p''$ with $a : x\epsilon C$. Since $=_{\beta,\iota}$ is Church-Rosser, we can not convert $x\epsilon C$ to $y\epsilon C$. So p' can not exist. $\square$

Lemma 13. $\vdash 0 = \mathsf{S}0 \to \bot$.

Proof. Assume $0 = \mathsf{S}0$, namely, $\Pi C^1.0\epsilon C \to \mathsf{S}0\epsilon C$ †. We want to show $\forall x.\forall y.\Pi A^1.x\epsilon A \to y\epsilon A$. Assume $x\epsilon A$ (1). We now instantiate C with $\iota u.(((\lambda n.n\ (\lambda z.y)\ x)\ u)\epsilon A)$ in †. By comprehension and beta reduction, we get $x\epsilon A \to y\epsilon A$ (2). By modus ponens of (1), (2), we get $y\epsilon A$. $\square$

We also need predecessor to prove Peano's axiom 7.

Definition 24. $\mathsf{Pred} := \lambda n.n(\lambda x.x)0$.

Lemma 14 (Congruence of Equality)**.** $\vdash \forall a.\forall b.\forall f.a = b \to fa = fb$.

Proof. Assume $\Pi C.a\epsilon C \to b\epsilon C$. Let $C := \iota x.fx\epsilon P$ with P free. Instantiate C for the assumption, we get $a\epsilon(\iota x.fx\epsilon P) \to b\epsilon(\iota x.fx\epsilon P)$. By conversion, we get $f\ a\epsilon P \to f\ b\epsilon P$. So by polymorphic generalization, we get $f\ a = f\ b$.

$\square$

Theorem 15 (Peano's Axiom 7)**.** $\vdash \forall n.n\epsilon\mathsf{Nat} \to (\mathsf{S}n = 0 \to \bot)$

Proof. We will use induction principle (theorem 13) to prove this. We instantiate C in theorem 13 with $\iota z.(\mathsf{S}z = 0 \to \bot)$, we have $\forall y.((\mathsf{S}y = 0 \to \bot) \to (\mathsf{SS}y = 0 \to \bot)) \to (\mathsf{S}0 = 0 \to \bot) \to \forall m.(m\epsilon\mathsf{Nat} \to (\mathsf{S}m = 0 \to \bot))$. Base case is by lemma 13. For the step case, we assume $\mathsf{S}y = 0 \to \bot$ (IH), we want to show $\mathsf{SS}y = 0 \to \bot$. Assuming $\mathsf{SS}y = 0$, we want to show $\bot$. By lemma 14, we know that $\mathsf{Pred}(\mathsf{SS}y) = \mathsf{Pred}0$. By beta-reduction, we have $\mathsf{S}y = 0$. Thus by IH, we have $\bot$. $\square$

[6]Note that the proof term for this theorem is Church successor.

[7]The proof terms for this theorem is $\lambda s.\lambda z.\lambda n.n\ s\ z$.

[8]Meaning the proof of this theorem relies on meta-level argument.

Theorem 16 (Peano's Axiom 8)**.** $\forall m.\forall n.m\epsilon\mathsf{Nat} \to n\epsilon\mathsf{Nat} \to \mathsf{S}m = \mathsf{S}n \to m = n$.

Proof. Assume $\mathsf{S}m = \mathsf{S}n$. By lemma 14, we have $\mathsf{Pred}(\mathsf{S}m) = \mathsf{Pred}(\mathsf{S}n)$. So by beta reduction, we have $m = n$. ☐

In order to state Peano's axiom 9, we need to extend the formula in $\mathfrak{G}$ with $F \wedge F'$. And the proof of $F \wedge F'$ consist of both the proof of F and the proof of F'.

