$(\lambda\zeta{:}\Omega \in \Omega.q\ \zeta\ \zeta))$

Set-Theoretic Paradoxes in λU and λU^-

Bachelor's Thesis in Mathematics

Author:	Rutger Kuyper
Student number:	0715204
Supervisor:	Herman Geuvers
Second reader:	Wim Veldman

Radboud University Nijmegen

Set-Theoretic Paradoxes in λU and λU^-

Spring 2010

Abstract

The intuitive notion of 'algorithm' can be formalized using the lambda calculus. Adding types to this formalism gives us type systems, which are in direct correspondence with intuitionistic logics. The type systems λU and λU^- are known to be inconsistent; that is, their corresponding logics are inconsistent. We analyze two proofs of this inconsistency, as presented by A.J.C. Hurkens in 1995 and A. Miquel in 2001. Furthermore, we try to construct a fixed point combinator by modifying the proof of A. Miquel.

Bachelor's Thesis in Mathematics

Author:	Rutger Kuyper
Student number:	0715204
Supervisor:	Herman Geuvers
Second reader:	Wim Veldman

Radboud University Nijmegen

Contents

Contents v

Introduction 1

1 Preliminaries 3
- 1.1 (Untyped) Lambda Calculus . . . 3
- 1.2 Typed Lambda Calculus . . . 6
- 1.3 Set Theory . . . 9

2 λU and λU^- 13
- 2.1 λU and λU^- as Pure Type Systems . . . 13
- 2.2 Logic in λU and λU^- . . . 15

3 Hurkens's Paradox 19
- 3.1 The Burali-Forti Paradox in Naive Set Theory . . . 19
- 3.2 Paradoxical Universes . . . 24
- 3.3 A Paradoxical Universe in λU^- . . . 26
- 3.4 Powerful Universes . . . 28
- 3.5 A Powerful Universe in λU^- . . . 30
- 3.6 Reduction Behavior . . . 31

4 Encoding Naive Set Theory in λU and λU^- 33
- 4.1 Non-Well-Founded Sets . . . 33
- 4.2 Interpretation in λU . . . 40
- 4.3 Interpretation in λU^- . . . 47
- 4.4 Reduction Behavior . . . 47

A Coq Formalization 51

Bibliography 61

Introduction

Since the introduction of the lambda calculus by A. Church in 1936, paradoxes have been an important topic in both typed and untyped lambda calculus. The logical inconsistency of the original lambda calculus follows from the *Kleene-Rosser paradox*, which is quite similar to Russell's paradox in naive set theory. To resolve this paradox, the first typed lambda calculus was introduced: $\lambda\!\!\rightarrow$, or the *simply typed lambda calculus*. While computationally weaker, this gives us a consistent system.

However, over time various other typed lambda calculi have been introduced, which are not always consistent. In this thesis we study the systems λU and λU^-. Logically, these systems correspond to extensions of higher order logic in which one is allowed to form impredicative domains.

These systems were originally conceived by J.-Y. Girard in 1972. At the same time, he showed the inconsistency of λU by formalizing a variant of the Burali-Forti paradox in naive set theory. At the time, he did not know if λU^- was consistent or inconsistent. This question was answered by T. Coquand in 1991: λU^- is inconsistent as well.

A much simpler proof of the inconsistency of both λU and λU^- was presented by A.J.C. Hurkens in 1995, again based on the Burali-Forti paradox. Because it is much simpler, it is much easier to analyze the proof (as we will discuss in a bit).

As discussed above, Girard's paradox is mainly inspired by the Burali-Forti paradox. While this is the historically first-known paradox in naive set theory, it is not the 'simplest' paradox. A more elementary paradox is Russell's paradox; one might want to formalize this paradox in λU and λU^-. This was done by Miquel in 2001, using so-called *non-well-founded sets* which allows one to encode sets as graphs. This gives another proof of the inconsistency of λU and λU^-.

Closely related to the inconsistency of λU and λU^- is the question if a *fixed point combinator* exists in these systems. In 1987, D.

Howe constructed a *looping combinator* (which is, roughly speaking, a combinator which is 'almost' a fixed point combinator) from Girard's paradox, using a construction proposed by A.R. Meyer and M.B. Reinhold in 1986. One can also apply this construction to the (much simpler to analyze) inconsistency proof of Hurkens. Unfortunately, G. Barthe and T. Coquand showed in 2006 that this does not give us a fixed point combinator. Up to this day, it is still an open question if a fixed point combinator exists in λU or λU^-.

This thesis will discuss the inconsistency proofs of Hurkens and Miquel. Furthermore, we will try to construct a fixed point combinator from Miquel's proof. The next chapter will briefly discuss the preliminaries for this thesis: both typed and untyped lambda calculus and some elementary set theory. In Chapter two we will exhibit the type systems λU and λU^- as so-called *Pure Type Systems*. Chapter three will discuss Hurkens's paradox in detail; in particular we will try to understand the thoughts and intuition hiding behind the paradox. Finally, Chapter four will discuss Miquel's formalization of Russell's paradox, including a discussion of the (im)possibilities to turn this paradox into a fixed point combinator.

1 Preliminaries

In this chapter, we briefly discuss the most important preliminaries necessary for understanding this thesis: untyped lambda calculus, typed lambda calculus and set theory.

1.1 (Untyped) Lambda Calculus

The lambda calculus is a formalism used to describe the intuitive notion of 'algorithm', just like the (more well-known) Turing machine. In fact, the expressive power of the lambda calculus is *equivalent* to that of Turing machines. These two formalisms can thus be used intermittently.

Then, what makes the lambda calculus different? This is easily seen if one looks at the way the formalism works: while a Turing machine can be seen as a big calculator, imperatively doing computations, lambda calculus uses some sort of abstraction of functions. This difference is best seen after we give the definition of the terms of the lambda calculus.

Definition 1.1.1. We presuppose an infinite set $\mathcal{V}ar$ of variables, denoted $x, y, \ldots$

We define the **terms** of the lambda calculus inductively:

- Every $x \in \mathcal{V}ar$ is a term.
- If M is a term and $x \in \mathcal{V}ar$, then $(\lambda x.M)$ is a term.
- If M, N are terms, then $(M\,N)$ is a term.

We call a term of the second kind an **abstraction**, and a term of the third kind an **application**.

We will use $M, N, \ldots$ to denote arbitrary lambda-terms.

The idea of a function should be clear: an abstraction $(\lambda x.M)$ can be seen as the function $x \mapsto M$. For example, the abstraction $(\lambda x.x)$ corresponds to the identity, while the abstraction $(\lambda x.(\lambda y.x))$ corresponds to the first projection of a cartesian product.

To avoid clutter, we will no longer write the brackets if the term does not become ambiguous by leaving them out. For example, we will write $\lambda x.x$ and $\lambda x.\lambda y.x$ for the terms above. Furthermore, we will assume application to be **left-associative**, i.e. we will write $N\, M_1 \cdots M_k$ for $(\cdots((N\, M_1)\, M_2)\cdots M_k)$.

Before we continue, we need the notion of *free and bound variables*.

Definition 1.1.2. Let M be a term. The set of **free variables** of M, denoted $\mathcal{FV}(M)$, is defined inductively by:

- $\mathcal{FV}(x) := \{x\}$,
- $\mathcal{FV}(\lambda x.M) := \mathcal{FV}(M) - \{x\}$,
- $\mathcal{FV}(M\, N) := \mathcal{FV}(M) \cup \mathcal{FV}(N)$.

