An Intuitionistic Set-theoretical Model of the Extended Calculus of Constructions

Masahiro Sato

Contents

1 Introduction

1.1 Type Theory

Lambda calculus is a mathematical way to represent computations or programs. The main purpose of type theory is to classify lambda terms[22, 9]. This is also relevant to logic, as lambda calculus can express not only programs but also proofs. This fact is well known as the Curry–Howard Correspondence[12, 14]. Yet another interpretation would see types as sets. We write '$t : T$' if the term t has the type T. This can be interpreted in several ways.

- Set Theoretical View
  $t : T$ means that t is an element of the set T.
- Programming Language View
  $t : T$ means that t is a value which has the type T.
- Proof Theoretical View
  $t : T$ means that t is a proof of the proposition T.

Type systems also have *typing rules*. By applying typing rules, one can infer the type of a term.

In order to have a logical meaning, a type system must be consistent. Consistency is the property that there doesn't exist a term t whose type is the *empty type*. The *empty type* is interpreted as the false proposition or the empty set.

A general way to show the consistency of a type system is to construct its model. A model interprets term and types into some semantical domain. Let us write the interpretation of $t : T$ as $[\![t : T]\!]$. For instance, in the set-theoretical model, the judgment '$t : T$' is interpreted into the proposition $[\![t : T]\!] :=$ '$t \in T$', i.e. a type is interpreted into some set, and a term is interpreted into an element. The model is said to be sound if the interpretation $[\![t : T]\!]$ is true whenever $t : T$ holds. Conversely, the model is said to be complete if $t : T$ holds whenever the interpretation $[\![t : T]\!]$ is true. Constructing a complete model of type theory is not easy, and the existing ones are complicated and hard to understand. This thesis attempts to construct an intuitive and simple model of type theory, yet close to 'completeness'.

1.2 Intuitionistic Logic

Almost all mathematicians use *classical logic* when proving mathematical theorems. However, the proof theory underlying type theory is that of *intuitionistic logic*[26] rather than classical logic. Intuitionism was first introduced by Brouwer, and formalized by Heyting later. Intuitionistic logics differ from classical logic in that the principle of excluded middle $P \vee \neg P$ does not hold in general. Therefore, some theorems cannot be proved in intuitionistic logic such as the intermediate value theorem, among others. However, by using intuitionistic logic, we get a proof which is more constructive. For instance, consider the following proposition.

Proposition 1.1. *There are irrational numbers x and y such that x^y is a rational number.*

Proof. We will prove two cases, whether $\sqrt{2}^{\sqrt{2}}$ is a rational number or not.

- If $\sqrt{2}^{\sqrt{2}}$ is a rational number.
  It is clear by $(x, y) = (\sqrt{2}, \sqrt{2})$.

- If $\sqrt{2}^{\sqrt{2}}$ is not a rational number.
  Let $(x, y) = (\sqrt{2}^{\sqrt{2}}, \sqrt{2})$, then

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\times\sqrt{2}} = \sqrt{2}^{2} = 2.$$

  Hence the proposition holds in this case.

□

In this proof, we used the excluded middle :$\sqrt{2}^{\sqrt{2}}$ is a rational number or not. In order to prove this proposition in intuitionistic logic, we would have to know which case is true.

Many mathematical theories have been developed using intuitionistic logic, such as constructive analysis[11], constructive set theory[6], constructive topological theory[4](formal topology[24]).

1.3 Soundness of Type Systems

There are various models of type system. Werner's Set-theoretical model [28] provides an intuitive model of ECC. It combines a functional view of predicative universes with a collapsed view of the impredicative sort Prop. However this model of Prop is so coarse that the principle of excluded middle $P \vee \neg P$ holds in it.

In this paper, we construct a set-theoretical model of ECC in which the principle of excluded middle $P \vee \neg P$ doesn't hold, and thus more complete.

ECC(the Extended Calculus of Constructions) extends CC with a hierarchy of predicative sorts Type_i and strong sums $\Sigma x : A.B$. CC(the Calculus of Constructions [12]) is a pure type system [8] with two sorts, impredicative Prop and predicative Type.

In [28], Werner provides a remarkably simple model of ECC without strong sums. In this model, $\lambda x : A.t$ is interpreted by a set-theoretical function for predicative sorts. Yet such a simple approach is known to fail for impredicative sorts as it runs afoul of Reynolds' paradox [23]. Therefore, the model for Prop is two-valued, and proofs are not distinguished. Hence the principle of excluded middle $P \vee \neg P$ is valid in this model. This simple approach is to be contrasted with Luo's original model of ECC which uses ω-sets [17]. This is the drawback of simplicity while this approach avoids many complications of more precise models, it is at times counter-intuitive, as it completely ignores the intuitionistic aspect of CC. Our goal has been to recover the intuitionistic part of CC without increasing the complexity of the model. To do this, we interpret Prop into some topological space. Topological spaces are instances of Heyting algebras. Despite the fact that the interpretation of Prop is many valued, we avoid Reynolds' paradox by making the interpretation of proofs undistinguished. Due to proof-irrelevance, this model still validates some propositions that are not provable, hence this model is still not complete. However this is sufficient to exclude many classical propositions such as the principle of excluded middle $P \vee \neg P$. Note that, to make the model coherent, we had to slightly restrict the type system. We believe the scope is still sufficient to make this model practical, but hope to remove these restrictions in the future.

This model is parametrized by a topological space $(X, \mathcal{O}(X))$ and a point $p \in X$, which is called the *reference point*[1]. By replacing the parameters of the model, we can make it more or less precise. For instance if its parameters are the topological space $(\{\cdot\}, \{\phi, \{\cdot\}\})$ and the reference point '$\cdot$', we obtain a model of classical logic, which is the coarsest one. It suffices to add one

[1]Our proof of soundness requires this reference point to satisfy a condition, which is called the *point condition*.

more point and shift the reference point to invalidate the principle of excluded middle.

In section 2, we define the language of two type systems, $\lambda^{\to}$ and ECC. In section 3, we introduce axiomatic set theories, ZFC, IZF and CZF. In section 4, we introduce 'Coq', proof assistant based on type theory. In section 5, we give our set-theoretical interpretation of ECC, and prove its soundness. In section 6, we show some examples of the model. In section 7, we analyze how we avoid Reynolds' paradox.

2 Type Theory

Type systems are logical systems, where one proves typing judgment through typing derivations. A typing judgment is composed three object, a context, a term and a type.

- Term
  The 'term' means a proof, value or function. It is the subject of the judgment.

- Type
  The 'type' characterizes the term. In general, a term has at most one type.

- Context
  The 'context' assigns types to the variables that may appear in the term. It is a list of pairs of term and type. More precisely, a context has the form $x_1 : T_1; x_2 : T_2; \cdots ; x_n; T_n$ where $\{x_i\}_i$ is a distinct variable and T_i is a term for each i.

Type systems also have typing rules. Typing rules are to decide the type of a given term. We write the *judgment* $\Gamma \vdash t : T$ when the typing rules assign type T to term t under context Γ. If such a type T exists, the term t is said to be *typable*. Typability says that a term is valid with respect to the typing rules.

There are various type systems such as CC[12], Martin-Löf Type theory[19]. In next subsection, we introduce $\lambda^{\to}$, which is the simplest type system.

2.1 $\lambda^{\rightarrow}$

$\lambda^{\rightarrow}$ is the simplest type system. We introduce the definition of $\lambda^{\rightarrow}$.

Definition 2.1. $\lambda^{\rightarrow}$ is composed of terms, types and contexts.

1. Term
   Let X be a set of variables. We define lambda terms recursively as follows.
   - If $x \in X$ then x is a lambda term.
   - If t_1 and t_2 are lambda terms, then $(t_1 t_2)$ is a lambda term.
   - If $x \in X$, T is a type and t is a lambda term, then $\lambda x : T.t$ is a lambda term.
2. Type
   Let $\mathbb{B}$ be the set of base types. We define types recursively.
   - If $T \in \mathbb{B}$ then T is a type.
   - If T_1 and T_2 are types, then $T_1 \rightarrow T_2$ is a type.
3. Context
   - $[]$ is a context. It is called the empty context.
   - If Γ is a context, then $\Gamma; (x : T)$ is a context where x is a variable and T is a type.

Next, we introduce the typing rules of $\lambda^{\rightarrow}$. We show the three typing rules of $\lambda^{\rightarrow}$ in Table 1.

$\dfrac{x : T \in \Gamma}{\Gamma \vdash x : T}$	(axiom)
$\dfrac{\Gamma; (x : T_1) \vdash t : T_2}{\Gamma \vdash \lambda x : T_1.t : T_1 \rightarrow T_2}$	(abstraction)
$\dfrac{\Gamma \vdash f : T_1 \rightarrow T_2 \quad \Gamma \vdash t : T_1}{\Gamma \vdash ft : T_2}$	(apply)

Table 1: Typing Rule of $\lambda^{\rightarrow}$

The term '$\lambda x : T.t$' means the function of an argument x of type T which returns the value t. For instance, the term '$\lambda n : \mathbb{N}.n + 1$'[2] is the function from n to $n + 1$. '$\Gamma ; (x : A) \vdash t : B$' means that the term t has the type B when the type of the variable x is A. Therefore the type of $\lambda x : A.t$ is $A \rightarrow B$.

The term '$f\ t$' means the function application, i.e. '$f(t)$'. For instance, the term '$(\lambda n : \mathbb{N}.n + 1)\ 3$' means $3 + 1 (= 4)$. '$\Gamma \vdash f : A \rightarrow B$' means that f is a function from T_1 into T_2. Therefore the type of the term '$f\ t$' is T_2 when the type of the term t is T_1.

In order to clarify the image of terms, we introduce an equivalence relation on terms, which is called *beta equality*. Before giving the definition of beta equality, we define the substitution of a term.

Definition 2.2 (Substitution)**.** Let t and v be terms and x be a variable. The substitution $t[x\backslash v]$, which means 'replace x by v in t', is defined inductively as follows:

(i) If y is variable, then $y[x\backslash v] = \begin{cases} v & (y = x) \\ x & (otherwise), \end{cases}$

(ii) $(t_1\ t_2)[x\backslash v] = (t_1[x\backslash v])\ (t_2[x\backslash v])$,

(iii) $(\lambda x' : T.t')[x\backslash v] = \lambda x' : T.t'[x\backslash v] \quad$ (when $x \neq x'$).

Now, we are ready to define beta equality.

Definition 2.3 (Beta Equality)**.** The transformation of '$(\lambda x : A.t)\ a$' into '$t[x\backslash a]$' is called beta reduction. The beta equality $=_\beta$ is the smallest equivalence relation such that following conditions hold.

(i) $(\lambda x : A.t)\ a =_\beta t[x\backslash a]$.