Theorem 17 (Peano's Axiom 9, Weak Induction)**.**
$\vdash \Pi C^1.(\forall y.(y\epsilon\mathsf{Nat} \wedge (y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to \forall m.(m\epsilon\mathsf{Nat} \to m\epsilon C)$

Proof. Assume $\forall y.(y\epsilon\mathsf{Nat} \wedge (y\epsilon C) \to (\mathsf{S}y)\epsilon C)$ † and $0\epsilon C$. We want to show that $\forall m.(m\epsilon\mathsf{Nat} \to m\epsilon C)$. We just need to show $\forall m.(m\epsilon\mathsf{Nat} \to (m\epsilon\mathsf{Nat} \wedge m\epsilon C))$. We prove this using theorem 13. For the base case, it is obvious that $0\epsilon\mathsf{Nat} \wedge 0\epsilon C$. For step case, assuming $z\epsilon\mathsf{Nat} \wedge z\epsilon C$ (IH), we need to show $\mathsf{S}z\epsilon\mathsf{Nat} \wedge \mathsf{S}z\epsilon C$. By theorem 12, we have $\mathsf{S}z\epsilon\mathsf{Nat}$. By †, we know that $\mathsf{S}z\epsilon C$. Thus $\forall z.(z\epsilon\mathsf{Nat} \to (z\epsilon\mathsf{Nat} \wedge z\epsilon C))$. ☐

We have proved all Peano's nine axioms. We could continue to define $<$ relation for natural number and prove strong induction principle from the weak induction principle, and go further to formalize more mathematics. But we will not do that here.

6 Reasoning about Programs

System $\mathfrak{G}$ is expressive enough to reason about programs. Here by programs we mean pure lambda calculus with Scott encoding. We will show some simple examples about Scott numerals, and then we show how to encode Vector with $\mathfrak{G}$.

Definition 25. $\mathsf{add} := \lambda n.\lambda m.n\ (\lambda p.\mathsf{add}\ p\ (\mathsf{S}m))\ m$

We know that the above recursive equation can be solved by fixpoint. But we do not bother to solve it. The way we treat it is use it as a kind of build in beta equality, when every we see a add, we one step unfold it.

Theorem 18. $\cdot \vdash \forall n.(n\epsilon\mathsf{Nat} \to \mathsf{add}\ n\ 0 = n)$.

Proof. We want to show $\forall n.(n\epsilon\mathsf{Nat} \to \mathsf{add}\ n\ 0 = n)$. Let $P := \iota x.\mathsf{add}\ x\ 0 = x$. Instantiate the C^1 in theorem 13 with P, we get $\forall y.(\mathsf{add}\ y\ 0 = y \to \mathsf{add}\ (\mathsf{S}y)\ 0 = \mathsf{S}y) \to \mathsf{add}\ 0\ 0 = 0 \to \forall m.(m\epsilon\mathsf{Nat} \to m\epsilon P)$. We just have to prove $\forall y.(\mathsf{add}\ y\ 0 = y \to \mathsf{add}\ (\mathsf{S}y)\ 0 = \mathsf{S}y)$ and $\mathsf{add}\ 0\ 0 = 0$. For the base case, we want to show $\Pi C.\mathsf{add}\ 0\ 0\epsilon C \to 0\epsilon C$. Assume $\mathsf{add}\ 0\ 0\epsilon C$, since $\mathsf{add}\ 0\ 0 \to_\beta 0$, by conversion, we get $0\epsilon C$. For the step case is a bit complicated, assume $\mathsf{add}\ y\ 0 = y$, we want to show $\mathsf{add}\ (\mathsf{S}y)\ 0 = \mathsf{S}y$. Since $\mathsf{add}\ y\ 0 \to_\beta y\ (\lambda p.\mathsf{add}\ p\ (\mathsf{S}0))\ 0$, And $\mathsf{add}\ (\mathsf{S}y)\ 0 \to_\beta \mathsf{add}\ y\ (\mathsf{S}0) \leftarrow_\beta^* \mathsf{S}(\mathsf{add}\ y\ 0)$. So lemma 14 will give us this. ☐

Theorem 19. $\cdot \vdash \forall n.(n\epsilon\mathsf{Nat} \to \forall m.(m\epsilon\mathsf{Nat} \to \mathsf{add}\ n\ m\epsilon\mathsf{Nat}))$. *After transformed to $\mathfrak{G}[t]$, we have* $\Vdash \mathsf{add}\epsilon\mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$. [9]