We call a variable x **bound** in M if it occurs in M but is not free. If a term M contains no free variables, we call M a **combinator**.

For example, the variable x is bound in $\lambda x.x$, and so are the variables x and y in $\lambda x.\lambda y.x$. The variable x is free in x, while the variable z is neither free nor bound in any of these terms.

Now take a look at the term $\lambda y.y$. Intuitively, this should be equal to $\lambda x.x$, since they both represent the identity. We formalize this intuition in the next definition.

Definition 1.1.3. Let M, N be terms. If we can obtain M from N by renaming bound variables, then $M = N$.

Next, we define substitution on lambda-terms.

Definition 1.1.4. Let M, N be terms and let $x \in \mathcal{Var}$. Then we define the **substitution** of N in M for x, written as $M[x := N]$, inductively by:

- $x[x := N] = N$,
- $y[x := N] = y$ if $y \neq x$,
- $(\lambda y.M_1)[x := N] := (\lambda y.M_1[x := N])$ if $x \neq y$ and $y \notin \mathcal{FV}(N)$,

- $(M_1\, M_2)[x := N] := (M_1[x := N]\, M_2[x := N])$.

The third clause requires some explanation. For example, look at the term $\lambda x.x$. Without the requirement that $x \neq y$, we could do the substitution $(\lambda x.x)[x := y]$, which would give $\lambda x.y$. Thus, our identity function turned into something like a constant function, which is behavior we certainly did not want. Conversely, without the requirement that $y \notin \mathcal{FV}(N)$, we could make the substitution $(\lambda y.x)[x := y]$, which would give $\lambda y.y$, turning a constant function into the identity function.

However, the free variable requirement can be fulfilled by renaming the bound variables in M to fresh variables before doing the substitution.

Finally, we turn to the last step in our construction of the lambda calculus. So far we have constructed terms and defined substitution on them, but we do not yet have any means of actually doing any calculations with them. The *β-reduction* rule changes this.

Definition 1.1.5. $\rightarrow_\beta$, called **β-reduction**, is the smallest relation satisfying

$$(\lambda x.M)\, N \rightarrow_\beta M[x := N]$$

which is also closed under:

- If $M \rightarrow_\beta N$, then $\lambda x.M \rightarrow_\beta \lambda x.N$,
- If $M_1 \rightarrow_\beta N$, then $M_1\, M_2 \rightarrow_\beta N\, M_2$,
- If $M_2 \rightarrow_\beta N$, then $M_1\, M_2 \rightarrow_\beta M_1\, N$.

We denote $=_\beta$ for the reflexive, symmetric and transitive closure of $\rightarrow_\beta$.

Using $\rightarrow_\beta$, we can finally do some computations. We first turn back to our familiar term $\lambda x.x$. We now see, that for an arbitrary term M, we have that $(\lambda x.x)\, M \rightarrow_\beta M$, illustrating that this term is indeed the identity.

An important result in the untyped lambda calculus is the existence of a *fixed point combinator* (since it forms an essential part of the proof that the lambda-calculus is Turing complete). Such a combinator is exactly what the name says: a combinator that assigns to each lambda term a fixed point of that term.

Definition 1.1.6. A **fixed point combinator** is a combinator M such that, for all terms f:

$$f\,(M\,f) =_\beta M\ f$$

The most famous fixed point combinator is the fixed point combinator Y, discovered by H.B. Curry, which we exhibit below.

Definition 1.1.7. $Y := \lambda f.(\lambda x.f\,(x\,x))\,(\lambda x.f\,(x\,x))$

Proposition 1.1.8. *Y is a fixed point combinator.*

Proof. For every term g we have:

$$\begin{aligned} Y\,g &= (\lambda f.(\lambda x.f\,(x\,x))\,(\lambda x.f\,(x\,x)))\,g \\ &\to_\beta (\lambda x.g\,(x\,x))\,(\lambda x.g\,(x\,x)) \\ &\to_\beta g((\lambda x.g\,(x\,x))\,(\lambda x.g\,(x\,x))) \\ &= g\,(Y\,g) \end{aligned}$$

□

Here ends our discussion of the untyped lambda calculus. A more elaborate (but still elementary) introduction can be found in [Barendregt and Barendsen, 1998]; if one truly wishes to dive into the subject one could take a look at Barendregt's major work [Barendregt, 1984].

1.2 Typed Lambda Calculus

Intuitively, the lambda calculus of the previous section is missing something: while usual functions carry the notions of domain and codomain, terms in the untyped lambda calculus do not. For example, one can apply the term $\lambda x.x$ to whatever term ones wants (even to itself!). Typed lambda calculi introduce the notion of *types* to solve this issue.

There are various ways to introduce these types, leading to various **type systems**. In this section we will study the idea behind these type systems by looking at a simple example: $\lambda\!\to$ (pronounce: lambda-arrow, or *simple type theory*).

First, we define the types in $\lambda\!\to$.

Definition 1.2.1. We presuppose an infinite set $\mathcal{TV}ar$ of **type variables**, denoted as $a, b, \ldots$

The **types** of $\lambda\!\to$ are defined inductively as follows:

- Every $a \in \mathcal{TV}ar$ is a type.

- If A, B are types, then $(A \to B)$ is a type.

We will use $A, B, \ldots$ to denote arbitrary types.

Intuitively, the types $A \to B$ will correspond with functions from A to B. As we will see below, this is exactly the case.

To avoid clutter, we will once again not write brackets if they are not necessary to avoid ambiguity. Furthermore, we will assume $\to$ to be **right-associative**, i.e. we will write $A_1 \to A_2 \to \cdots \to A_n$ for $(A_1 \to (A_2 \to \cdots \to (A_{n-1} \to A_n) \cdots))$.

Next, we will define the *pseudoterms* of $\lambda\!\to$. These pseudoterms will still contain 'invalid' terms, corresponding to invalid function application. Therefore, after defining the pseudoterms we will present derivation rules to select the terms from the pseudoterms.

Definition 1.2.2. We presuppose an infinite set $\mathcal{V}ar$ of variables, denoted $x, y, \ldots$.

The **pseudoterms** of $\lambda\!\to$ are defined inductively by:

- Every $x \in \mathcal{V}ar$ is a pseudoterm.
- If M is a pseudoterm, $x \in \mathcal{V}ar$ and A is a type, then $(\lambda x{:}A.M)$ is a pseudoterm.
- If M, N are pseudoterms, then $(M\,N)$ is a pseudoterm.

Once again, we will leave out brackets where possible and assume application to be left-associative.

The pseudoterms of $\lambda\!\to$ are almost the same as the terms of the lambda-calculus, with one slight (but important) modification: the addition of a type to the abstractions. These types will be used to denote the 'domain' of the abstraction: for example, the term $\lambda x{:}A.x$ will be the identity function of type $A{\to}A$.

In order to proceed, we first need to define *contexts*:

Definition 1.2.3. A **context** is a finite list of declarations of the form $x : A$, where $x \in \mathcal{V}ar$ and A is a type, such that every variable occurs on the left-hand-side of a declaration most once.

In $\lambda\!\to$ we will ignore the order of the declarations, and thus look at a context as just a set instead of a list.

We will use $\Gamma, \Delta, \ldots$ to denote arbitrary contexts.