(ii) If $t_1 =_\beta t'_1$ and $t_2 =_\beta t'_2$ then $t_1\ t_2 =_\beta t'_1\ t'_2$.

(iii) If $t =_\beta t'$ then $\lambda x : A.t =_\beta \lambda x : At'$.

We can interpret $\lambda^{\rightarrow}$ in the following ways.

- Set Theoretical View

[2]When assuming that the symbols the symbol '+' and 1 are defined as terms.

  - $t : T$ means that t is an element of the set T.
  - $f : T_1 \to T_2$ means that f is a function from T_1 into T_2.
  - Beta reduction means applying the function.

- Programming Language View
  - $t : T$ means that t is a value whose type is T.
  - $f : T_1 \to T_2$ means that f is a program with argument of type T_1 which returns a value of type T_2.
  - Beta reduction means executing the program.

- Proof Theoretical View
  - $t : T$ means that t is a proof of the proposition T.
  - $f : T_1 \to T_2$ means that f is a proof of the proposition $T_1 \Rightarrow T_2$.
  - Beta reduction corresponds to cut-elimination.

2.2 Definition of ECC

We define the type system ECC as follows (omitting strong sums, as in [28]).

Definition 2.4 (Term)**.**

- x is a term for $x \in V$.
- If t_1 and t_2 are terms, then $t_1 t_2$ is a term.
- If t and T is are terms, and $x \in V$ then, $\lambda x : T.t$ is a term.
- If T_1 and T_2 are terms, and $x \in V$ then $\forall x : T_1.T_2$ is a term.
- $\mathrm{Prop}, \mathrm{Type}_i$ are terms ($i = 0, 1, 2, 3, 4, ...$).

Type_0 is named "Set" in Coq.

Definition 2.5 (Context)**.**

- $[]$ is a context.
- If Γ is a context, and T is a term and $x \in V$, then $\Gamma; (x : T)$ is a context.

In the absence of dependencies, order is irrelevant in contexts. We show the typing rules of ECC in Table 2. They are standard, except that we restricted the PI-Type rule in the case P : Prop and Q : Prop, and removed the subtyping rule from Prop to Type. The unrestricted Prop-Prop PI-Type rule creates difficulties when building an intuitionistic model, and if we do not remove the subtyping rule it becomes possible to use the Prop-Type case of the PI-Type rule in place of the restricted Prop-Prop case, which would make the model incoherent. We believe these restrictions are reasonable, as the proof component is seldom used in the PI-Type rule, with the notable exception of the generic statement of proof-irrelevance. Removing the subtyping between Prop and Type does not change the expressive power, as it is still possible to explicitly duplicate properties using Type to Prop. We hope to solve these problems in the future, and allow the standard typing rules.

$\Gamma \vdash \mathrm{Prop} : \mathrm{Type}_i$	$\Gamma \vdash \mathrm{Type}_i : \mathrm{Type}_{i+1}$	(Axiom)
$\dfrac{\Gamma \vdash A : \mathrm{Type}_i}{\Gamma \vdash A : \mathrm{Type}_{i+1}}$		(Subtyping)
$\dfrac{\Gamma \vdash A : \mathrm{Type}_i \quad \Gamma; (x : A) \vdash B : \mathrm{Type}_j}{\Gamma \vdash \forall x : A.B : \mathrm{Type}_{\max(i,j)}}$	$\dfrac{\Gamma \vdash A : \mathrm{Prop} \quad \Gamma; (x : A) \vdash B : \mathrm{Type}_j}{\Gamma \vdash \forall x : A.B.\mathrm{Type}_j}$	(PI-Type)
$\dfrac{\Gamma \vdash A : \mathrm{Type}_i \quad \Gamma; (x : A) \vdash Q : \mathrm{Prop}}{\Gamma \vdash \forall x : A.Q : \mathrm{Prop}}$	$\dfrac{\Gamma \vdash P : \mathrm{Prop} \quad \Gamma \vdash Q : \mathrm{Prop} \quad x \text{ does not appear in } Q}{\Gamma \vdash \forall x : P.Q : \mathrm{Prop}}$	
$\dfrac{\Gamma; (x : A) \vdash t : B \quad \Gamma \vdash \forall x : A.B : \mathrm{Type_i}}{\Gamma \vdash \lambda x : A.t : \forall x : A.B}$	$\dfrac{\Gamma; (x : A) \vdash t : B \quad \Gamma \vdash \forall x : A.B : \mathrm{Prop}}{\Gamma \vdash \lambda x : A.t : \forall x : A.B}$	(Abstract)
$\dfrac{\Gamma \vdash u : \forall x : A.B \quad \Gamma \vdash v : A}{\Gamma \vdash (uv) : B[x\backslash v]}$		(Apply)
$\dfrac{(x : A) \in \Gamma \quad \Gamma \vdash A : \mathrm{Type}_i}{\Gamma \vdash x : A}$	$\dfrac{(x : A) \in \Gamma \quad \Gamma \vdash A : \mathrm{Prop}}{\Gamma \vdash x : A}$	(Variable)
$\dfrac{\Gamma \vdash x : A \quad A =_\beta B}{\Gamma \vdash x : B}$		(Beta Equality)

Table 2: Typing Rule of ECC

In Table 2, $=_\beta$ denotes *beta equality* and $B[x\backslash v]$ denotes substitution. They are defined in Definitions 2.6 and 2.7 below.

Definition 2.6 (Substitution)**.** Let t and v be terms and x be a variable. The substitution $t[x\backslash v]$, which means v replaces x in t, is defined inductively as follows:

(i) If y is variable, then $y[x\backslash v] = \begin{cases} v & (y = x) \\ x & (otherwise) \end{cases}$,

(ii) $(t_1 t_2)[x\backslash v] = (t_1[x\backslash v])(t_2[x\backslash v])$,

(iii) $(\lambda x' : T.t')[x\backslash v] = \lambda x' : (T[x\backslash v]).t'[x\backslash v] \quad (\text{when } x \neq x')$,

(iv) $(\forall x' : T_1.T_2)[x\backslash v] = \forall x' : (T_1[x\backslash v]).(T_2[x\backslash v])$,

(v) $(\text{Prop})[x\backslash v] = \text{Prop}$,

(vi) $(\text{Type}_i)[x\backslash v] = \text{Type}_i \quad (i = 1, 2, 3, ...)$.

Definition 2.7 (Beta Equality)**.** Let $=_\beta$ be the smallest equivalence relation such that following conditions hold.

(i) $(\lambda x : A.t)\ a =_\beta t[x\backslash a]$.

(ii) If $t_1 =_\beta t'_1$ and $t_2 =_\beta t'_2$ then $t_1 t_2 =_\beta t'_1 t'_2$.

(iii) If $t =_\beta t'$ and $A =_\beta A'$ then $\lambda x : A.t =_\beta \lambda x : A' t'$.

(iv) If $A =_\beta A'$ and $B =_\beta B'$ then $\forall x : A.B =_\beta \forall x : A' B'$.

In this type system, the new sort Prop is introduced. It represents the set of propositions.

Definition 2.8.

1. Propositional Term
   The term P is called a propositional term for Γ iff $\Gamma \vdash P : \text{Prop}$ is derivable.

2. Proof Term
   The term p is called a proof term for Γ iff $\Gamma \vdash p : P$ is derivable for some P which is a propositional term for Γ

3. Provable Propositional Term
   The term P is called a provable propositional term for Γ iff P is a propositional term for Γ and there exists p such that $\Gamma \vdash p : P$ is derivable.

Lemma 2.9. *The following statements are equivalent.*

- *p is a proof(resp. propositional) term for the context $\Gamma; (x : U); \Delta$.*
- *$p[x\backslash u]$ is a proof(resp. propositional) term for the context $\Gamma; \Delta[x\backslash u]$.*

This lemma is consequence of the following proposition.

Proposition 2.10. *If $\Gamma \vdash u : U$ is derivable, then $\Gamma; (x : U); \Delta \vdash t : T$ is derivable if and only if $\Gamma; \Delta[x\backslash u] \vdash t[x\backslash u] : T[x\backslash u]$ is derivable.*

Proposition 2.10 can be proved in the same way as in [20].

Lastly, here are some notations allowing to use other logical symbols [9].

Definition 2.11.

$$\begin{aligned}
A \to B &:= \forall x : A.B \quad \text{(when `$x$' does not occur freely in `B')}, \\
\bot &:= \forall P : \mathrm{Prop}.P, \\
\neg A &:= A \to \bot, \\
A \wedge B &:= \forall P : \mathrm{Prop}.(A \to B \to P) \to P, \\
A \vee B &:= \forall P : \mathrm{Prop}.(A \to P) \to (B \to P) \to P, \\
\exists x : A.Q &:= \forall P : \mathrm{Prop}.(\forall x : A.(Q \to P)) \to P, \\
A \leftrightarrow B &:= (A \to B) \wedge (B \to A), \\
x =_A y &:= \forall Q : (A \to \mathrm{Prop}).Q\ x \leftrightarrow Q\ y.
\end{aligned}$$

3 Set Theory in Intuitionistic Logic

Before working on the models of type theory, I was interested in the formalization of logic. This led me to implement ZFC and IZF in Coq. Before looking at this formalization, we discuss axiomatizations of set theory. There are several axiomatical set theories, such as ZFC, IZF and CZF. The most familiar one is ZFC(Zermelo-Fraenkel set theory and axiom of Choice). We will also discuss IZF(Intuitionistic Zermelo-Fraenkel) and CZF(Constructive Zermelo-Fraenkel)[6]. Some axiom of ZFC can imply excluded middle, IZF prevents it. Hence IZF is weaker than ZFC.

3.1 ZF

The most familiar axiomatical set theory is ZF. We introduce the axiom of ZF[3].

- Extensionality Axiom
  $\forall z, (z \in x \Leftrightarrow z \in y) \;\Rightarrow\; x = y$
- Pairing Axiom
  If x and y are sets, then $\{x, y\}$ is a set.
- Union Axiom
  If A is a set, then its union $\bigcup A$ is a set.
- Power Set Axiom
  If A is a set, then its power set $\mathcal{P}(A)$ is a set.
- Infinity Axiom
  There exists the set $\mathbb{N}$ of all natural number.
- Comprehension Scheme
  Let $P(x)$ be any formula with a variable x. If A is a set, then $\{x \in A | P(x)\}$ is a set.
- Replacement Scheme
  Let $F(x)$ be any function on sets, then the set $F(A)$ exists for any set A, i.e. the image of A by F
- Foundation Axiom
  $\in$ is a well-founded relation on sets.

3.2 IZF

IZF is the intuitionistic version of ZF. Here are the differences between ZF and IZF.

- Remove the Foundation Axiom, but introduce the following new axiom

  $$\forall a((\forall x \in a, \phi(x)) \Rightarrow \phi(a)) \Rightarrow \forall y, \phi(y) \quad \text{(for any formula } \phi(x)\text{)}$$

  which is called the *Set Induction Scheme.*

[3]For detail, see [16].