Proof. Let $P := \iota z.\forall m.(m\epsilon\mathsf{Nat} \to \mathsf{add}\ z\ m\epsilon\mathsf{Nat})$. We instantiate the C in theorem 13, we have $(\forall y.((y\epsilon P) \to (\mathsf{S}y)\epsilon P)) \to 0\epsilon P \to \forall m.(m\epsilon\mathsf{Nat} \to m\epsilon P)$. For the base case, we need to show $\forall m.(m\epsilon\mathsf{Nat} \to \mathsf{add}\ 0\ m\epsilon\mathsf{Nat})$. By $\mathsf{add}\ 0\ m \to_\beta m$, we have the base case. For the step case, assuming $\forall m.(m\epsilon\mathsf{Nat} \to \mathsf{add}\ y\ m\epsilon\mathsf{Nat})$ (IH), we need to show $\forall m.(m\epsilon\mathsf{Nat} \to \mathsf{add}\ (\mathsf{S}y)\ m\epsilon\mathsf{Nat})$. We know that $\mathsf{add}\ (\mathsf{S}y)\ m \to_\beta^* \mathsf{add}\ y\ (\mathsf{S}m)$. By (IH), we know $\mathsf{add}\ y\ (\mathsf{S}m)\epsilon\mathsf{Nat}$. So $\mathsf{add}\ (\mathsf{S}y)\ m\epsilon\mathsf{Nat}$. ☐

6.1 Termination Analysis in System $\mathfrak{G}$

In this section, we will show that elements in the inductive defined set are solvable. A direct consequence of this result is that these elements is terminating with respect to head reduction.

[9]We write $U \to U'$ if $\Pi x : U.U'$ with $x \notin \mathrm{FV}(U')$.

6.1.1 Preliminary

The definitions, lemmas and theorems in this subsection come from Barendregt's [1], Chapter 8.3.

Definition 26 (Solvability)**.**

- *A closed lambda term t, i.e.* $\mathrm{FV}(t) = \emptyset$*, is solvable if there exists* $t_1, ..., t_n$ *such that* $tt_1...t_n =_\beta \lambda x.x$.
- *An arbitrary term t is solvable if the closure* $\lambda x_1...\lambda x_n.t$*, where* $\{x_1, ..., x_n\} = \mathrm{FV}(t)$*, is solvable.*
- *t is unsolvable iff t is not solvable.*

Lemma 15. *Every term t is of the following forms:*

- $\lambda x_1....\lambda x_n.xt_1...t_m$*, where* $n, m \geq 0$*. It is called head normal form.*
- $\lambda x_1....\lambda x_n.((\lambda y.t)t_1)...t_m$*, where* $(\lambda y.t)t_1$ *is called head redex.*

Definition 27 (Head Reduction)**.** $t \to_h t'$ *if t' is resulting from contracting the head redex of t.*

Theorem 20. *A term t has a head normal form iff it is terminating with respect to head reduction.*

Theorem 21 (Wadsworth)**.** *t is solvable iff t has a head normal form. In particular, all terms in normal forms are solvable, and unsolvable terms have no normal form.*

Theorem 22 (Genericity)**.** *For a unsolvable term t, if $t_1 t =_\beta t_2$, where t_2 in normal form, then for any t', we have $t_1 t' =_\beta t$.*

So unsolvable in general is computational irrelavance, thus it is reasonable to equate all unsolvable terms.

Definition 28 (Omega-Reduction)**.** *Let Ω be* $(\lambda x.xx)\lambda x.xx$*, then* $t \to_\omega \Omega$ *iff t is unsolvable and* $t \not\equiv \Omega$.

Theorem 23. $\to_\beta \cup \to_\omega$ *is Church-Rosser.*

6.1.2 Head Normalization

We add Omega-reduction as part of the term reduction in $\mathfrak{G}$. We now define another notion of contradictory: $\perp' := \forall x.x = \Omega$. Note that this will imply $\forall x.\forall y.x = y$, thus we can safely take it as contradictory.

Theorem 24. $\vdash \forall n.(n\epsilon\mathsf{Nat} \to (n = \Omega \to \perp'))$.