Next, we will present the rules to select the valid terms. We do this by assigning types to certain pseudoterms, and elevating those pseudoterms with a valid type to the class of terms.

Definition 1.2.4. The **typing rules** of $\lambda\!\rightarrow$ are defined as follows:

$$\textbf{(var)} \quad \frac{}{\Gamma, x : A \vdash x : A} \qquad \text{if } x \notin \Gamma$$

$$\textbf{(}\boldsymbol{\lambda}\textbf{)} \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : A \to B}$$

$$\textbf{(app)} \quad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B}$$

The **terms** of $\lambda\!\rightarrow$ are the pseudoterms M such that there exists a context Γ and a type A for which we can derive $\Gamma \vdash M : A$.

For example, the following derivation shows that $\lambda x{:}A.x$ is a valid term and is of type $A \to A$:

$$\dfrac{\dfrac{}{x : A \vdash x : A}}{\vdash \lambda x{:}A.x : A \to A}$$

One might think that we are now done with our discussion of $\lambda\!\rightarrow$, since we presented the entire system. However, there is one very important and surprising result which we want to discuss next: the *Curry-Howard-De Bruijn isomorphism*.

This isomorphism gives a correspondence between type theories and certain intuitionistic logics. In our case, $\lambda\!\rightarrow$ corresponds to *minimal first-order propositional logic*, that is, propositional logic with $\rightarrow$ as only connective.

We will not go into too much detail, but will instead sketch the relation. First, we take a look at the relation between propositions and types. We can relate these two quite naturally:

- A propositional variable can be seen as a type variable,
- The connective $\rightarrow$ can be seen as the $\rightarrow$ used to construct types.

Next, we take a look at the terms. The isomorphism relates terms of a certain type to *derivations* or *proofs* of the corresponding proposition. More concretely:

- An assumption A corresponds to a variable of type A,
- The introduction rule for $\rightarrow$ corresponds to abstraction,
- The elimination rule for $\rightarrow$ corresponds to application.

For example, the term $\lambda x{:}A.x$ corresponds to the following proof of the proposition $A \to A$:

$$\frac{\not{A}^{(x)}}{A \to A}\ (x)$$

This correspondence allows for two major applications of typed lambda calculi. First, we can use them for *proof checking*: by converting a mathematical proof to a lambda term, we can check the validity of the proof by checking the type of the term. But we can do even more: given the term corresponding to the proof, we can look at the computational behavior of this proof, allowing us to extract programs from proof. This practice is called *program extraction*.

We now end our discussion of $\lambda\!\to$ and typed lambda calculi. For a deeper introduction we refer the reader to [van Raamsdonk, 2008]; the truly fascinated reader is again encouraged to take a look at [Barendregt, 1984].

1.3 Set Theory

We presuppose some basic knowledge of axiomatic set theory. In this section, we briefly recall the axioms of the Zermelo-Fraenkel axiom system and we exhibit some elementary results which we will need later in this thesis.

Extensionality axiom

$$\forall x \forall y[\forall z[z \in x \leftrightarrows z \in y] \to x = y]$$

Axiom scheme of specification
Let $\varphi = \varphi(x_1, \ldots, x_n, y)$ be a formula. Then:

$$\forall x_1 \forall x_2 \cdots \forall x_n \forall z \exists u \forall y[y \in u \leftrightarrows (y \in z \wedge \varphi(x_1, \ldots, x_n, y))]$$

Pair axiom

$$\forall x \forall y \exists z \forall t[t \in z \leftrightarrows (t = x \vee t = y)]$$

Union axiom

$$\forall x \exists y \forall t[t \in y \leftrightarrows \exists u[u \in x \wedge t \in u]]$$

Axiom scheme of replacement
Let $\varphi := \varphi(x_1, \ldots, x_n, y, z)$ be a formula. Then:

$$\forall x_1 \forall x_2 \cdots \forall x_n \forall u[\forall y \in u \exists! z[\varphi(x_1, \ldots, x_n, y, z)] \\ \rightarrow \exists w \forall y \in u \exists z \in w[\varphi(x_1, \ldots, x_n, y, z)]]$$

Infinity axiom

$$\exists x[\exists y[y \in x \wedge \forall z[\neg(z \in y)]] \wedge \\ \forall y[y \in x \rightarrow \exists z[z \in x \wedge \forall t[t \in z \leftrightarrows (t \in y \vee t = y)]]]]$$

Power set axiom

$$\forall x \exists y \forall z[z \in y \leftrightarrows \forall t[t \in z \rightarrow t \in x]]$$

The axioms presented up to now form the axiom system ZF^-. We obtain ZF by additionally assuming the *axiom scheme of foundation*:
Axiom scheme of foundation
Let $\varphi = \varphi(x_1, \ldots, x_n, y)$ be a formula. Then:

$$\forall x_1 \forall x_2 \cdots \forall x_n[\forall y[\forall z \in y[\varphi(x_1, \cdots, x_n, z)] \rightarrow \varphi(x_1, \ldots, x_n, y)] \\ \rightarrow \forall y[\varphi(x_1, \ldots, x_n, y)]]$$

We remark that this is not the form the axiom is usually given in. The reason for this is that the usual formulation implies the law of the excluded middle, which, given our intuitionistic point of view in this thesis, is not a valid principle. The formulation we gave above (which is often called the *principle of $\in$-induction*) is classically equivalent to the usual axiom, but is intuitionistically safe.

We remark the following direct corollary of the axiom scheme of foundation:

Corollary 1.3.1. *Axiom scheme of foundation* $\models \forall y[\neg(y \in y)]$.

Proof. Directly using $\in$-induction. □

There is one particular result which we will use in this thesis, which we present without proof. A proof can be found in e.g. [Veldman, p. 99].

Lemma 1.3.2. *Let x be a set. Then there exists a unique set $TC(x)$ (called the **transitive closure** of x) such that:*

- $x \subseteq TC(x)$,
- $TC(x)$ *is* ***transitive****:* $\forall u \forall v[(u \in v \wedge v \in TC(x)) \rightarrow u \in TC(x)]$,
- $TC(x)$ *is the smallest transitive set containing* x*; i.e. if* y *is a transitive set such that* $x \subseteq y$*, then* $TC(x) \subseteq y$.

2

λU and λU^-

The main type systems studied in this thesis are λU and λU^-. Therefore, we will use this chapter to exhibit those systems and present some of their properties.

2.1 λU and λU^- as Pure Type Systems

We will introduce λU and λU^- as specific instances of a class of type systems called *Pure Type Systems*. These Pure Type Systems can be seen as a generalization of Barendregt's Lambda Cube.

Definition 2.1.1. A *Pure Type System* is given by a triple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ with:

- $\mathcal{S}$ a set, called the set of **sorts**;
- $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ the set of **axioms**;
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ the set of **rules**.

For $(s_1, s_2, s_3) \in \mathcal{R}$ with $s_2 = s_3$ we will write $(s_1, s_2) \in \mathcal{R}$.