- Replace the Replacement Scheme into the following new axiom

  $$\forall x \in a \exists y, \theta(x,y) \Rightarrow \exists b, \forall x \in a \exists y \in b, \theta(x,y) \quad \text{(for any formula } \theta(x,y)\text{)}$$

  which is called the *Collection Scheme*

It is known that the Foundation Axiom and the Set Induction Scheme are equivalent in classical logic. However the Foundation Axiom is properly stronger than the Set Induction Axiom in intuitionistic logic, since the Foundation Axiom can imply the principle of excluded middle[15]. Similarly, it is known that the Replacement Scheme and the Collection Scheme are equivalence in classical logic, but the Collection Scheme is properly stronger than the Replacement Scheme in intuitionistic logic. Moreover, the axiom of choice can imply the principle of excluded middle whereas the countable axiom of choice cannot.

3.3 CZF

CZF is a predicative version of IZF. CZF excludes impredicative sets, hence CZF is weaker than IZF. Here are the differences between ZF and IZF.

- Remove the Powerset Axiom, but introduce a new axiom

  $$\begin{aligned}&\exists c \forall u (\forall x \in a \exists y \in b \phi(x,y,u) \Rightarrow \\ &\qquad \exists d \in c(\forall x \in a \exists y \in d \phi(x,y,u) \wedge \forall y \in d \exists x \in a \phi(x,y,u)))\end{aligned}$$

  which is called *Subset Collection.*

- Restrict the Separation Scheme, i.e. assume the scheme

  $$\forall a \exists b \forall c (c \in b \Leftrightarrow c \in a \wedge \phi(c)) \quad \text{(where } \phi(x) \text{ is } \Delta_0\text{-formula).}$$

  which is called the *Bounded Separation Scheme.*

A Δ_0-formula is a formula whose quantifiers are all bounded. The formulas $\forall x \in A, ...$ and $\exists x \in A, ...$ are called bounded quantifiers.

The Powerset Axiom is equivalent to Subset Collection and the following lemma holds.

$$\mathcal{P}(\{\phi\}) = \{\phi, \{\phi\}\}^4$$

[4] $\mathcal{P}(X)$ is power set of X.

Peter Aczel worked on the relationship between type theory and CZF[1, 2, 3], to construct a model of CZF.

In CZF, many important sets for general mathematics cannot be constructed, for instance open set families in discrete topological spaces. Hence, several axioms are sometimes assumed with CZF, for instance REM(Regular Extension Axiom[3, 5]), SGA(Set Generated Axiom[5]), some restricted Axiom of Choice(Countable Axiom of Choice, $\Sigma\Pi$-AC[3]) and more.

Moreover, there is a topological theory in CZF, which is called *Formal Topology*[24, 4].

4 Formalizing Set Theory in Coq

4.1 Proof Assistant

Proof assistants are like a computer programming language. However in proof assistants, we write proof code in stead of algorithm code. The main role of proof assistants are to check the correctness of the proof code[5]. There are various proof assistant such as Coq and Agda. Especially, some proof assistants use type theory in checking the proof. For instance, Coq uses the type system CIC. Proofs written by human may have some mistakes. Proof assistants can prevent them. Moreover, proof assistants can remember many hypotheses while proving theorems. Hence it makes easier proving such complex theorems.

4.2 Type system of Coq

The Coq proof assistant is based on CIC, the Calculus of Inductive Constructions. It extends the Calculus of Constructions with (co)Inductive definitions [13, 21] and a hierarchy of universes.

Coq has the types ‘Type_i’, ‘Prop’ and ‘Set’ as its base type, which are called *sorts*. In Coq, we write

```
t : T
```

to denote t has the type T. And, the types(terms) $\forall x : A.B$, $A \to B$ and $\lambda x : A.t$ are written as follows.

[5]Note that in general proof assistants do not prove theorems automatically.

```
forall x : A.B
A -> B
fun x : A => t
```

Next, we introduce the definition of inductive type. The types 'bool' and 'nat' are defined as inductive types as follows.

```
Inductive bool : Set :=
| true : bool
| false : bool
.
Inductive nat : Set :=
| O : nat
| S : nat -> nat
.
```

In this definition, the 'bool' type has only two term 'true' and 'false'. On the other hand, the term 'S' in the definition of 'nat' means successor, hence every natural number is defined recursively as follows.

$$\begin{array}{rcl} 0 & := & \mathrm{O} \\ 1 & := & \mathrm{S\ O} \\ 2 & := & \mathrm{S\ (S\ O)} \\ 3 & := & \mathrm{S\ (S\ (S\ O))} \\ & & \vdots \end{array}$$

The language system of Coq also has *match statement*.

```
fun b : bool :=
  match b with
  | true => S O
  | false => O
  end
```

This term denotes the function which maps the value 'false' into the value '0', and maps the value 'true' into the value '1'.

Other logical symbol '$\lor, \land, \neg$' are written as follows.

```
A \/ B
A /\ B
~A
```

Inductive definition in CIC is very strong. In fact, the above three logical symbols can be defined by inductive definitions.

Coq also has Axiom declarations. It enables one make the theory of Coq stronger. For instance, if we insert the following text

```
Axiom classic : forall P : Prop, P \/ ~P.
```

then Coq's theory becomes classical logic[6]. Note that using axioms incurs some problems, since making the theory of Coq stronger may break consistency.

4.3 Formalizing set theory

There are several attempts at implementing set theory in Coq. Here are some examples.

- Benjamin Werner (IZF + ΣΠI-AC + REA + ...)[27]
- Bruno Barras(Finite ZF)[10]
- Carlos Simpson(ZFC)[25]
- Guillaume Alexandre(ZF)[7]
- Our Implementation(IZF)

4.3.1 Werner's implementation

Werner started from Aczel's model of CZF in Martin-Löf type theory[7]. This model show that Martin-Löf Type Theory is strong enough to encode CZF.

On the other hand, CIC has an impredicative sort Prop. Hence, CIC can also represent IZF

He defines a type of sets 'Ens' following Aczel's construction as follows.

```
Inductive Ens : Type :=
  sup : forall A : Type, (A -> Ens) -> Ens.
```

[6]Coq defaults to intuitionistic logic.

[7]See[1, 2, 3].

This type ‘Ens’ contains the functions from any type ‘A’ to ‘Ens’ itself. For instance, let A be the bottom type $\bot$, then there exists f whose type is ‘$\bot \to$ Ens’. Then ‘Ens $\bot$ f’ represents the empty set. Next, we can define a paring set $\{a, b\}$ as follows.

```
Definition Pair E1 E2 :=
  sup bool (fun b : bool =>
    match b with
    | true => E1
    | false => E2
    end
  ).
```

4.3.2 Barras’ implementation

He defines a set as the following inductive type hf.

```
Inductive hf : Set :=
  HF : list hf -> hf.
```

The term HF generates a set from a list of set.

The following code is the definition of the empty set and pair.

```
Definition empty := HF nil.
Definition pair x y := HF (x :: y :: nil).
```

Next, he defines the relations $=$ and $\in$ as follows.

```
Definition eq_hf x y :=
  List.forallb (fun x’ => List.existsb (fun y’ => eq_hf x’ y’) y) x &&
  List.forallb (fun x’ => List.existsb (fun y’ => eq_hf x’ y’) x) y

Definition in_hf x y := List.existsb (fun y’ => eq_hf x y’) y.
```

However, we can only define elements of V_ω[8]. We cannot define infinite sets in this implement. As a result, the relation $\in$ is decidable.

[8]See [16]

4.3.3 Simpson's implementation

```
Definition E := Type.

Axiom R : forall x : E, x -> E.
Axiom R_inf : forall (x : E) (a b : x), R a = R b -> a = b.

Definition inc (x y : E) := exists a : y, R a = x.
```

In this definition, every type(i.e. term of sort) is treated as a set. However, this implementation may be stronger than IZF or ZFC.

4.3.4 Alexandre's implementation

His implementation is direct and intuitive.

Firstly, he implements the type of sets as follows.

```
Parameter E : Set.
Parameter In : E -> E -> Prop.
```

The type 'E' means the type of set, and 'In' means the membership relation '$\in$'. We write a part of his code.

```
Axiom
  axs_extensionnalite :
    forall v0 v1 : E,
    (forall v2 : E, In v2 v0 <-> In v2 v1) -> v0 = v1.

Variable paire : E -> E -> E.
Axiom
  axs_paire : forall v0 v1 v3 : E,
  In v3 (paire v0 v1) <-> v3 = v0 \/ v3 = v1.
```

The axiom 'axs_extensionnalite' in above code means extensionality axiom. And, 'paire' and 'asx_paire' mean both axiom of paring. In this implementation, the axiom is written directly. However, one needs two axiom statements per axiom of the theory, 'paire' and 'axs_paire'. The axiom 'paire' introduces a function which maps two sets a and b into the set $\{a, b\}$. The axiom 'axs_paire' expresses the conditions on 'paire'.

4.3.5 Our Implementation

In our implementation, we formalize IZF by coding each IZF axiom by an axiom statement in a way similar to G.Alexander's. However, we accept the following two axioms.

```
Definition Unique (P : SET -> Prop) :=
  (exists x, P x) /\
  (forall a b, P a /\ P b  -> a = b).
Axiom UniqueOut (P : SET -> Prop) : Unique P -> SET.
Axiom HUniqueOut :
  forall (P : SET -> Prop) (unq : Unique P),
  P (UniqueOut P unq).
```

Therefore we succeed to reduce the number of axiom sentence. For instance, the following code represent a axiom of pair.

```
Definition IsPair a b c :=
  forall x, In x c <-> x = a \/ x = b.
Axiom axiom_of_pair :
  forall a b, exists c, IsPair a b c.

Theorem UniqueIsPair : forall a b, Unique(fun c => IsPair a b c).
Proof.
  .
  .
  .
Qed.

Definition Pair a b :=
  UniqueOut (fun c => IsPair a b c) (UniqueIsPair a b).
```

In this code, there is only one axiom. It can be proved in ZF that the set which contains only a and b exists uniquely. Hence, we could reduce the number of axioms by using 'UniqueOut'.

4.4 Problem of consistency

Yet our implementation still uses many axiom sentences. Therefore it is necessary to prove the consistency of those axiom statements. And it is necessary to prove the independence from the principle of excluded middle, since we implement IZF. Hence, our next task is to construct a model of this type system.

5 Interpretation

In section 4, we succeeded in formalizing the implementation of IZF in Coq. However, we used some axiom sentence in this implementation. Is this implementation also consistent, and independent from the principle of excluded middle? To prove this, we construct a model which we restrain to intuitionistic logic. For simplicity, we construct only a model of ECC in spite of Coq's type system being CIC[9].