Proof. We will prove this by induction. Recall the induction principle:
$\Pi C^1.(\forall y.((y\epsilon C) \to (\mathsf{S}y)\epsilon C)) \to 0\epsilon C \to \forall m.(m\epsilon\mathsf{Nat} \to m\epsilon C)$.
We instantiate C with $\iota z.(z = \Omega \to \perp')$, by comprehension, we then have $(\forall y.((y = \Omega \to \perp') \to (\mathsf{S}y = \Omega \to \perp')) \to (0 = \Omega \to \perp') \to \forall m.(m\epsilon\mathsf{Nat} \to (m = \Omega \to \perp'))$. It is enough to show that $0 = \Omega \to \perp'$ and $\mathsf{S}y = \Omega \to \perp'$. We know that for Scott numerals we have $0 := \lambda s.\lambda z.z$ and $\mathsf{S}y := \lambda s.\lambda z.sy$. Assume $0 = \Omega = \lambda x_1.\lambda x_2.\Omega$, let $F := \lambda u.u\ p\ q$. Assume $q\epsilon X^1$, then $F\ 0\epsilon X^1$ (since $F0 =_\beta q$). So $F\ (\lambda x_1.\lambda x_2.\Omega)\epsilon X^1$, thus $\Omega\epsilon X^1$. Thus we just show $\forall X^1.(q\epsilon X^1 \to \Omega\epsilon X^1)$, which means $\forall q.q = \Omega$. So $0 = \Omega \to \perp'$. Now let us show $\mathsf{S}y = \Omega \to \perp'$. Assume $\lambda s.\lambda z.sy = \Omega = \lambda x_1.\lambda x_2.\Omega$. Let $F := \lambda n.n\ (\lambda p.q)\ z$. Assume $q\epsilon X^1$, then $F\ (\lambda s.\lambda z.sy)\epsilon X^1$, thus $F\ (\lambda x_1.\lambda x_2.\Omega)\epsilon X^1$, meaning $\Omega\epsilon X^1$. So we just show $\Pi X^1.(q\epsilon X \to \Omega\epsilon X)$. Thus $\forall q.q = \Omega$. So $\mathsf{S}y = \Omega \to \perp'$. □

Above theorem implies that all the member of Nat has a head normal form and it can be generalized to show that the elements of inductive describable set are solvable. To see this, we prove the following meta-theorem.

Theorem 25. *If* $\vdash t\epsilon\mathsf{Nat}$*, then* $t \neq_{\beta,\omega} \Omega$.

Proof. By theorem 24, we know that $\vdash t = \Omega \to \perp$. We know that by the *conv* rule, if $t =_{\beta,\omega} t'$, then $\vdash t = t'$ in $\mathfrak{G}$. By contraposition, we have if $\not\vdash t = t'$, then $t \neq_{\beta,\omega} t'$. Since $\mathfrak{G}$ is consistent (theorem 14), we know that $\not\vdash t = \Omega$. So $t \neq_{\beta,\omega} \Omega$. □

6.2 Leibniz Equality in $\mathfrak{G}$

We know that by the *conv* rule, if $t =_{\beta,\eta,\omega} t'$, then $\vdash t = t'$ in $\mathfrak{G}$. It is natural to consider wether the other direction is the case, namely, to prove: if $\vdash t = t'$, then $t =_{\beta,\eta,\omega} t'$. By the contraposition, we have: if $t \neq_{\beta,\eta,\omega} t'$, then $\not\vdash t = t'$. We conjecture that it is hard to prove this. Due to the genericity (theorem 22) property in lambda calculus. Oraclely, if t is solvable and t' is unsolvable, we can not define a lambda term F such that $Ft =_\beta x$ and $Ft' =_\beta y$. Because by genericity, we would have $Ft =_\beta y$, thus $x =_\beta y$, which is impossible for beta-reduction. However, when t, t' both are solvable and $t \neq_{\beta,\eta} t'$, then by the results of Coppo et al. [3], we can indeed define a lambda term F such that $Ft =_\beta x$ and $Ft' =_\beta y$. So we can derive $\vdash t = t' \to \bot$ in System $\mathfrak{G}$.