Let $\mathcal{V}ar$ be a set of variables. The **pseudoterms** of the type system are:

$$T ::= \mathcal{S} \mid \mathcal{V}ar \mid (\Pi \mathcal{V}ar{:}T.T) \mid (\lambda \mathcal{V}ar{:}T.T) \mid T\,T$$

The **typing rules** of the type system are (where $s \in \mathcal{S}$):

(sort)	$\dfrac{}{\vdash s_1 : s_2}$	if $(s_1, s_2) \in \mathcal{A}$
(var)	$\dfrac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$	if $x \notin \Gamma$
(weak)	$\dfrac{\Gamma \vdash A : s \quad \Gamma \vdash M : C}{\Gamma, x : a \vdash M : C}$	if $x \notin \Gamma$
(Π)	$\dfrac{\Gamma \vdash A : s_1 \quad \Gamma, x : a \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_3}$	if $(s_1, s_2, s_3) \in \mathcal{R}$
(λ)	$\dfrac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x{:}A.B : s}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}$	
(app)	$\dfrac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B[x := N]}$	
(conv$_\beta$)	$\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B}$	if $A =_\beta B$

If x has no free occurences in B, we also write $A \to B$ for $\Pi x{:}A.B$. In spirit of the Curry-Howard-De Bruijn isomorphism, if $\Gamma, x : A \vdash \varphi : \mathcal{P}rop$ we will also write $\forall x{:}A.\varphi$ for $\Pi x{:}A.\varphi$.

There are three Pure Type Systems that are relevant to our purpose: λHOL, λU^- and λU. They all have the same sorts and axioms, but their rules differ slightly, as can be seen on the next page.

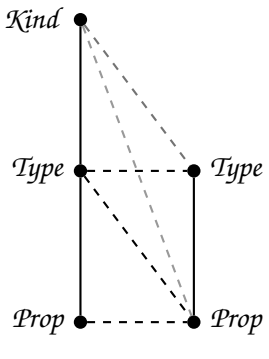
Under the Curry-Howard-De Bruijn isomorphism, λHOL corresponds to Higher Order Logic. λU^- is obtained from λHOL by adding the rule $(\mathcal{K}ind, \mathcal{T}ype)$. In the corresponding logic, this extra rule allows one to form impredicative domains.

If we also add the rule $(\mathcal{K}ind, \mathcal{P}rop)$ to λU^-, we obtain λU. The effect of adding this rule is easily seen if one considers *polymorphism*. In λU and λU^- one has a polymorphic identity:

$$id : \Pi X{:}\mathcal{T}ype.X \to X := \lambda X{:}\mathcal{T}ype.\lambda x{:}X.x$$

One might want to express $\forall X{:}\mathcal{T}ype.\forall x{:}X.(id\ X\ x) =_\beta x$. It is easily seen that we need exactly the $(\mathcal{K}ind, \mathcal{P}rop)$-rule in order to do this. Thus, λU is obtained from λU^- by allowing one to express universal properties of polymorphic functions.

$$
\begin{array}{lcl}
\mathcal{S} & := & \{\mathcal{P}rop, \mathcal{T}ype, \mathcal{K}ind\} \\
\mathcal{A} & := & \{(\mathcal{P}rop, \mathcal{T}ype), (\mathcal{T}ype, \mathcal{K}ind)\} \\
\\
\lambda\text{HOL} & & \\
\mathcal{R}_{\lambda\text{HOL}} & := & \{(\mathcal{P}rop, \mathcal{P}rop), (\mathcal{T}ype, \mathcal{T}ype), \\
 & & \quad (\mathcal{T}ype, \mathcal{P}rop)\} \\
\\
\lambda U^- & & \\
\mathcal{R}_{\lambda U^-} & := & \mathcal{R}_{\lambda\text{HOL}} \cup \{(\mathcal{K}ind, \mathcal{T}ype)\} \\
\\
\lambda U & & \\
\mathcal{R}_{\lambda U} & := & \mathcal{R}_{\lambda U^-} \cup \{(\mathcal{K}ind, \mathcal{P}rop)\}
\end{array}
$$



There are a lot of facts which one can prove about Pure Type Systems in general. More details can be found in e.g. [Geuvers and Nederhof, 1991].

2.2 Logic in λU and λU^-

Using the $(\mathcal{T}ype, \mathcal{P}rop)$ rule, which is available in all three systems above, it is possible to perform second-order quantification over propositions in the corresponding logic. Using this, we can express all standard logical constants, connectives and quantifiers by modeling their elimination rules.

Definition 2.2.1. Let $\varphi, \psi : \mathcal{P}rop$ and let A be a variable not free in φ, ψ. Then we define:

$$
\begin{array}{lcl}
\top & := & \forall A{:}\mathcal{P}rop.A \to A \\
\bot & := & \forall A{:}\mathcal{P}rop.A \\
\neg\varphi & := & \varphi \to \bot \\
\varphi \wedge \psi & := & \forall A{:}\mathcal{P}rop.(\varphi \to \psi \to A) \to A \\
\varphi \vee \psi & := & \forall A{:}\mathcal{P}rop.(\varphi \to A) \to (\psi \to A) \to A \\
\exists x : A.\varphi & := & \forall A{:}\mathcal{P}rop.(\forall x{:}A.\varphi \to A) \to A
\end{array}
$$

It is easy to check that these terms are typable in all three systems mentioned above, and that all these terms adhere to their usual introduction and elimination rules.

We call a PTS *inconsistent* iff the corresponding logic is inconsistent. Since we now have a term for $\bot$, we can use this term to formalize what it means for a PTS to be inconsistent.

Definition 2.2.2. A Pure Type System is called **inconsistent** iff there exists a term M such that $\vdash M : \bot$.

λHOL is consistent (i.e. it is not inconsistent). In fact, even if one adds the $(\mathcal{K}\mathit{ind}, \mathcal{P}\mathit{rop})$-rule to it, it remains consistent. However, both λU and λU^- are inconsistent; it is their inconsistency which will be the main topic of this thesis.

Closely related to this inconsistency is the question if a *fixed point combinator of sort* $\mathcal{P}\mathit{rop}$ exists in λU or λU^-.

Definition 2.2.3. A **fixed point combinator** of sort s is a term $M : \Pi A{:}s.(A \to A) \to A$ such that for all $A : s$ and $f : A \to A$ we have:

$$f\,(M\,A\,f) =_\beta M\,A\,f$$

It is not yet known if a fixed point combinator of sort $\mathcal{P}\mathit{rop}$ exists in λU or λU^-. Observe that such a combinator in particular yields an inconsistency proof, by applying it to $\bot$ and $\lambda x{:}\bot.x$. Therefore, a logical approach to finding a fixed point combinator of sort $\mathcal{P}\mathit{rop}$ would be to make some modifications to an inconsistency proof. We will discuss the results of this approach for the inconsistency proofs presented in this thesis.

Further in this thesis we will need some notion of equality. To this end, we will use so-called *Leibniz equality*.

Definition 2.2.4. Let A be a type. Then we define $=_A$, called **Leibniz equality** on A, by:

$$a =_A b \quad := \quad \forall P{:}A{\to}\mathcal{P}\mathit{rop}.(P\,a) \to (P\,b)$$

As the name might suggest, for each type A we have that $=_A$ is an equivalence relation on A.

Proposition 2.2.5. *For every type A, $=_A$ is an equivalence relation on A.*

Proof. We have:

- $\lambda P{:}A{\to}\mathcal{P}rop.\lambda 0{:}(P\,a).0 : a =_A a$.
- Let $0 : a =_A b$ and let $1 : a =_A a$. Then $0\,(\lambda u{:}A.u =_A a)\,1 : b =_A a$.
- Let $0 : a =_A b$ and let $1 : b =_A c$. Then $1\,(\lambda u{:}A.a =_A u)\,0 : a =_A c$.