5.1 Lattice

In this paper, we use *Heyting algebras*[18, 26]. Heyting algebras provide models of intuitionistic logic. Topological spaces form Heyting algebras, and as such provide models of intuitionistic logic too[26]. We give a definition of lattice and Heyting algebra as follows.

Definition 5.1 (Lattice)**.** Let $(A, \leq)$ be a partial order set(i.e. reflexivity, antisymmetry, and transitivity). $(A, \leq)$ is called Lattice when any two elements a and b of A have a supremum '$a \sqcup b$' and infimum '$a \sqcap b$', which are called join and meet[10]. A lattice is also called *complete lattice* if every subset S of A has supremum '$\bigsqcup S$' and infimun '$\bigsqcap S$'. If a lattice has an *exponential operator* a^b such that

$$x \leq z^y \Leftrightarrow x \sqcap y \leq z$$

holds, then we call it Heyting Algebra.

The following lemma show that complete lattice is stronger than Heyting algebra.

Lemma 5.2. *If $(A, \leq)$ is a complete lattice, then this is also a Heyting algebra.*

[9]Constructing a model of CIC is one of our future goals.

[10]We use the lattice operation symbols join '$\sqcup$' and meet '$\sqcap$' instead of '$\vee$' and '$\wedge$', since we use these in another way in this paper.

Proof. Let y^x be

$$\bigsqcup\{t | t \sqcap x \leq y\}.$$

$\square$

Lemma 5.3. *For any set X, the topological space $(X, \mathcal{O}(X))$ is a Heyting algebra, moreover it is a complete lattice.*

Proof. In fact let $a \leq b$ be $a \subset b$, and define each operation as follows:

$$\begin{aligned}
\mathbb{I} &:= X, \\
\mathbb{O} &:= \phi, \\
\bigsqcup S &:= \bigcup S, \\
\bigsqcap S &:= \bigsqcup\{t \mid \forall s \in S, t \leq s\} = \left(\bigcap S\right)^{\circ} \quad (\textit{where } A^{\circ} \textit{ is the interior of } A), \\
b^a &:= \bigsqcup\{t \mid t \sqcap a \leq b\}.
\end{aligned}$$

$\square$

The following lemma states well known properties of complete Heyting algebras.

Lemma 5.4. *Let $(A, \leq)$ be a complete Heyting algebra. Then the following conditions hold.*

$$\begin{aligned}
(x^b)^a &= x^{a \sqcap b}, & (1) \\
\bigsqcap\{t^{t^a} \mid t \in A\} &= a, & (2) \\
x^a \sqcap x^b &= x^{a \sqcup b}, & (3) \\
\bigsqcap\{a^t \mid t \in S\} &= a^{\bigsqcup S}, & (4) \\
x &\leq x^y, & (5) \\
x^y \sqcap y^x = 1 &\Rightarrow x = y, & (6) \\
\bigsqcap S = 1 &\Rightarrow \forall a \in S, a = 1. & (7)
\end{aligned}$$

5.2 Preparation of interpretation

Let p, which is called the *reference point*, be some point of the topological space $(X, \mathcal{O}(X))$ such that the following condition

$$\bigcap \mathcal{U}(p) \textit{ is an open set}$$

hold where $\mathcal{U}(p)$ is an open neighborhood[11] of p. We will parametrize our model with $\mathcal{O}(X)$ and p. Let us call this condition the *point condition*. The point condition is necessary to prove soundness.

Definition 5.5 (Dependent Function)**.** Let A be a set, and $B(a)$ be a set with parameter $a \in A$. We define dependent function domain as follows

$$\prod_{a\in A} B(a) := \{f \subset \coprod_{a\in A} B(a) \mid \forall a \in A, \exists! b \in B(a), (a,b) \in f\}$$

that is functions whose graph belongs to

$$\coprod_{a\in A} B(a) := \{(x,y) \in A \times \bigcup_{a\in A} B(a) \mid y \in B(x)\}.$$

The function PT called *Product Type* is defined as follow.

Definition 5.6 (Product Type)**.**

$$PT_{\Gamma,x}(A,B) := \begin{cases} \mathrm{PP} & (A \text{ is a propositional term for } \Gamma \\ & \quad \text{and } B \text{ is a propositional term for } \Gamma) \\ \mathrm{TP} & (A \text{ is not a propositional term for } \Gamma \\ & \quad \text{and } B \text{ is a propositional term for } (\Gamma; x : A)) \\ \mathrm{T} & (\text{otherwise}) \end{cases}$$

The function $PT_{\Gamma,x}$ maps two types into string symbols $\{\mathrm{PP}, \mathrm{TP}, \mathrm{T}\}$. Its goal is to discriminate cases of $\forall x : A.B$ to give them different interpretations.

Next, we introduce the Grothendieck universes as in [28].

Definition 5.7. Let α be an ordinal. We define V_α as follows.

- $V_0 = \phi$
- $V_\alpha = \bigcup_{\beta<\alpha} \mathcal{P}(V_\beta)$

And we define the Grothendieck universe $\mathscr{U}(i)$ as follows

$$\mathscr{U}(i) = V_{\lambda_i}$$

where λ_i is i-th inaccessible cardinal.

Lemma 5.8. *$A \in \mathscr{U}(i)$ and $B(a) \in \mathscr{U}(i)$ for each $a \in A$ imply $\prod_{a\in A} B(a) \in \mathscr{U}(i)$.*

[11] An open neighborhood of p is a set of open sets containing the point p

5.3 Interpretation of the judgments

In this model, a type T is interpreted into a set $[\![T]\!]$, and a context $x_1 : T_1; x_2 : T_2; \cdots ; x_n : T_n$ is interpreted into a tuple in $[\![T_1]\!] \times [\![T_2]\!] \times \cdots \times [\![T_n]\!]$ (when there are no dependent types in the context).

First, we define the interpretation of contexts $[\![-]\!]$, judgments $[\![- \vdash -]\!]$ and strict judgments $[\![- \vdash -]\!]'$ by mutual recursion as follows.

Definition 5.9 (interpretation)**.** Let $(X, \mathcal{O}(X))$ be a topological space, and p be a reference point of X satisfying the *point condition.*

(i) Definition of the strict-interpretation of a judgment $[\![- \vdash -]\!]'$

$$[\![\Gamma \vdash A]\!]'(\gamma) = \begin{cases} [\![\Gamma \vdash A]\!](\gamma) \cap \{p\} & (A \text{ is a propositional term in } \Gamma) \\ [\![\Gamma \vdash A]\!](\gamma) & (\text{otherwise}) \end{cases}$$

(ii) Definition of the interpretation of a context $[\![-]\!]$

$$\begin{array}{rcl} [\![[]]\!] & := & \{()\} \\ [\![\Gamma; (x : A)]\!] & := & \{(\gamma, \alpha) \mid \gamma \in [\![\Gamma]\!] \text{ and } \alpha \in [\![\Gamma \vdash A]\!]'(\gamma)\} \\ & = & \coprod_{\gamma \in [\![\Gamma]\!]} [\![\Gamma \vdash A]\!]'(\gamma) \end{array}$$

(iii) Definition of the interpretation of a judgment $[\![- \vdash -]\!]$
If t is a proof term, then

$$[\![\Gamma \vdash t]\!] = p$$

otherwise,

$$\begin{array}{rcl} [\![\Gamma \vdash \mathrm{Type}_i]\!](\gamma) & := & \mathscr{U}(i) \\ [\![\Gamma \vdash \mathrm{Prop}]\!](\gamma) & := & \mathcal{O}(X) \end{array}$$

$$
\begin{aligned}
[\![\Gamma \vdash \forall x : P.Q]\!](\gamma) &:= \begin{cases}
\Big([\![\Gamma \vdash Q]\!](\gamma)\Big)^{[\![\Gamma \vdash P]\!](\gamma)} \\
\qquad (\text{when } PT_{\Gamma,x}(P,Q) = \text{PP}) \\[2ex]
\bigsqcap\{[\![\Gamma; (x : P) \vdash Q]\!](\gamma, \alpha) \mid \alpha \in [\![\Gamma \vdash P]\!](\gamma)\} \\
\qquad (\text{when } PT_{\Gamma,x}(P,Q) = \text{TP}) \\[2ex]
\prod_{\alpha \in [\![\Gamma \vdash P]\!]'(\gamma)} [\![\Gamma; (x : P) \vdash Q]\!](\gamma, \alpha) \\
\qquad (\text{when } PT_{\Gamma,x}(P,Q) = \text{T})
\end{cases} \\
[\![\Gamma \vdash \lambda x : A.t]\!](\gamma) &:= \Big\{\big(\alpha, [\![\Gamma; (x : A) \vdash t]\!](\gamma, \alpha)\big) \mid \alpha \in [\![\Gamma \vdash A]\!]'(\gamma)\Big\} \\
[\![\Gamma \vdash uv]\!](\gamma) &:= [\![\Gamma \vdash u]\!](\gamma)\Big([\![\Gamma \vdash v]\!](\gamma)\Big) \\
[\![\Gamma \vdash x_i]\!](\gamma) &:= \gamma_i
\end{aligned}
$$

For simplicity, we write $[\![T]\!]$ for $[\![[] \vdash T]\!]()$, when the context is empty.

The interpretation of a context $[\![\Gamma]\!]$ is a sequence whose length is the length of Γ. $[\![\Gamma \vdash t]\!]$ is the function whose domain is Γ and which maps to some set. Most cases are similar to Werner's interpretation, so we only explain the interpretation of $\forall x : P.Q$. There are three cases, according to the result of $PT_{\Gamma,x}(P,Q)$. When $PT_{\Gamma,x}(P,Q) = \text{PP}$, the interpretation of $[\![\Gamma \vdash \forall x : P.Q]\!]$ represents the logical implication $P \Rightarrow Q$. We use the Heyting algebra representation of this implication. Here we assume that x does not appear in Q, thanks to our restriction. Otherwise we would need to build the interpretation of $[\![\Gamma; (x : P) \vdash Q]\!](\gamma, p)$, but this requires that $p \in [\![\Gamma \vdash P]\!](\gamma)$, which is not always true. When $PT_{\Gamma,x}(P,Q) = \text{TP}$ the interpretation of $[\![\Gamma \vdash \forall x : P.Q]\!]$ represents universal quantification, and again we use the infinite meet operator of the complete Heyting algebra to express it. In the last case only the representation becomes a set theoretical dependent function.

Next, we introduce the substitution lemma as follows.

Lemma 5.10 (substitution lemma). *We assume* $\Gamma \vdash u : U$ *is derivable. If*

$$(\gamma, [\![\Gamma \vdash u]\!](\gamma), \delta) \in [\![\Gamma; (x : U); \Delta]\!]$$

holds for any γ and δ, then

$$[\![\Gamma; \Delta[x\backslash u] \vdash t[x\backslash u]]\!](\gamma, \delta) = [\![\Gamma; (x : U); \Delta \vdash t]\!](\gamma, [\![\Gamma \vdash u]\!](\gamma), \delta)$$

for all t and Δ.