Theorem 26. *Assume t_1, t_2 are solvable terms. If $\vdash t_1 = t_2$ in $\mathfrak{G}$, then $t_1 =_{\beta\eta} t_2$.*

Proof. By contraposition, we want to prove: if $t_1 \neq_{\beta\eta} t_2$, then $\not\vdash t_1 = t_2$. Since t_1 and t_2 are *distinct*, then t_1 and t_2 are *separable*[10]. i.e. there exists a lambda term F such that $Ft_1 =_\beta x$ and $Ft_2 =_\beta y$. Thus we can derive $\vdash t = t' \to \bot$. Since $\mathfrak{G}$ is consistent, we have $\not\vdash t_1 = t_2$. $\square$

The developments in this section together with section 6.1.2 shows that if $\vdash t = t' \to \bot$ in $\mathfrak{G}$, then $t \neq_{\beta,\eta,\omega} t'$. And if t_1, t_2 are solvable terms, then $\vdash t_1 = t_2$ in $\mathfrak{G}$ implies $t_1 =_{\beta\eta} t_2$.

6.3 Vector Encoding in System $\mathfrak{G}$

In order to do vector encoding in $\mathfrak{G}$, we need to extend the formulas of $\mathfrak{G}$ to specify binary relation, so we add the following syntatic category.

Definition 29 (Relation)**.**
Formula F ::= ... $\mid (t;t')\epsilon R \mid \Pi X^2.F$
Binary Relation R ::= $X^2 \mid \iota(x;y).F$
Relational Comprehension $(t;t')\epsilon\iota(x;y).F =_\iota [(t;t')/(x;y)]F$

Definition 30 (Vector)**.**
$\mathsf{vec}(U,n) :=$
$\iota x.\Pi C^2.(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y; \mathsf{S}m)\epsilon C)) \to (\mathsf{nil};0)\epsilon C \to (x;n)\epsilon C$
$\mathsf{nil} := \lambda y.\lambda x.x$
$\mathsf{cons} := \lambda n.\lambda v.\lambda l.\lambda y.\lambda x.y\ n\ v\ l.$

Lemma 16. $\vdash \mathsf{nil}\epsilon\mathsf{vec}(U,0)$.

Lemma 17. $\vdash \forall n.n\epsilon\mathsf{Nat} \to \forall u.(u\epsilon U \to \forall l.(l\epsilon\mathsf{vec}(U,n) \to (\mathsf{cons}\ n\ u\ l)\epsilon\mathsf{vec}(U,\mathsf{S}n)))$. *Transform to $\mathfrak{G}[t]$, we get* $\Vdash \mathsf{cons}\epsilon\Pi n : \mathsf{Nat}.U \to \mathsf{vec}(U,n) \to \mathsf{vec}(U,\mathsf{S}n)$.

Proof. Assume $n\epsilon\mathsf{Nat}, u\epsilon U, l\epsilon\mathsf{vec}(U,n)$. We want to show $(\mathsf{cons}\ n\ u\ l)\epsilon\mathsf{vec}(U,\mathsf{S}n)$. By comprehension, we need to show $\Pi C^2.(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon C)) \to (\mathsf{nil};0)\epsilon C \to ((\mathsf{cons}\ n\ u\ l);\mathsf{S}n)\epsilon C$. Assume that we have $\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon C)$ † and $(\mathsf{nil};0)\epsilon C$, we need to show that $((\mathsf{cons}\ n\ u\ l);\mathsf{S}n)\epsilon C$. We know that $l\epsilon\mathsf{vec}(U,n)$, by comprehension, we have
$\Pi C^2.(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon C)) \to (\mathsf{nil};0)\epsilon C \to (l;n)\epsilon C$.
By modus ponens, we have $(l;n)\epsilon C$. Instantiate y with l, m with n, u with u in †, we have $n\epsilon\mathsf{Nat} \to u\epsilon U \to (l;n)\epsilon C \to (\mathsf{cons}\ n\ u\ l;\mathsf{S}n)\epsilon C$. So by modus ponens, we have $(\mathsf{cons}\ n\ u\ l;\mathsf{S}n)\epsilon C$.
$\square$

Theorem 27 (Induction Principle)**.**
$\vdash \mathsf{Ind}(U,n) :=$
$\Pi C^2.(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon C)) \to (\mathsf{nil};0)\epsilon C \to \forall l.(l\epsilon\mathsf{vec}(U,n) \to (l;n)\epsilon C)$

[10]See Barendregt's [1], Page 256

Proof. Assume we have $l\epsilon\mathsf{vec}(U,n)$ and
$\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon C), (\mathsf{nil};0)\epsilon C$.
We want to show $(l;n)\epsilon C$. By comprehension, we have
$\Pi C^2.(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon C \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon C)) \to (\mathsf{nil};0)\epsilon C \to (l;n)\epsilon C$.
By modus ponens, we have $(l;n)\epsilon C$.