Therefore, $=_A$ is indeed an equivalence relation on A. □

We can also formalize the notion of the *power set* of a type. We first observe that the subsets of a set V are essentially the same as the predicates on that set: given a predicate P on V we can form $\{v \in V \mid P(v)\}$. Conversely, given a subset $U \subseteq V$ we can form the predicate $P(v) := v \in U$. It is easily checked that these operations are the inverse of each other (modulo logical equivalence). Thus, we can formalize the power set of a type A as the type of predicates on A.

Definition 2.2.6. Let A be a type. Then we define:

$$\wp A := A{\to}\mathcal{P}rop$$

Finally, we define the image function $f^{\to}$ and the inverse image function $f^{\leftarrow}$ of a function f, and the dual relation R^{δ} of a relation R.

Definition 2.2.7. Let $f : A{\to}B$. Then we define:

$$f^{\to} : \wp A{\to}\wp B := \lambda X{:}\wp A.\lambda b{:}B.\exists a{:}A.X\,a \wedge f\,a =_B b$$
$$f^{\leftarrow} : \wp B{\to}\wp A := \lambda Y{:}\wp B.\lambda a{:}A.Y\,(f\,a)$$

Definition 2.2.8. Let $R : A{\to}B{\to}C$. Then we define:

$$R^{\delta} : B{\to}A{\to}C := \lambda b{:}B.\lambda a{:}A.R\,a\,b$$

3

Hurkens's Paradox

In this chapter we will study Hurkens's paradox in λU^-, as laid out in Hurkens [1995]. In particular, we will try to understand the set-theoretic inspiration behind it. Since his paradox is greatly inspired by the Burali-Forti paradox in naive set theory[1], we will first study this paradox.

3.1 The Burali-Forti Paradox in Naive Set Theory

In 1897, Cesare Burali-Forti published in [Burali-Forti, 1897a] what would later become known as the *Burali-Forti paradox*. This is the historically first known paradox in naive set theory. In comparison, Russell didn't discover his (more elemental) paradox until 1901.

In this coming section we will derive the Burali-Forti paradox. To this end, we will study so-called *ordinal numbers*, which are closely related to *well-founded ordered sets*. Therefore, we will take a look at these well-founded ordered sets first.

Definition 3.1.1. Let V be a set. An **ordering** on V is a $< \subseteq V \times V$ such that:

$<$ is irreflexive:	$\forall u[\neg(u < u)]$
$<$ is transitive:	$\forall u, v, w[((u < v) \wedge (v < w)) \Rightarrow (u < w)]$

If $<$ is an ordering, we call $(V, <)$ an **ordered set**. We define $\mathcal{O}$ to be the set of all ordered sets.

[1] By naive set theory, we refer to the set theories as studies by Cantor and Frege during the end of the nineteenth century. In these set theories, one can form the set of all objects satisfying a given property.

Definition 3.1.2. Let V be a set. Then we call $< \subseteq V \times V$ **well-founded** iff the *principle of transfinite induction* holds for $<$, i.e.:

For every predicate $P(x)$ we have that $\forall v \in V[\forall w \in W[w < v \to P(w)] \to P(v)]$ implies $\forall v \in V[P(v)]$.

If $(V, <)$ is an ordered set such that $<$ is well-founded, we will call $(V, <)$ a **well-founded ordered set**.

We first present the following helpful proposition.

Proposition 3.1.3. *Let V be a set and let $< \subseteq V \times V$ be well-founded. Then $<$ is irreflexive.*

Proof. We prove, using transfinite induction: $\forall v \in V[\neg(v < v)]$:

Let $v \in V$ and assume that $\forall w \in W[w < v \to \neg(w < w)]$. To derive a contradiction, assume that $v < v$. Then, by induction hypothesis we have $\neg(v < v)$, contradicting our assumption that $v < v$. Thus, $\neg(v < v)$. □

Intuitively, one can restrict an order on V to a $W \subseteq V$ by 'forgetting' what the order does on the elements not in W. We can formalize this as follows.

Definition 3.1.4. Let V be a set, let $< \subseteq V \times V$ and let $W \subseteq V$. Then $<_{\restriction W} := < \cap (W \times W)$.

It is easy to see that, if $(V, <)$ is an ordered set, then $(W, <_{\restriction W})$ is also an ordered set. The next lemma shows that the same holds for well-foundedness.

Lemma 3.1.5. *Let V be a set and let $< \subseteq V \times V$ be well-founded. Let $W \subseteq V$. Then $<_{\restriction W}$ is well-founded.*

Proof. Let $P(x)$ be a predicate such that:

$$\forall w \in W[\forall x \in W[x <_{\restriction W} w \to P(x)] \to P(w)] \tag{3.1}$$

Define $Q(x) := v \in W \wedge Q(v)$. Then 3.1 is equivalent to:

$$\forall v \in V[\forall w \in V[w < v \to Q(w)] \to Q(v)]$$

Since $<$ is well-founded, this implies that $\forall v \in V[Q(v)]$, which is again equivalent to $\forall w \in W[P(w)]$. Thus, $<_{\restriction W}$ is indeed well-founded. □

We will now turn our focus back towards the ordinal numbers we wish to create. To this end, we take a look at structure-preserving maps on ordered sets.

Definition 3.1.6. Let $(V, <_V)$ and $(W, <_W)$ be ordered sets. An **order-isomorphism** is a bijection $f : V \to W$ such that, for all $u, v \in V$:

$$u <_V v \Leftrightarrow f(u) <_W f(v)$$

Two ordered sets $(V, <_V)$ and $(W, <_W)$ are called **order-isomorphic** (notation: $(V, <_V) \cong (W, <_W)$) iff there exists an order-isomorphism $f : V \to W$.

It is easy to see that the relation of two ordered sets being 'order-isomorphic' is an equivalence relation on $\mathcal{O}$, which induces a partition of $\mathcal{O}$. Using this, we can define *order types* and our *ordinal numbers*.

Definition 3.1.7. The **order type** of an ordered set $(V, <)$ (denoted as $\tau(V, <)$) is the unique $\alpha \in \mathcal{O}\backslash\cong$ such that $(V, <) \in \alpha$.

If $(V, <)$ is a well-founded ordered set, we call the order type of $(V, <)$ the **ordinal number** of $(V, <)$. We denote Ord for the set of all ordinal numbers.

Our definition of ordinal numbers differs from the usual one. Most authors use the term *ordinal number* to denote the order type of a *well-ordered set*, which is a well-founded ordered set $(V, <)$ which is also total, i.e. $\forall x, y \in V[x < y \vee y < x \vee x = y]$. However, requiring a well-founded order to be total before being allowed to assign it an ordinal number only complicates our proof of the Burali-Forti paradox. With our definition we obtain a simpler derivation while the essence remains the same (and it is this essence that is important to us, since we will abstract from ordinal numbers in the next section anyway). Furthermore, totality is (from an intuitionistic point of view) a very strong property. Thus, to simplify matters, we drop the totality criterion.

Now that we have constructed the ordinal numbers, our next aim will be to construct a well-founded ordering $<_{\mathit{Ord}}$ on them. That way, we can assign an ordinal number to Ord, and we will show that this ordinal number shows contradictory behavior with respect to $<_{\mathit{Ord}}$. In order to construct this ordering we will first take a look at so-called *initial segments*.