This lemma appears already in [28] and [20]. Here we give a new proof for our model. To prove it, we introduce the following two lemmas.

Lemma 5.11. *Weakening the context does not change the interpretation, i.e.*

$$[\![\Gamma \vdash u]\!](\gamma) = [\![\Gamma; \Delta \vdash u]\!](\gamma, \delta)$$

for any γ and δ such that $(\gamma, \delta) \in [\![\Gamma; \Delta]\!]$

Lemma 5.12. *If $[\![\Gamma; (x : U); \Delta \vdash t]\!]$ is well-defined, then so is $[\![\Gamma; \Delta[x\backslash u] \vdash t[x\backslash u]]\!]$. And more, $(\gamma, [\![\Gamma \vdash u]\!](\gamma), \delta) \in [\![\Gamma; (x : U); \Delta]\!]$ implies $(\gamma, \delta) \in [\![\Gamma; \Delta[x\backslash u]]\!]$.*

Next, we are ready to prove the substitution lemma 5.10.

Proof of Lemma 5.10. If t is a proof term, it is clear by Lemma 2.9. We will prove it in the case where t is not a proof term. It is provable by induction on term t. We write α for $[\![\Gamma \vdash u]\!](\gamma)$.

1. When $t = \text{Prop}, \text{Type}_i$.
   Clear.

2. When t is variable.

   - When $t = x$.
     By Definition of interpretation of judgment, the equation

     $$[\![\Gamma; (x : U); \Delta \vdash x]\!](\gamma, [\![\Gamma \vdash u]\!](\gamma), \delta) = [\![\Gamma \vdash u]\!](\gamma)$$

     holds. Then by Lemma 5.11,

     $$[\![\Gamma \vdash u]\!](\gamma) = [\![\Gamma; \Delta[x\backslash u] \vdash u]\!](\gamma, \delta)$$

     also holds. Hence, the statement holds in this case.

   - When $t \neq x$.
     Clear.

3. When $t = \lambda y : A.t'$.
   By hypothesis of induction, the following two conditions

$$\begin{aligned}
[\![\Gamma; (x:U); \Delta \vdash A]\!](\gamma,\alpha,\delta) &= [\![\Gamma; \Delta[x\backslash u] \vdash A[x\backslash u]]\!](\gamma,\delta) \\
[\![\Gamma; (x:U); \Delta; (y:A) \vdash t']\!](\gamma,\alpha,\delta,\upsilon) &= \\
[\![\Gamma; (x:U); \Delta[x\backslash u]; (y:A[x\backslash u]) &\vdash t'[x\backslash u]]\!](\gamma,\alpha,\delta,\upsilon)
\end{aligned}$$

   hold. Hence we have the following equation.

$$\begin{aligned}
& [\![\Gamma; (x:U); \Delta \vdash \lambda y : A.t']\!](\gamma,\alpha,\delta) \\
= \; & \Big\{ \big(\upsilon, [\![\Gamma; (x:U); \Delta; (y:A) \vdash t']\!](\gamma,\alpha,\delta,\upsilon)\big) \mid \upsilon \in [\![\Gamma; (x:U); \Delta \vdash A]\!](\gamma,\alpha,\delta) \Big\} \\
= \; & \Big\{ \big(\upsilon, [\![\Gamma; \Delta[x\backslash u]; (y:A[x\backslash u]) \vdash t']\!](\gamma,\alpha,\delta,\upsilon)\big) \mid \upsilon \in [\![\Gamma; \Delta[x\backslash u] \vdash A]\!](\gamma,\alpha,\delta) \Big\} \\
= \; & [\![\Gamma; \Delta[x\backslash u] \vdash \lambda y : (A[x\backslash u]).(t[x\backslash u])]\!](\gamma,\delta)
\end{aligned}$$

4. When $t = t_1\ t_2$.
   By hypothesis of induction, the following two conditions

$$\begin{aligned}
[\![\Gamma; (x:U); \Delta \vdash t_1]\!](\gamma,\alpha,\delta) &= [\![\Gamma; \Delta[x\backslash u] \vdash t_1[x\backslash u]]\!](\gamma,\delta) \\
[\![\Gamma; (x:U); \Delta \vdash t_2]\!](\gamma,\alpha,\delta) &= [\![\Gamma; \Delta[x\backslash u] \vdash t_2[x\backslash u]]\!](\gamma,\delta)
\end{aligned}$$

   hold. Therefore, we have following equation.

$$\begin{aligned}
& [\![\Gamma; (x:U); \Delta \vdash t_1\ t_2]\!](\gamma,\alpha,\delta) \\
= \; & [\![\Gamma; (x:U); \Delta \vdash t_1]\!](\gamma,\alpha,\delta)\big([\![\Gamma(x:U); \Delta \vdash t_2]\!](\gamma,\alpha,\delta)\big) \\
= \; & [\![\Gamma; \Delta[x\backslash u] \vdash t_1[x\backslash u]]\!](\gamma,\delta)\big([\![\Gamma; \Delta \vdash t_2[x\backslash u]]\!](\gamma,\delta)\big) \\
= \; & [\![\Gamma; \Delta[x\backslash u] \vdash (t_1[x\backslash u])(t_2[x\backslash u])]\!](\gamma,\delta)
\end{aligned}$$

5. When $t = \forall y : A.B$.

   - When $PT_{\Gamma,x}(A,B) = \mathrm{PP}$.
     By hypothesis of induction, the following two conditions hold.

$$\begin{aligned}
& [\![\Gamma; (x:U); \Delta \vdash A]\!](\gamma,\alpha,\delta) \\
& \qquad = [\![\Gamma; \Delta[x\backslash u] \vdash A[x\backslash u]]\!](\gamma,\delta) \\
& [\![\Gamma; (x:U); \Delta; (y:A) \vdash B]\!](\gamma,\alpha,\delta,p) \\
& \qquad = [\![\Gamma; \Delta[x\backslash u]; (y:A[x\backslash u]) \vdash B[x\backslash u]]\!](\gamma,\delta,p)
\end{aligned}$$

Since the variable y does not appear freely in B, the following equation holds.

$$[\![\Gamma; (x : U); \Delta \vdash B]\!](\gamma, \alpha, \delta) = [\![\Gamma; \Delta[x\backslash u] \vdash B[x\backslash u]]\!](\gamma, \delta)$$

Therefore, we have following equation.

$$\begin{aligned}
& [\![\Gamma; (x : U); \Delta \vdash \forall y : A.B]\!](\gamma, \alpha, \delta) \\
= \ & \big([\![\Gamma; (x : U); \Delta \vdash B]\!](\gamma, \alpha, \delta)\big)^{[\![\Gamma; (x:U); \Delta \vdash A]\!](\gamma,\alpha,\delta)} \\
= \ & \big([\![\Gamma; \Delta[x\backslash u] \vdash B[x\backslash u]]\!](\gamma, \delta)\big)^{[\![\Gamma; \Delta[x\backslash u] \vdash A[x\backslash u]]\!](\gamma,\delta)} \\
= \ & \big([\![\Gamma; \Delta[x\backslash u] \vdash \forall y : (A[x\backslash u]).(B[x\backslash u])]\!](\gamma, \delta)
\end{aligned}$$

- When $PT_{\Gamma,x}(A, B) = \mathrm{TP}$
  By hypothesis of induction, the following two conditions hold.

$$\begin{aligned}
& [\![\Gamma; (x : U); \Delta \vdash A]\!](\gamma, \alpha, \delta) \\
& \qquad = [\![\Gamma; \Delta[x\backslash u] \vdash A[x\backslash u]]\!](\gamma, \delta) \\
& [\![\Gamma; (x : U); \Delta; (y : A) \vdash B]\!](\gamma, \alpha, \delta, v) \\
& \qquad = [\![\Gamma; \Delta[x\backslash u]; (y : A[x\backslash u]) \vdash B]\!](\gamma, \delta, v)
\end{aligned}$$

  Therefore, we have the following equation for TP.

$$\begin{aligned}
& [\![\Gamma; (x : U); \Delta \vdash \forall y : A.B]\!](\gamma, \alpha, \delta) \\
= \ & \textstyle\bigsqcap_{v \in [\![\Gamma; (x:U); \Delta \vdash A]\!](\gamma,\alpha,\delta)} [\![\Gamma; (x : U); \Delta; (y : A) \vdash B]\!](\gamma, \alpha, \delta, v) \\
= \ & \textstyle\bigsqcap_{v \in [\![\Gamma; \Delta[x\backslash u] \vdash A[x\backslash u]]\!](\gamma,\delta)} [\![\Gamma; \Delta[x\backslash u]; (y : A[x\backslash u]) \vdash B[x\backslash u]]\!](\gamma, \delta, v) \\
= \ & [\![\Gamma; \Delta[x\backslash u] \vdash \forall y : (A[x\backslash u]).(B[x\backslash u])]\!](\gamma, \delta)
\end{aligned}$$

- When $PT_{\Gamma,x}(A, B) = \mathrm{T}$.
  Respectively for T, we have the same equation replacing $\bigsqcap$ by $\prod$.

□

We introduce the theorem for the interpretation of logical symbols in definition 2.11. In this theorem, the validity of the interpretation can be seen.