□

Definition 31 (Append)**.**
$\mathsf{app} := \lambda n_1.\lambda n_2.\lambda l_1.\lambda l_2.l_1(\lambda m.\lambda h.\lambda t.\mathsf{cons}\ (m+n_2)\ h\ (\mathsf{app}\ m\ n_2\ t\ l_2))l_2$

Theorem 28. $\Vdash \mathsf{app}\epsilon\Pi n_1 : \mathsf{Nat}.\Pi n_2 : \mathsf{Nat}.\mathsf{vec}(U,n_1) \to \mathsf{vec}(U,n_2) \to \mathsf{vec}(U,n_1+n_2)$

Proof. Note that we state the theorem in $\mathfrak{G}[t]$. So we want to derive
$n_1\epsilon\mathsf{Nat}, n_2\epsilon\mathsf{Nat} \Vdash \lambda l_1.\lambda l_2.l_1(\lambda m.\lambda h.\lambda t.\mathsf{cons}\ (m+n_2)\ h\ (\mathsf{app}\ m\ n_2\ t\ l_2))l_2\ \epsilon$
$\mathsf{vec}(U,n_1) \to \mathsf{vec}(U,n_2) \to \mathsf{vec}(U,n_1+n_2)$.
We now transform it back to $\mathfrak{G}$, we have:
$n_1\epsilon\mathsf{Nat}, n_2\epsilon\mathsf{Nat} \vdash \forall x_1.x_1\epsilon\mathsf{vec}(U,n_1) \to$
$\forall x_2.(x_2\epsilon\mathsf{vec}(U,n_2) \to x_1(\lambda m.\lambda h.\lambda t.\mathsf{cons}\ (m+n_2)\ h\ (\mathsf{app}\ m\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,n_1+n_2))$.
We instantiate the C in theorem 27 by
$P := \iota\ (l;n)\ .\forall x_2.(x_2\epsilon\mathsf{vec}(U,n_2) \to$
$l\ (\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,\ n\ +n_2))$.
So we get
$(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon P \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon P)) \to (\mathsf{nil};0)\epsilon P \to \forall l.(l\epsilon\mathsf{vec}(U,n) \to (l;n)\epsilon P)$.
For the base case, we can easily prove
$\forall x_2.(x_2\epsilon\mathsf{vec}(U,n_2) \to (\mathsf{nil}(\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,0+n_2)))$.
For the step case, assume (IH)
$\forall x_2.(x_2\epsilon\mathsf{vec}(U,n_2) \to y(\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,m+n_2))$,
we want to show that
$\forall x_2.(x_2\epsilon\mathsf{vec}(U,n_2) \to (\mathsf{cons}\ m\ u\ y)(\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,\mathsf{S}m+n_2))$.
We know that
$(\mathsf{cons}\ m\ u\ y)(\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2 \to_\beta^*$
$\mathsf{cons}(m+n_2)\ u\ (\mathsf{app}\ m\ n_2\ y\ x_2) \to_\beta^*$
$\mathsf{cons}\ (m+n_2)\ u\ (y\ (\lambda m'.\lambda h.\lambda t.\mathsf{cons}(m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2))$.
By (IH), we know that
$y(\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,m+n_2)$.
By lemma 17 $\mathsf{cons}\ (m+n_2)\ u\ (y\ (\lambda m'.\lambda h.\lambda t.\mathsf{cons}(m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2))\epsilon\mathsf{vec}(U,\mathsf{S}(m+n_2))$. Thus $(\mathsf{cons}\ m\ u\ y)(\lambda m'.\lambda h.\lambda t.\mathsf{cons}\ (m'+n_2)\ h\ (\mathsf{app}\ m'\ n_2\ t\ x_2))x_2\epsilon\mathsf{vec}(U,\mathsf{S}(m+n_2))$. Of course, we assume we have $\mathsf{S}(m+n_2) = \mathsf{S}m+n_2$, so we have the proof.