Definition 3.1.8. Let $(V, <)$ be an ordered set and let $v \in V$. Let $W := \{w \in V \mid w < v\}$. Then $I_{(V,<),v}$, the **initial segment** of $(V, <)$ from v, is defined as $I_{(V,<),v} := (W, <_{\restriction W})$.

It is easy to see that, if $(V, <)$ is a (well-founded) ordered set, then its initial segments are (well-founded) ordered sets. We can also look at their ordinal numbers, and intuitively we want to call those ordinal numbers smaller than the ordinal number of the original ordered set. We use this intuition to define a well-founded ordering on the ordinal numbers.

Definition 3.1.9. Let α, β be ordinal numbers. Let $(V, <_V)$ be a well-founded ordered set of ordinal number α and let $(W, <_W)$ be a well-founded ordered set of ordinal number β. Then we define $\alpha <_{\mathcal{Ord}} \beta$ iff $(V, <_V)$ is order-isomorphic to some initial segment of $(W, <_W)$.

Proposition 3.1.10. *$(\mathcal{Ord}, <_{\mathcal{Ord}})$ is a well-founded ordered set.*

Proof. Since order-isomorphic ordered sets have order-isomorphic initial segments (as can be seen through restricting the isomorphism), $<_{\mathcal{Ord}}$ is well-defined.

It is easy to check that $<_{\mathcal{Ord}}$ is transitive. So, by proposition 3.1.3 we are done if we show that $<_{\mathcal{Ord}}$ is well-founded.

Therefore, let $P(x)$ be a predicate such that:

$$\forall \alpha \in \mathcal{Ord}[\forall \beta \in \mathcal{Ord}[\beta <_{\mathcal{Ord}} \alpha \to P(\beta)] \to P(\alpha)] \tag{3.2}$$

Now let $\gamma \in \mathcal{Ord}$; we need to show that $P(\gamma)$ holds. Choose a well-founded ordered set $(V, <)$ of ordinal number γ. Define:

$$Q(v) := P(\tau(I_{(V,<),v}))$$

Then 3.2, applied to $\tau(I_{(V,<),v})$, is equivalent to:

$$\forall w \in V[w < v \to Q(w)] \to Q(v)$$

Since $<$ is well-founded, this implies that $Q(v)$ holds for all $v \in V$. Combining this with 3.2, applied to γ, we therefore see that $P(\gamma)$ holds.

Thus, $<_{\mathcal{Ord}}$ is indeed well-founded, and therefore $(\mathcal{Ord}, <_{\mathcal{Ord}})$ is indeed a well-founded ordered set. □

Theorem 3.1.11 (Burali-Forti paradox). *$\mathcal{Ord}$ is not a set.*

Proof. By the preceding lemma, $(\mathcal{Ord}, <_{\mathcal{Ord}})$ is well-founded. Let Ω be its ordinal number.

For each ordinal number α, we see from lemma 3.1.5 that $I_{(\mathcal{Ord},<_{\mathcal{Ord}}),\alpha}$ is also well-founded. Denote Δ_α for its ordinal number. Then we directly see that $\Delta_\alpha <_{\mathcal{Ord}} \Omega$, so in particular we have $\Delta_\Omega <_{\mathcal{Ord}} \Omega$.

However, we next prove, through transfinite induction, that for each ordinal number α we have $\Delta_\alpha \not<_{Ord} \alpha$:

Let α be an ordinal number and assume that for all $\beta <_{Ord} \alpha$ we have $\Delta_\beta \not<_{Ord} \beta$. To derive a contradiction, assume $\Delta_\alpha <_{Ord} \alpha$. By induction hypothesis we conclude that $\Delta_{\Delta_\alpha} \not<_{Ord} \Delta_\alpha$. However, since $\Delta_\alpha <_{Ord} \alpha$ it is easily seen that $I_{(Ord,<_{Ord}),\Delta_\alpha} \cong I_{(I_{(Ord,<_{Ord})},\alpha),\Delta_\alpha}$ and thus $\Delta_{\Delta_\alpha} <_{Ord} \Delta_\alpha$, which leads to a contradiction. Thus, $\Delta_\alpha \not<_{Ord} \alpha$.

So, in particular we see $\Delta_\Omega \not<_{Ord} \Omega$, which contradicts our earlier observation that $\Delta_\Omega <_{Ord} \Omega$. □

Historical remarks

In his original paper [Burali-Forti, 1897a], Burali-Forti confused well-ordered sets with what he called *perfectly ordered sets*. He called a set V perfectly ordered iff:

1. V has a first element;
2. Every element of V (provided it is not the last) has an immediate successor;
3. For every element $x \in V$ we have either:
   (a) x has no immediate predecessor, or
   (b) there is an $y \in V$ such that y precedes x, y has no immediate predecessor, and only a finite number of elements of V lie between y and x.

It can be easily shown that every well-ordered set is perfectly ordered; the converse was shown to be false by K.G. Hagström in [Hagström, 1914].

However, Burali-Forti quickly realized his mistake and published a note with a correction [Burali-Forti, 1897b], stating that his result could be obtained for well-ordered sets just as easily as for his perfectly ordered sets.

Nonetheless, Burali-Forti's paradox failed to create as much as a stir as one would expect it to. Among the reasons were the mistake mentioned above, but also the fact that Burali-Forti did not present his result as a contradiction. Instead, he presented it as part of his attempt to prove that $<_{Ord}$ is not total. However, Cantor already proved in [Cantor, 1897] that this ordering is in fact total and his proof was far more convincing.

A deeper discussion of the history of the Burali-Forti Paradox can be found in [Copi, 1958].

3.2 Paradoxical Universes

We will now try to abstract the essence of the Burali-Forti paradox, in order to obtain something that we can formalize in λU and λU^-. We first observe that there are two critical functions which are used in the proof of the Burali-Forti paradox:

- A function $\sigma : \mathit{Ord} \to \wp\mathit{Ord}$ defined by:

$$\sigma(\alpha) := \left\{\beta \in \mathit{Ord} \mid \beta <_{\mathit{Ord}} \alpha\right\}$$

  Thus, σ fully characterizes the ordering on Ord.

- A function $\tau : \wp\mathit{Ord} \to \mathit{Ord}$ which assigns to each set X of ordinal numbers the unique ordinal number of $(X, <_{\mathit{Ord}})$.

Thus, we have two functions to move between Ord and $\wp\mathit{Ord}$. One naturally wonders if their compositions exhibit any specific properties.

Let $X \in \wp\mathit{Ord}$ be a set of ordinal numbers such that $X \in \mathit{Im}(\sigma)$. Then, for every $\beta \in X$ we have that all ordinal numbers smaller than β are also in X, and therefore every initial segment of $(X, <_{\mathit{Ord}})$ is of the form $(\sigma(\beta), <_{\mathit{Ord}})$ for some $\beta \in \mathit{Ord}$. From this we see:

$$\sigma \circ \tau(X) = \{\alpha \mid \alpha <_{\mathit{Ord}} \tau(X)\} = \{\tau \circ \sigma(\beta) \mid \beta \in X\} = (\tau \circ \sigma)^{\rightarrow}(X). \tag{3.3}$$

It is this property that we capture in the next definition, albeit in a slightly stronger form (since we will require this property to be true for all $X \in \wp\mathit{Ord}$).