Theorem 5.13 (interpretation of logical symbols)**.**

(i) $[\![\Gamma \vdash \bot]\!] = \phi$

(ii) $[\![\Gamma \vdash A \wedge B]\!](\gamma) = ([\![\Gamma \vdash A]\!](\gamma)) \sqcap ([\![\Gamma \vdash B]\!](\gamma))$

(iii) $[\![\Gamma \vdash A \vee B]\!](\gamma) = ([\![\Gamma \vdash A]\!](\gamma)) \sqcup ([\![\Gamma \vdash B]\!](\gamma))$

(iv) $[\![\Gamma \vdash \exists x : A.Q]\!](\gamma) = \bigsqcup_{\alpha \in [\![\Gamma \vdash A]\!](\gamma)} [\![\Gamma; (x : A) \vdash Q]\!](\gamma, \alpha)$

(v) $[\![\Gamma \vdash A \leftrightarrow B]\!](\gamma) = X \Rightarrow [\![\Gamma \vdash A]\!](\gamma) = [\![\Gamma \vdash B]\!](\gamma)$

(vi) $[\![\Gamma \vdash x =_A y]\!](\gamma) = X \Rightarrow [\![\Gamma \vdash x]\!](\gamma) = [\![\Gamma \vdash y]\!](\gamma)$

Proof. Let $a, b, q(\alpha)$ be

$$\begin{aligned} a &:= [\![\Gamma \vdash A]\!](\gamma) \\ b &:= [\![\Gamma \vdash B]\!](\gamma) \\ q(\alpha) &:= [\![\Gamma; (x : A) \vdash Q]\!](\gamma, \alpha). \end{aligned}$$

By using Lemma 5.4 and Lemma 5.11 we have the followings:

(i) The proof of $[\![\Gamma \vdash \bot]\!] = \phi$.

$$\begin{aligned} [\![\Gamma \vdash \bot]\!](\gamma) &= [\![\Gamma \vdash \forall P : \text{Prop}.P]\!](\gamma) \\ &= \bigsqcap \{[\![\Gamma; (P : \text{Prop}) \vdash P]\!](\gamma, x) \mid x \in [\![\Gamma \vdash \text{Prop}]\!](\gamma)\} \\ &= \bigsqcap \{x | x \in \mathcal{O}(X)\} \\ &= \phi \end{aligned}$$

(ii) The proof of $[\![\Gamma \vdash A \wedge B]\!](\gamma) = ([\![\Gamma \vdash A]\!](\gamma)) \sqcap ([\![\Gamma \vdash B]\!](\gamma))$.

$$\begin{aligned} [\![\Gamma \vdash A \wedge B]\!](\gamma) &= [\![\Gamma \vdash \forall P : \text{Prop}.(A \to (B \to P)) \to P]\!](\gamma) \\ &= \bigsqcap \{x^{(x^b)^a} \mid x \in \mathcal{O}(X)\} \\ &= \bigsqcap \{x^{x^{a \sqcap b}} \mid x \in \mathcal{O}(X)\} \quad \text{(by Lemma 5.4 (1))} \\ &= a \sqcap b \quad \text{(by Lemma 5.4 (2))} \\ &= [\![\Gamma \vdash A]\!](\gamma) \sqcap [\![\Gamma \vdash B]\!](\gamma) \end{aligned}$$

(iii) The proof of $[\![\Gamma \vdash A \vee B]\!](\gamma) = ([\![\Gamma \vdash A]\!](\gamma)) \sqcup ([\![\Gamma \vdash B]\!](\gamma))$.

$$\begin{aligned}
[\![\Gamma \vdash A \vee B]\!](\gamma) &= [\![\Gamma \vdash \forall P : \text{Prop}.(A \to P) \to ((B \to P) \to P)]\!](\gamma) \\
&= \bigsqcap\{(x^{x^b})^{x^a} \mid x \in \mathcal{O}(X)\} \\
&= \bigsqcap\{x^{x^a \sqcap x^b} \mid x \in \mathcal{O}(X)\} \quad \text{(by Lemma 5.4 (1))} \\
&= \bigsqcap\{x^{x^{a \sqcup b}} \mid x \in \mathcal{O}(X)\} \quad \text{(by Lemma 5.4 (3))} \\
&= a \sqcup b \quad \text{(by Lemma 5.4 (2))} \\
&= [\![\Gamma \vdash A]\!](\gamma) \sqcup [\![\Gamma \vdash B]\!](\gamma)
\end{aligned}$$

(iv) The proof of $[\![\Gamma \vdash \exists x : A.Q]\!](\gamma) = \bigsqcup_{\alpha \in [\![\Gamma \vdash A]\!](\gamma)} [\![\Gamma; (x : A) \vdash Q]\!](\gamma, \alpha)$.

$$\begin{aligned}
[\![\Gamma \vdash \exists a : A.Q]\!](\gamma) &= [\![\Gamma \vdash \forall P : \text{Prop}.(\forall a : A.(Q \to P) \to P]\!](\gamma) \\
&= \bigsqcap\{x^{\bigsqcap\{x^{q(\alpha)} \mid \alpha \in a\}} \mid x \in \mathcal{O}(X)\} \\
&= \bigsqcap\{x^{x^{\bigsqcup\{q(\alpha) \mid \alpha \in a\}}} \mid x \in \mathcal{O}(X)\} \quad \text{(by Lemma 5.4 (4))} \\
&= \bigsqcup\{q(\alpha) \mid \alpha \in a\} \quad \text{(by Lemma 5.4 (2))} \\
&= \bigsqcup_{\alpha \in [\![\Gamma \vdash A]\!](\gamma)} [\![\Gamma; (a : A) \vdash Q]\!](\gamma, \alpha)
\end{aligned}$$

(v) The proof of $[\![\Gamma \vdash A \leftrightarrow B]\!](\gamma) = X \Rightarrow [\![\Gamma \vdash A]\!](\gamma) = [\![\Gamma \vdash B]\!](\gamma)$.

$$\begin{aligned}
[\![\Gamma \vdash A \leftrightarrow B]\!](\gamma) &= [\![\Gamma \vdash A \to B]\!](\gamma) \sqcap [\![\Gamma \vdash B \to A]\!](\gamma) \\
&= a^b \sqcap b^a
\end{aligned}$$

Hence we have $a = b$ by Lemma 5.4 (6) since $a^b \sqcap b^a = X$.

(vi) The proof of $[\![\Gamma \vdash x =_A y]\!](\gamma) = X \Rightarrow [\![\Gamma \vdash x]\!](\gamma) = [\![\Gamma \vdash y]\!](\gamma)$.

$$\begin{aligned}
[\![\Gamma \vdash x =_A y]\!](\gamma) &= [\![\Gamma \vdash \forall Q : (A \to \text{Prop}).Q\ x \leftrightarrow Q\ y]\!](\gamma) \\
&= \bigsqcap_{f : a \to \mathcal{O}(X)} [\![\Gamma; (Q : A \to \text{Prop}) \vdash Q\ x \leftrightarrow Q\ y]\!](\gamma, f)
\end{aligned}$$

Since $[\![\Gamma \vdash x =_A y]\!](\gamma) = X$, we have the following fact:

$$\forall f : a \to \mathcal{O}(X), [\![\Gamma; (Q \to \text{Prop}) \vdash Q\ x \leftrightarrow Q\ y]\!](\gamma, f) = X$$

Therefore we have $f([\![\Gamma \vdash x]\!](\gamma)) = f([\![\Gamma \vdash y]\!](\gamma))$ for any function $f : a \to \mathcal{O}(X)$. Hence, the statement holds.

□

5.4 Proof of the Soundness

We are ready to prove soundness of this type system.

Theorem 5.14 (soundness)**.** *We assume $[\![\Gamma]\!]$ is non empty set.*

1. *If $t_1 =_\beta t_2$, and $\Gamma \vdash t_1 : T, \Gamma \vdash t_2 : T$ is derivable, then $[\![\Gamma \vdash t_1]\!](\gamma) = [\![\Gamma \vdash t_2]\!](\gamma)$.*

2. *If $\Gamma \vdash t : T$ is derivable and $[\![\Gamma]\!]$ is non-empty set, then $[\![\Gamma \vdash t]\!](\gamma) \in [\![\Gamma \vdash T]\!](\gamma)$.*

Proof of Theorem 5.14.
1. It is sufficient that $[\![\Gamma \vdash (\lambda x : U.t)\ u]\!](\gamma) = [\![\Gamma \vdash t[x\backslash u]]\!](\gamma)$. By using Lemma 5.10,

$$\begin{aligned} & [\![\Gamma \vdash (\lambda x : U.t)u]\!] \\ = \; & [\![\Gamma \vdash \lambda x : U.t]\!](\gamma)\big([\![\Gamma \vdash u]\!](\gamma)\big) \\ = \; & [\![\Gamma; (x : U) \vdash t]\!](\gamma, [\![\Gamma \vdash u]\!](\gamma)) \\ = \; & [\![\Gamma \vdash t[x\backslash u]]\!](\gamma) \end{aligned}$$

Hence, the statement holds.

2. This is proved by induction on Typing Rules in Table 2. We assume that p is a reference point.

1. Case of Axiom
   $[\![\Gamma \vdash \text{Prop}]\!](\gamma) \in [\![\Gamma \vdash \text{Type}_i]\!](\gamma)$ is clear. Similarly, $[\![\Gamma \vdash \text{Type}_i]\!](\gamma) \in [\![\Gamma \vdash \text{Type}_{i+1}]\!](\gamma)$ is also clear.

2. Case of Subtyping
   The fact that $[\![\Gamma \vdash A]\!](\gamma) \in [\![\Gamma \vdash \text{Type}_i]\!](\gamma)$ implies $[\![\Gamma \vdash A]\!](\gamma) \in [\![\Gamma \vdash \text{Type}_{i+1}]\!](\gamma)$ is clear.

3. Case of PI-Type
   We will show the fact that

$$\begin{aligned} & \big(\forall \gamma, \alpha, [\![\Gamma \vdash A]\!](\gamma) \in [\![\Gamma \vdash s_1]\!](\gamma) \\ & \qquad \textit{and } [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \in [\![\Gamma; (x : A) \vdash s_2]\!](\gamma, \alpha))\big) \\ & \Rightarrow \; \forall \gamma, [\![\Gamma \vdash \forall x : A.B]\!](\gamma) \in [\![\Gamma \vdash s_3]\!](\gamma). \end{aligned}$$

   There are three cases as follows.

- $PT_{\Gamma,x}(A,B) = T$
  By definition of the interpretation of judgment, the following equation

  $$[\![\Gamma \vdash \forall x : A.B]\!](\gamma) = \prod_{\alpha \in [\![\Gamma \vdash A]\!]'(\gamma)} [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha)$$

  holds. There are the following two cases:

  - A is not a propositional term for Γ
    Since $[\![\Gamma \vdash A]\!](\gamma) \in \mathscr{U}(i)$, $[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \in \mathscr{U}(j)$ for any γ, α and Lemma 5.8, we have

    $$\prod_{\alpha \in [\![\Gamma \vdash A]\!](\gamma)} [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \in \mathscr{U}(\max(i, j)).$$

  - A is a propositional term for Γ
    Since $[\![\Gamma \vdash A]\!](\gamma) \in \mathcal{O}(\mathcal{X}) \subset \mathcal{U}(j)$, $[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \in \mathscr{U}(j)$ for any γ, α and Lemma 5.8, we have

    $$\prod_{\alpha \in [\![\Gamma \vdash A]\!](\gamma)} [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \in \mathscr{U}(j).$$

  Hence, the statement holds.

- $PT_{\Gamma,x}(A,B) = \mathrm{TP}$
  It is clear since $[\![\Gamma \vdash \forall x : A.B]\!](\gamma)$ is an open set by definition of the interpretation of judgment.

- $PT_{\Gamma,x}(A,B) = \mathrm{PP}$
  It is clear since $[\![\Gamma \vdash \forall x : A.B]\!](\gamma)$ is an open set by definition of the interpretation of judgment.