□

Theorem 29 (Associativity)**.** $\vdash \forall(n_1.n_2.n_3.v_1.v_2.v_3).(n_1\epsilon\mathsf{Nat} \to n_2\epsilon\mathsf{Nat} \to n_3\epsilon\mathsf{Nat} \to v_1\epsilon\mathsf{vec}(U,n_1) \to v_2\epsilon\mathsf{vec}(U,n_2)) \to v_3\epsilon\mathsf{vec}(U,n_3) \to$
$\mathsf{app}\ n_1\ (n_2+n_3)\ v_1\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (n_1+n_2)\ n_3\ (\mathsf{app}\ n_1\ n_2\ v_1\ v_2)\ v_3$

Proof. Assume $n_1\epsilon\mathsf{Nat}, n_2\epsilon\mathsf{Nat}, n_3\epsilon\mathsf{Nat}, v_2\epsilon\mathsf{vec}(U,n_2)), v_3\epsilon\mathsf{vec}(U,n_3)$. We want to show
$\forall v_1.(v_1\epsilon\mathsf{vec}(U,n_1) \to \mathsf{app}\ n_1\ (n_2+n_3)\ v_1\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (n_1+n_2)\ n_3\ (\mathsf{app}\ n_1\ n_2\ v_1\ v_2)\ v_3)$.
Let $P := \iota(y;z).(\mathsf{app}\ z\ (n_2+n_3)\ y\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (z+n_2)\ n_3\ (\mathsf{app}\ z\ n_2\ y\ v_2)\ v_3)$. We instantiate the C in $\mathsf{Ind}(U,n_1)$ with P, by comprehension we have
$(\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (y;m)\epsilon P \to (\mathsf{cons}\ m\ u\ y;\mathsf{S}m)\epsilon P)) \to (\mathsf{nil};0)\epsilon P \to \forall l.(l\epsilon\mathsf{vec}(U,n) \to (l;n)\epsilon P)$.
So we just need to prove base case:
$\mathsf{app}\ 0\ (n_2+n_3)\ \mathsf{nil}\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (0+n_2)\ n_3\ (\mathsf{app}\ 0\ n_2\ \mathsf{nil}\ v_2)\ v_3$
and step case:

$\forall y.\forall m.\forall u.(m\epsilon\mathsf{Nat} \to u\epsilon U \to (\mathsf{app}\ m\ (n_2{+}n_3)\ y\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (m{+}n_2)\ n_3\ (\mathsf{app}\ m\ n_2\ y\ v_2)\ v_3) \to$ $(\mathsf{app}\ \mathsf{S}m\ (n_2+n_3)\ (\mathsf{cons}\ m\ u\ y)\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (\mathsf{S}m+n_2)\ n_3\ (\mathsf{app}\ \mathsf{S}m\ n_2\ (\mathsf{cons}\ m\ u\ y)\ v_2)\ v_3))$.
For the base case, $\mathsf{app}\ 0\ (n_2{+}n_3)\ \mathsf{nil}\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) \to_\beta^* \mathsf{app}\ n_2\ n_3\ v_2\ v_3 \leftarrow_\beta^* \mathsf{app}\ (0{+}n_2)\ n_3\ (\mathsf{app}\ 0\ n_2\ \mathsf{nil}\ v_2)\ v_3$.
For the step case, we assume $\mathsf{app}\ m\ (n_2+n_3)\ y\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) = \mathsf{app}\ (m{+}n_2)\ n_3\ (\mathsf{app}\ m\ n_2\ y\ v_2)\ v_3$(IH), we want to show
$\mathsf{app}\ \mathsf{S}m\ (n_2+n_3)\ (\mathsf{cons}\ m\ u\ y)\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) =$
$\mathsf{app}\ (\mathsf{S}m+n_2)\ n_3\ (\mathsf{app}\ \mathsf{S}m\ n_2\ (\mathsf{cons}\ m\ u\ y)\ v_2)\ v_3$(Goal).
We know that
$\mathsf{app}\ \mathsf{S}m\ (n_2{+}n_3)\ (\mathsf{cons}\ m\ u\ y)\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3) \to_\beta^* \mathsf{cons}(m{+}n_2{+}n_3)\ u\ \boxed{(\mathsf{app}\ m\ (n_2+n_3)\ y\ (\mathsf{app}\ n_2\ n_3\ v_2\ v_3))}$.
The right hand side of the (Goal) can be reduced to $\mathsf{cons}(m{+}n_2{+}n_3)\ u\ \boxed{(\mathsf{app}\ (m+n_2)\ n_3\ (\mathsf{app}\ m\ n_2\ y\ v_2)\ v_3)}$.
So (IH) is enough to give us the (goal). □