Definition 3.2.1. A triple $(\mathcal{U}, \sigma, \tau)$ with $\mathcal{U}$ a set, $\sigma : \mathcal{U} \to \wp\mathcal{U}$ and $\tau : \wp\mathcal{U} \to \mathcal{U}$ is called a **paradoxical universe** iff for each $X \subseteq \mathcal{U}$ we have $\sigma \circ \tau(X) = (\tau \circ \sigma)^{\rightarrow}(X)$.

For $x, y \in \mathcal{U}$, we will say that x is a **predecessor** of y (written as $x < y$) iff $x \in \sigma(y)$.

As we will shortly see, this notion captures enough of the essence of the Burali-Forti paradox to obtain a contradiction from the existence

of such a paradoxical universe, in a method quite like the one employed in the derivation of the Burali-Forti paradox above. Afterwards, we will construct a paradoxical universe in λU^-, finishing our proof of its inconsistency.

In order to mimic our derivation of the Burali-Forti paradox, we want to use something like transfinite induction. However, we do not know if $<$ is well-founded, so we cannot use transfinite induction directly. We remedy this by only looking at those elements of $\mathcal{U}$ which behave decently enough to still comply with transfinite induction. This is formulated below using subsets instead of predicates; however, as discussed in section 2.2, these are essentially the same.

Definition 3.2.2. Let $(\mathcal{U}, \sigma, \tau)$ be a paradoxical universe.

A subset $X \subseteq \mathcal{U}$ is called **inductive** iff for each $x \in \mathcal{U}$, if all predecessors are in X, then $x \in X$; i.e. $\forall x \in \mathcal{U}[\forall y \in \mathcal{U}[y < x \Rightarrow y \in X] \Rightarrow x \in X]$.

We call an element $x \in \mathcal{U}$ **decent** iff x is in each inductive $X \subseteq \mathcal{U}$.

We now proceed with the main result of this section: the derivation of a contradiction from the existence of a paradoxical universe.

Theorem 3.2.3. *There exists no paradoxical universe.*

Proof. Let $(\mathcal{U}, \sigma, \tau)$ be a paradoxical universe.

Since we want to mimic the Burali-Forti paradox, we want to define Ω to be the τ of some large set such that both $\tau \circ \sigma(\Omega) < \Omega$ and $\tau \circ \sigma(\Omega) \not< \Omega$. However, since we want to use transfinite induction in our argument, we want Ω to be decent. To accomplish this, we do not take $\tau(\mathcal{U})$ but instead we define $\Omega := \tau(\{x \in \mathcal{U} \mid x \text{ is decent}\})$.

In the proof of the Burali-Forti paradox we took Ω to be the τ of *all* ordinal numbers. However, one can easily check through transfinite induction that all ordinal numbers are decent. Thus, in the case of ordinal numbers this definition coincides with the one in the Burali-Forti paradox.

We will first show that Ω is decent. So, let $X \subseteq \mathcal{U}$ be inductive. We need to show that $\Omega \in X$. Since X is inductive it is enough to show that all predecessors of Ω are in X. Since $(\mathcal{U}, \sigma, \tau)$ is paradoxical, those predecessors are:

$$\sigma(\tau(\{x \in \mathcal{U} \mid x \text{ is decent}\})) = \{\tau \circ \sigma(w) \mid w \in \mathcal{U} \mid w \text{ is decent}\} \quad (3.4)$$

Therefore, let $w \in \mathcal{U}$ be decent. To show that $\tau \circ \sigma(w) \in X$ it is enough to show that the set $(\tau \circ \sigma)^{-1}(X)$ is inductive:

Let $x \in \mathcal{U}$ be such that for each $y < x$ we have $y \in (\tau \circ \sigma)^{-1}(X)$, i.e. $\tau \circ \sigma(y) \in X$. The predecessors of $\tau \circ \sigma(x)$ are:

$$\sigma(\tau \circ \sigma(x)) = \{\tau \circ \sigma(y) \mid y \in \sigma(x)\} = \{\tau \circ \sigma(y) \mid y < x\} \tag{3.5}$$

Thus, all predecessors of $\tau \circ \sigma(x)$ are in X, and therefore $\tau \circ \sigma(x) \in X$, which is the same as $x \in (\tau \circ \sigma)^{-1}(X)$. So, $(\tau \circ \sigma)^{-1}(X)$ is indeed inductive, as claimed.

Since w is decent, we therefore have $w \in Y$, which is equivalent to $\tau \circ \sigma(w) \in X$. Thus, all predecessors of Ω are in X and since X is inductive this implies that $\Omega \in X$. Therefore, Ω is indeed decent.

We thus see see from (3.4) that $\tau \circ \sigma(\Omega) < \Omega$. On the other hand, the set $Z := \{y \in \mathcal{U} \mid \tau \circ \sigma(y) \not< y\}$ is inductive:

Let $x \in \mathcal{U}$ be such that for each $y < x$ we have $y \in Z$, i.e. $\tau \circ \sigma(y) \not< y$. Suppose that $\tau \circ \sigma(x) < x$. Then (by taking $y = \tau \circ \sigma(x)$) we see that $\tau \circ \sigma(\tau \circ \sigma(x)) \not< \tau \circ \sigma(x)$. However, from (3.5) we also know that $\tau \circ \sigma(\tau \circ \sigma(x)) < \tau \circ \sigma(x)$. Thus, $\tau \circ \sigma(x) \not< x$, which is the same as $x \in Z$. Therefore, Z is indeed inductive.

Since Ω is decent, we thus see $\tau \circ \sigma(\Omega) \not< \Omega$. This contradicts our earlier observation that $\tau \circ \sigma(\Omega) < \Omega$ □

3.3 A Paradoxical Universe in λU^-

It is easy to see that the entire argument above can be formalized in λHOL. However, λHOL is not expressive enough to find a paradoxical universe (in the empty context), which of course makes sense if one recalls that λHOL is consistent. We will now try to find a paradoxical universe which we can formalize in λU^-.

Since λU^- is obtained from λHOL by making *Type* impredicative and since the Burali-Forti paradox heavily relies on impredicativity (namely by taking the ordinal number of the set of all ordinal numbers), it would make sense to construct a paradoxical universe through impredicative means. Therefore, we choose to study pairs (X, r) with $X : \mathit{Type}$ and $r : \wp X \to X$. Such a pair can be seen as a paradoxical universe $(\mathcal{U}, \sigma, \tau)$ in which we have left out the function σ (in particular, the paradoxical universe that we are now constructing also yields such a pair, illustrating the impredicativity of our definition).

Intuitively, one can see r as a function which 'reflects' each subset of X back into X. Therefore, we will colloquially refer to such an r as a *reflection* of X.

We could assign to each such pair (X, r) an element of $\wp X$, e.g. we could send the pair (X, r) to $\{r(X), r(\emptyset)\}$, or we could take the entire image of r. These operations can intuitively be seen as 'reflecting the reflections'. We will construct a paradoxical universe by looking at all such reflections of reflections. Therefore, we define:

$$\mathcal{U} := \Pi X{:}\mathit{Type}.(\wp X \to X) \to \wp X$$

Thus, a term $u : \mathcal{U}$ sends a reflection (X, r) to a set $\wp X$. We have a natural way to turn this into a paradoxical universe.