4. Case of Abstraction
   We will show the fact that

   $$\begin{aligned} &\big(\forall \gamma, \alpha, [\![\Gamma; (x : A) \vdash t]\!](\gamma, \alpha) \in [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \\ &\quad \textit{and } [\![\Gamma \vdash \forall x : A.B]\!](\gamma) \in [\![\Gamma \vdash s]\!](\gamma)\big) \\ &\Rightarrow \ \forall \gamma, [\![\Gamma \vdash \lambda x : A.t]\!](\gamma) \in [\![\Gamma \vdash \forall x : A.B]\!](\gamma) \end{aligned}$$

   There are three cases as follows.

- $PT_{\Gamma,x}(A,B) = T$
  By definition of the interpretation, we have the following equations:

$$\begin{array}{rcl}
[\![\Gamma \vdash \lambda x : A.t]\!](\gamma) & = & \Big\{\big(\alpha, [\![\Gamma; (x:A) \vdash t]\!](\gamma,\alpha)\big) \mid \alpha \in [\![\Gamma \vdash A]\!]'(\gamma)\Big\} \\
[\![\Gamma \vdash \forall x : A.B]\!](\gamma) & = & \prod_{\alpha \in [\![\Gamma \vdash A]\!]'(\gamma)} [\![\Gamma; (x:A) \vdash B]\!](\gamma,\alpha) \\
 & = & \{f \subset \coprod_{a \in A} B(a) \mid \forall a \in [\![\Gamma \vdash A]\!](\gamma), \exists! b, (a,b) \in f\}
\end{array}$$

  Then, we must prove the following equation:

$$\Big\{\big(\alpha, [\![\Gamma; (x:A) \vdash t]\!](\gamma,\alpha)\big) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\Big\} \\ \in \{f \subset \coprod_{a \in A} B(a) \mid \forall a \in [\![\Gamma \vdash A]\!](\gamma)$$

  But it is clear[12] by induction of hypothesis.

- $PT_{\Gamma,x}(A,B) = \mathrm{TP}$
  Since $\lambda x : A.t$ is a proof term, we have the following equations

$$[\![\Gamma \vdash \lambda x : A.t]\!](\gamma) = p$$

  Hence, the fact we must prove is

$$p \in [\![\Gamma \vdash \forall x : A.B]\!](\gamma)$$

  By definition we have the following equation.

$$[\![\Gamma \vdash \forall x : A.B]\!](\gamma) = \bigsqcap\{[\![\Gamma; (x:A) \vdash B]\!](\gamma,\alpha) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}.$$

  If $[\![\Gamma \vdash A]\!](\gamma)$ is the empty set, then the statement holds since $[\![\Gamma \vdash \forall x : A.B]\!](\gamma) = X$[13] . We assume that $[\![\Gamma \vdash A]\!](\gamma)$ is a non-empty set. We have

$$\forall \alpha \in [\![\Gamma \vdash A]\!](\gamma), p \in [\![\Gamma; (x:A) \vdash B]\!](\gamma,\alpha).$$

[12] Especially, If $[\![\Gamma \vdash A]\!](\gamma)$ is the empty set, then $[\![\Gamma \vdash \forall x : A.B]\!](\gamma) = \{\phi\}$ and $[\![\Gamma \vdash \lambda x : A.t]\!](\gamma) = \phi$.

[13] $\bigsqcap \phi = X$

since $[\![\Gamma; (x : A) \vdash t]\!](\gamma, \alpha) = p$. Therefore, we have the following equation:

$$p \in \bigcap \{[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}$$

However $\bigsqcap S \neq \bigcap S$ hold in general, since $\bigsqcap S$ is the interior of $\bigcap S$ when S is non empty subset of X. Now, we apply the point condition here[14]. We have

$$\begin{aligned}
[\![\Gamma \vdash \forall x : A.B]\!](\gamma) &= \bigsqcap \{[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\} \\
&= \bigcap \{[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}
\end{aligned}$$

since $\bigcap \{[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}$ is an open set by the point condition. Hence, the condition holds in this case.

- $PT_{\Gamma,x}(A, B) = \mathrm{PP}$
  Since $\lambda x : A.B$ is a proof term, we have the following equation

  $$[\![\Gamma \vdash \lambda x : A.t]\!](\gamma) = p$$

  Hence, the fact we must prove is

  $$p \in [\![\Gamma \vdash \forall x : A.B]\!](\gamma)$$

  By definition of the interpretation of judgment, we have

  $$[\![\Gamma \vdash \forall x : A.B]\!](\gamma) = \Big([\![\Gamma \vdash B]\!](\gamma)\Big)^{[\![\Gamma \vdash A]\!](\gamma)}.$$

  By characteristic of Heyting algebra,

  $$[\![\Gamma \vdash B]\!](\gamma) \subset [\![\Gamma \vdash \forall x : A.B]\!](\gamma).$$

  By induction hypothesis $p \in [\![\Gamma \vdash B]\!](\gamma)$, so that the condition holds in this case.

5. Case of Apply
   We will show the fact that

   $$\begin{aligned}
   &\big(\forall \gamma, [\![\Gamma \vdash u]\!](\gamma) \in [\![\Gamma \vdash \forall x : A.B]\!](\gamma) \textit{ and } [\![\Gamma \vdash v]\!](\gamma) \in [\![\Gamma \vdash A]\!](\gamma)\big) \\
   &\Rightarrow \ \forall \gamma, [\![\Gamma \vdash u\ v]\!](\gamma) \in [\![\Gamma \vdash B[x \backslash v]]\!](\gamma)
   \end{aligned}$$

   There are three cases as follows.

[14]This is the place we need it in the proof.

- $PT_{\Gamma,x}(A, B) = \mathrm{T}$
  By definition of the interpretation of judgment, the following equation

  $$\begin{array}{rcl} [\![\Gamma \vdash u\ v]\!](\gamma) & = & [\![\Gamma \vdash u]\!](\gamma)\big([\![\Gamma \vdash v]\!](\gamma)\big) \\ [\![\Gamma \vdash u]\!](\gamma) & \in & \displaystyle\prod_{\alpha \in [\![\Gamma \vdash A]\!]'(\gamma)} [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \end{array}$$

  holds. Therefore, we have

  $$[\![\Gamma \vdash u\ v]\!](\gamma) \in [\![\Gamma; (x : A) \vdash B]\!](\gamma, [\![\Gamma \vdash v]\!](\gamma))$$

  By Lemma5.10, we have

  $$[\![\Gamma; (x : A) \vdash B]\!](\gamma, [\![\Gamma \vdash v]\!](\gamma)) = [\![\Gamma \vdash B[x\backslash v]]\!](\gamma).$$

  Hence, the statement holds in this case.
- $PT_{\Gamma,x}(A, B) = \mathrm{TP}$
  We will show that $p \in [\![\Gamma \vdash B[x\backslash v]]\!](\gamma)$ since $[\![\Gamma \vdash u]\!](\gamma) = [\![\Gamma \vdash u\ v]\!](\gamma) = p$. We have the following equation

  $$p \in \bigcap\{[\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha) \mid \alpha \in [\![\Gamma \vdash A]\!](\gamma)\}.$$

  This equation implies the fact that

  $$\forall \alpha \in [\![\Gamma \vdash A]\!](\gamma), p \in [\![\Gamma; (x : A) \vdash B]\!](\gamma, \alpha).$$

  By Lemma 5.10 and the fact $[\![\Gamma \vdash v]\!](\gamma) \in [\![\Gamma \vdash A]\!](\gamma)$, we have

  $$p \in [\![\Gamma \vdash B[x\backslash v]]\!](\gamma).$$

  Hence, the statement holds in this case.
- $PT_{\Gamma,x}(A, B) = \mathrm{PP}$
  We will show that $p \in [\![\Gamma \vdash B]\!](\gamma)$ since $[\![\Gamma \vdash u]\!](\gamma) = [\![\Gamma \vdash v]\!](\gamma) = [\![\Gamma \vdash u\ v]\!](\gamma) = p$ and the variable x does not appear freely in B. The following equation holds.

  $$[\![\Gamma \vdash \forall x : A.B]\!](\gamma) = \Big([\![\Gamma \vdash B]\!](\gamma)\Big)^{[\![\Gamma \vdash A]\!](\gamma)}$$

By definition of Heyting algebra, we have

$$[\![\Gamma \vdash \forall x : A.B]\!](\gamma) \cap [\![\Gamma \vdash A]\!](\gamma) \subset [\![\Gamma \vdash B]\!](\gamma).$$

Then we have

$$p \in [\![\Gamma \vdash B]\!](\gamma).$$

by induction hypothesis. Hence, the statement holds in this case.

6. Case of Variable
   We must show that

$$\big((x : A) \in \Gamma \quad \textit{and} \quad \forall\gamma, [\![\Gamma \vdash A]\!](\gamma) \in [\![\Gamma \vdash s]\!](\gamma)\big)$$
$$\Rightarrow \forall\gamma, [\![\Gamma \vdash x]\!](\gamma) \in [\![\Gamma \vdash A]\!](\gamma)$$

   It is clear by definition of $[\![\Gamma]\!]$.

7. Case of Beta Equality
   We must show that

$$\big(\forall\gamma, [\![\Gamma \vdash x]\!](\gamma) \in [\![\Gamma \vdash A]\!](\gamma) \; \textit{and} \; A =_\beta B\big)$$
$$\Rightarrow \forall\gamma, [\![\Gamma \vdash x]\!](\gamma) \in [\![\Gamma \vdash B]\!](\gamma)$$

   It is clear by Theorem 5.14 (1).

□

Corollary 5.15. *If P is a provable propositional term for Γ, then*

$$\forall\gamma \in [\![\Gamma]\!], p \in [\![\Gamma \vdash P]\!](\gamma)$$

holds.

6 Application

Let's compare Werner's classical model with our intuitionistic model on some simple cases.

x^y	0	1	2
0	2	0	0
1	2	2	1
2	2	2	2

Table 3: value of x^y

6.1 Classical model

Let

$$\begin{aligned} X &:= 1 = \{\phi\}, \\ \mathcal{O}(X) &:= \{0,1\} = \{\phi, \{\phi\}\}, \\ p &:= 0 = \phi. \end{aligned}$$

This coincides with Werner's Model [28]. But this model is classical, since

$$\begin{aligned} 0 &\in [\![\forall P : \text{Prop}.P \vee \neg P]\!] \\ &= \prod_{o \in \mathcal{O}(X)} o \vee \neg o = 1 \end{aligned}$$

holds.