7 Summary

We present System $\mathfrak{G}$, which is suitable to serve as a foundational system for both formalize mathematics and reasoning about programs. We develop Peano's axioms and Vector encoding in System $\mathfrak{G}$ as evidents for its potentials. The usefulness of $\mathfrak{G}[t]$ is not obvious in this chapter. However, if we switch the data-type encoding to Church encoding, then $\mathfrak{G}[t]$ will become more useful. The existence of $\mathfrak{G}[t]$ shows that we can understand the phenomina of polymorphic-dependent type with $\mathfrak{G}$. It is important to note that the essential difference between System $\mathfrak{G}$ and PTS style system is that computation at formula level is currently not possible in $\mathfrak{G}$. We think that computation on formula may be better treated as external feature of a logic system instead of support it internally within the logical system. Of course, it is not a conclusive assertion, more research will be needed to explore this issue.

When comparing to usual typed functional programming language, the set in System $\mathfrak{G}$ is more precise than the notion of type in typed functional programming language. A direct consequence is that it is impossible to fully automate the reasoning with $\mathfrak{G}$. However, a degree of automation is still possible, together with human guidence, it would be an attracting tool to have besides the usual type system. Again, we would have to leave this to future research to see whether if it is feasible to incorporate System $\mathfrak{G}$ into a typed functional programming language.

References

[1] Hendrik Pieter Barendregt. *The lambda calculus: Its syntax and semantics*, volume 103. North Holland, 1985.

[2] Henk Barendregt, S. Abramsky, D. M. Gabbay, T. S. E. Maibaum, and H. P. Barendregt. Lambda calculi with types. In *Handbook of Logic in Computer Science*, pages 117–309. Oxford University Press, 1992.

[3] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca. (semi)-separability of finite sets of terms in scott's D_∞-models of the lambda-calculus. In *Proceedings of the Fifth Colloquium on Automata, Languages and Programming*, pages 142–164, London, UK, UK, 1978. Springer-Verlag.

[4] Gottlob Frege. The basic laws of arithmetic: Exposition of the system, translated and edited with an introduction by montgomery furth, 1967.

[5] Peng Fu, Aaron Stump, and Jeffrey Vaughan. A framework for internalizing relations into type theory. In *PSATTT'11: International Workshop on Proof-Search in Axiomatic Theories and Type Theories*, 2011.

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

[7] William S Hatcher. *The logical foundations of mathematics*, volume 10. Pergamon Press Oxford, 1982.

[8] Jean-Louis Krivine. Lambda-calculus types and models. 2002.

[9] Daniel Leivant. Reasoning about functional programs and complexity classes associated with type disciplines. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 460–469. IEEE, 1983.

[10] Willard V Quine. New foundations for mathematical logic. *The American mathematical monthly*, 44(2):70–80, 1937.

[11] G. Takeuti. *Proof Theory*. Volume 81 of Studies in logic and the foundations of mathematics, ISSN 0049-237X. North-Holland Publishing Company, 1975.

[12] A.N. Whitehead and B. Russell. *Principia mathematica. 2.1927*. Principia Mathematica. Cambridge University Press, 1927.

[13] Ernst Zermelo. Untersuchungen über die grundlagen der mengenlehre. i. *Mathematische Annalen*, 65(2):261–281, 1908.