6.2 Models disproving excluded middle

Let

$$\begin{aligned} X &:= 2 = \{0,1\}, \\ \mathcal{O}(X) &:= \{0,1,2\} = \{\phi, \{\phi\}, \{\phi, \{\phi\}\}\}, \\ p &:= 1 = \{\phi\}. \end{aligned}$$

In this model, we have the following fact

$$1 \notin [\![\forall P : \text{Prop}.P \vee \neg P]\!] = 1$$

by using the following equations

$$\neg 0 = 2,$$

x^y	ϕ	α	β	γ	X
ϕ	X	ϕ	ϕ	ϕ	ϕ
α	X	X	α	α	α
β	X	β	X	β	β
γ	X	X	X	X	γ
X	X	X	X	X	X

Table 4: value of x^y

$$\begin{aligned} \neg 1 &= 0, \\ \neg 2 &= 0. \end{aligned}$$

Hence this model avoids the principle of excluded middle.

However in this model, $P \to Q \vee Q \to P$ holds, since we have the following fact by Table 3.

$$\begin{aligned} & [\![\forall P : \text{Prop}.\forall Q : \text{Prop}.(P \to Q) \vee (Q \to P)]\!] \\ = & \prod_{o_1, o_2 \in \mathcal{O}(X)} o_1^{o_2} \vee o_2^{o_1} \\ = & \ 2. \end{aligned}$$

By adding more elements we can refine the model further. Let

$$\begin{aligned} X &:= \{a, b, x\} \\ \mathcal{O}(X) &:= \{\phi, \alpha, \beta, \gamma, X\}, \\ &= \{\phi, \{a\}, \{b\}, \{a, b\}, \{a, b, x\}\}, \\ p &:= x. \end{aligned}$$

In this model, $P \to Q \vee Q \to P$ does not hold, since we have the following fact by Table 4.

$$x \notin [\![\forall P : \text{Prop}.\forall Q : \text{Prop}.(P \to Q) \vee (Q \to P)]\!] = \gamma$$

7 Reynolds' Paradox

There is a problem when expanding the set theoretical model, which is called Reynolds' paradox [23]. Basically the Reynolds' paradox says that if the interpretation of an impredicative sort has more than one element, it causes a cardinality paradox in the set theoretical model. This seems to be in contradiction with our model, so in this section we will analyze its assumptions.

7.1 Outline of the Paradox

Let $\mathbb{T}$ be an impredicative sort, i.e. if $\Gamma \vdash A : s$ and $\Gamma ; (x : A) \vdash B : \mathbb{T}$ are derivable for any sort s then $\Gamma \vdash \forall x : A.B : \mathbb{T}$ is derivable. We assume that there exists a type B whose sort is $\mathbb{T}$ such that $[\![B]\!]$ has at least two elements, i.e.

$$\vdash B : \mathbb{T} \quad \textit{and} \quad \sharp[\![B]\!] \geq 2.$$

In [23] Reynolds says that the existence of such a term B causes a paradox in set-theoretical models. First, we define the category $\mathbf{Sets}_{\mathbb{I}}$ and the endofunctor T of $\mathbf{Sets}_{\mathbb{I}}$.

Definition 7.1.

- Let $\mathbf{Sets}_{\mathbb{I}}$ be a category with:
  - $\mathrm{Obj}(\mathbf{Sets}_{\mathbb{I}}) := \{[\![P]\!] \mid \vdash P : \mathbb{I} \text{ is derivable } \}$
  - $\mathrm{Hom}([\![P_1]\!], [\![P_2]\!]) := [\![P_1]\!] \to [\![P_2]\!]$
- Let T be a endofunctor of $\mathbf{Sets}_{\mathbb{I}}$ with
  - $T([\![P]\!]) := ([\![P]\!] \to [\![B]\!]) \to [\![B]\!]$
  - $T(\rho) := h \in T([\![P_1]\!]) \mapsto \{(g, h(g \circ \rho)) | g \in [\![P_2]\!] \to [\![B]\!]\}$
    where $\rho \in [\![P_1]\!] \to [\![P_2]\!]$

The paper [23] claims the following lemma:

Lemma 7.2.

- $\exists u \in \mathrm{Obj}(\mathbf{Sets}_{\mathbb{I}}), \exists H \in \mathrm{Hom}(Tu, u)$ *s.t.*
  $\forall s \in \mathrm{Obj}(\mathbf{Sets}_{\mathbb{I}}), \forall f \in \mathrm{Hom}(Ts, s), \exists ! \rho \in \mathrm{Hom}(u, s)$ *s.t.*
  following diagram commutes.

$$\begin{tikzcd} Tu \arrow[r, "T\rho"] \arrow[d, "H"'] & Ts \arrow[d, "f"] \\ u \arrow[r, "\rho"] & s \end{tikzcd}$$

- *Tu and u are equivalent, i.e. $Tu \cong u$.*

By definition of endofunctor T, $\sharp[\![B]\!] \geq 2$ implies Tu and u have different cardinalities in spite of Tu and u being isomorphism. Therefore, the existence of a type B of impredicative sort such that $\sharp[\![B]\!] \geq 2$ causes a paradox.

7.2 Avoiding the Paradox

In ECC, we have an impredicative sort Prop, and there is a type B of Prop such that $\sharp[\![B]\!] \geq 2$. However, this doesn't cause a paradox. In fact, to prove the existence of a function $H \in Tu \to u$, Reynolds constructs a term t of the type $((P \to B) \to B) \to P)$ in the proof of lemma 2 in [23], where P is a type such that $[\![P]\!] = u$. However in our model $[\![(P \to B) \to B]\!]$ is not equal to the set theoretical function $T([\![P]\!]) = ([\![P]\!] \to [\![B]\!]) \to [\![B]\!]$ but is just some open set of $(X, \mathcal{O}(X))$

$$[\![(P \to B) \to B]\!] = [\![B]\!]^{[\![B]\!]^{[\![P]\!]}}$$

since both P and B are propositional terms. Thus this discussion moves to the Heyting algebra part of the model where we need not fear such paradox.

8 Future Work

There are still three remaining questions we would like to answer in the future: whether the *point condition* is really needed to prove soundness, whether we can handle full ECC, without our restrictions on the type system, and how close to completeness is our model.

The point condition is very restrictive. It seems to allow only discrete models. Hence we would like to remove it to allow a wider variety of models. In fact we have not found any counterexample when removing the *point condition*, up to now.

We would also like to lift the restrictions on the PI-Type rule, which prohibits statements about proofs, and on the subtyping rule. They come from the fact that, in the interpretation of contexts, we use the strict interpretation, which restricts all propositional terms to either ϕ or the singleton $\{p\}$, so that we cannot build an element when the non-strict interpretation, while being non-empty, does not contain p. We are considering several approaches to overcome this problem.

While this model rejects the excluded middle, it still admits proof-irrelevance

$$\forall t_1, t_2, (t_1, t_2 \; is \; proof \; term \; for \; \Gamma) \Rightarrow [\![\Gamma \vdash t_1]\!](\gamma) = [\![\Gamma \vdash t_2]\!](\gamma).$$

Since the existence of t such that following condition

$$\Gamma; (p_1 : P); (p_2 : P) \vdash t : p_1 =_P p_2 \quad (\textit{where } \Gamma \vdash P : \mathrm{Prop} \textit{ is derivable})$$

holds is not provable in general, this means that our model is still not complete. We are now investigating how close to completeness it is, with and without the restriction.

References

[1] Peter Aczel. The type theoretic interpretation of constructive set theory. *Studies in Logic and the Foundations of Mathematics (Logic Colloquium '77)*, 96:55–66, 1978.

[2] Peter Aczel. The type theoretic interpretation of constructive set theory: choice principles. *Studies in Logic and the Foundations of Mathematics (Proceedings of the L. E. J. Brouwer Centenary Symposium)*, 110:1–40, 1982.

[3] Peter Aczel. The type theoretic interpretation of constructive set theory: inductive definitions. *Logic Methodology and Philosophy of Science*, 7:17–49, 1986.

[4] Peter Aczel. Aspects of general topology in constructive set theory. *Annals of Pure and Applied Logic*, 137, 2006.

[5] Peter Aczel, Hajime Ishihara, Takako Nemoto, and Yasushi Sangu. Generalized geometric theories and set-generated classes. Preprint available from http://www.newton.ac.uk/preprints/NI12032.pdf, 2012.

[6] Peter Aczel and Michael Rathjen. Notes on constructive set theory, 2001.

[7] Guillaume Alexandre. Coq user's contributions: ZF. `http://www.lix.polytechnique.fr/coq/pylons/contribs/view/ZF/trunk`.

[8] Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[9] Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda calculus with types.* Cambridge University Press, 2013.

[10] Bruno Barras. Sets in coq, coq in sets. *Journal of Formalized Reasoning*, 3(1):29–48, 2010.

[11] Errett Bishop and Douglas Bridges. *Constructive Analysis.* Springer, 1985.

[12] Thierry Coquand and Gerard Huet. The calculus of constructions. *Information and computation*, 76(2):95–120, 1988.

[13] Eduarde Giménez. Codifying guarded definitions with recursive schemes. In *Types for Proofs and Programs*, volume 996 of *Lecture Notes in Computer Science*, pages 39–59, 1995.

[14] Jean-Yves Girard. *Proofs and types.* Cambridge University Press, 1989.

[15] Michael J.Beeson. *Foundations of constructive mathematics.* Springer, 1985.

[16] Kenneth Kunen. *Set theory: an introduction to independence proofs.* College Publ, 2011.

[17] Zhaohui Luo. A higher-order calculus and theory abstraction. *Information and Computation*, 90(1):107–137, 1991.

[18] Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory.* Springer, 1992.

[19] Per Martin-Löf. An intuitionistic theory of types: Predicative part. *Studies in Logic and the Foundations of Mathematics (Proceedings of the Logic Colloquium '73)*, 80:73–118, 1975.

[20] Alexandre Miquel and Benjamin Werner. The not so simple proof-irrelevant model of CC. In *Types for Proof and Programs*, volume 2426 of *Lecture Notes in Computer Science*, pages 240–258, 2003.

[21] Paulin-Mohring. Inductive definitions in the system coq rules and properties. *Typed Lambda Calculi and Applications*, 664:328–345, 1993.

[22] Benjamin Pierce. *Types and programming languages.* MIT press, 2002.

[23] John Reynolds. Polymorphism is not set-theoretic. volume 173 of *Lecture Notes in Computer Science*, pages 145–156, 1984.

[24] Giovanni Sambin. Some points in formal topology. *Theoretical computer science*, 305:347–408, 2003.

[25] Carlos Simpson. Set-theoretical mathematics in coq. arXiv preprint math/0402336, 2004.

[26] Dirk van Dalen. Intuitionistic logic. *Handbook of Philosophical Logic*, III:225–339, 1984.

[27] Benjamin Werner. Coq user's contributions: ZFC. `http://www.lix.polytechnique.fr/coq/pylons/contribs/view/ZFC/trunk`.

[28] Benjamin Werner. Sets in types, types in sets. In *Theoretical aspects of computer software*, volume 1281 of *Lecture Notes in Computer Science*, pages 530–546, 1997.