First, we construct $\tau : \wp\mathcal{U} \to \mathcal{U}$. Let $V : \wp\mathcal{U}$. We want to construct a term of type $\mathcal{U}$, so let $X : \mathit{Type}$ and $r : \wp X \to X$. Looking at the definition of paradoxical universes, it would make sense to try finding a suitable term $\varphi_{(X,r)} : \mathcal{U} \to X$ and then take $\tau\, V\, X\, r$ to be $\varphi^{\rightarrow}_{(X,r)}\, V$.

So, let $f : \mathcal{U}$. Such an f gives us a term of type $\wp X$ by taking $f\, X\, r$. However, we can easily reflect this into a term of type X by taking $r\,(f\, X\, r)$. This leads us to the following definition:

$$\varphi_{(X,r)} : \mathcal{U} \to X := \lambda f{:}\mathcal{U}.r\,(f\, X\, r)$$

By the argument given above, we now define τ as follows:

$$\tau : \wp\mathcal{U} \to \mathcal{U} := \lambda V{:}\wp\mathcal{U}.\lambda X{:}\mathit{Type}.\lambda r{:}\wp X \to X.\varphi^{\rightarrow}_{X,r}\, V$$

We now have a natural way to construct $\sigma : \mathcal{U} \to \wp\mathcal{U}$. Let $f : \mathcal{U}$. Then we can look at how f reflects the reflection τ, that is, we can take $\sigma\, f$ to be $f\, \mathcal{U}\, \tau$. From this we obtain:

$$\sigma : \mathcal{U} \to \wp\mathcal{U} := \lambda f{:}\mathcal{U}.f\, \mathcal{U}\, \tau$$

It is now easy to check that $(\mathcal{U}, \tau, \sigma)$ forms a paradoxical universe. First observe that for every $f : \mathcal{U}$ we have $\varphi_{U,\tau}\, f =_\beta (\tau \circ \sigma)\, f$. Now we have for every $V \in \wp\mathcal{U}$:

$$(\sigma \circ \tau)\, V =_\beta (\tau\, V)\, \mathcal{U}\, \tau =_\beta \varphi^{\rightarrow}_{\mathcal{U},\tau}\, V =_\beta (\tau \circ \sigma)^{\rightarrow}\, V$$

Thus, $(\mathcal{U}, \tau, \sigma)$ forms a paradoxical universe. It is easy to see that this universe can be formalized in λU^-, which proves the inconsistency of λU^-.

Theorem 3.3.1. *λU^- is inconsistent.*

3.4 Powerful Universes

The Burali-Forti paradox can be further abstracted than we have done above, which leads us to a proof term which exhibits simpler reduction behavior. This time, we obtain our inspiration from the following functions on Ord and $\wp\wp\mathit{Ord}$:

- A function $\sigma' : \mathit{Ord} \to \wp\wp\mathit{Ord}$ defined by:

  $$\sigma'(\alpha) := \left\{V \in \wp\mathit{Ord} \mid \sigma(\alpha) \subseteq V\right\}$$

  So, a set $X \subseteq \mathit{Ord}$ is inductive in the sense of definition 3.2.2 iff for every $\alpha \in \mathit{Ord}$ we have $X \in \sigma'(\alpha) \Rightarrow \alpha \in X$. We can thus use σ' to fully characterizes the inductive subsets of Ord.

- A function $\tau' : \wp\wp\mathit{Ord} \to \mathit{Ord}$ which assigns to $C \in \wp\wp\mathit{Ord}$ the ordinal number of $\cap C$.

So, instead of looking at predecessors as we did for powerful universes, we now abstract to inductive subsets. Still, we do not fully lose sight of our predecessors, since for every $\alpha \in \mathit{Ord}$ we have $\cap(\sigma'(\alpha)) = \sigma(\alpha)$ which contains exactly the predecessors of α. However, the notion of predecessor now comes second to the notion of inductive sets.

The compositions still exhibit a nice property. Let $C \in \wp\wp\mathit{Ord}$ be such that $C \in \mathit{Im}(\sigma')$. Take $\alpha \in \mathit{Ord}$ such that $C = \sigma'(\alpha)$. Then we have:

$$\begin{aligned}\sigma' \circ \tau'(C) &= \left\{V \subseteq \mathit{Ord} \mid \sigma \circ \tau(\cap C) \subseteq V\right\} \\ &= \left\{V \subseteq \mathit{Ord} \mid \sigma \circ \tau(\sigma(\alpha)) \subseteq V\right\}\end{aligned}$$

Using equation 3.3, we now find:

$$\begin{aligned}&= \left\{V \subseteq \mathit{Ord} \mid \{\tau \circ \sigma(\beta) \mid \beta \in \sigma(\alpha)\} \subseteq V\right\} \\ &= \left\{V \subseteq \mathit{Ord} \mid \sigma(\alpha) \subseteq (\tau \circ \sigma)^{\leftarrow}(V)\right\}\end{aligned}$$

Finally, using the definition of σ':

$$\begin{aligned}&= \left\{V \subseteq \mathit{Ord} \mid (\tau \circ \sigma)^{\leftarrow}(V) \in C\right\} \\ &= ((\tau \circ \sigma)^{\leftarrow})^{\leftarrow}(C) = ((\tau' \circ \sigma')^{\leftarrow})^{\leftarrow}(C)\end{aligned}$$

We capture our new-found abstraction in the next definition, again in a slightly stronger form.

Definition 3.4.1. A triple $(\mathcal{U}, \sigma, \tau)$ with $\mathcal{U}$ a set, $\sigma : \mathcal{U} \to \wp\wp\mathcal{U}$ and $\tau : \wp\wp\mathcal{U} \to \mathcal{U}$ is called a **powerful universe** iff for each $C \in \wp\wp\mathcal{U}$ we have $\sigma \circ \tau(X) = ((\tau \circ \sigma)^{\leftarrow})^{\leftarrow}(C)$.

A set $X \subseteq \mathcal{U}$ is called **inductive** iff $\forall x \in \mathcal{U}[X \in \sigma(x) \Rightarrow x \in X]$, and we call an element $x \in \mathcal{U}$ **decent** iff x is in each inductive $X \subseteq \mathcal{U}$.

For $x, y \in \mathcal{U}$, we will say that x is a **predecessor** of y (written as $x < y$) iff $x \in \cap(\sigma(y))$.

We now proceed with the main result of this section: we show that we can derive a contradiction from the existence of a powerful universe.

Theorem 3.4.2. *There exists no powerful universe.*

Proof. Let $(\mathcal{U}, \sigma, \tau)$ be a powerful universe.

We once again want Ω to be the τ of some large set such that both $\tau \circ \sigma(\Omega) < \Omega$ and $\tau \circ \sigma(\Omega) \not< \Omega$. We also once again want Ω to be decent. To accomplish this, we take $\Omega := \tau(\{X \subseteq \mathcal{U} \mid X \text{ is inductive}\})$.

One can easily check, through transfinite induction, that the only inductive subset of Ord is Ord itself. Thus, in the case of ordinal numbers, Ω remains the same as in the Burali-Forti paradox.

We will first show that Ω is decent. So, let $X \subseteq \mathcal{U}$ be inductive. We need to show that $\Omega \in X$, and since X is inductive it is thus enough to show that $X \in \sigma(\Omega)$. Since $(\mathcal{U}, \sigma, \tau)$ is powerful, we find:

$$\begin{aligned} \sigma(\Omega) &= \sigma(\tau(\{X \subseteq \mathcal{U} \mid X \text{ is inductive}\})) \\ &= \{X \subseteq \mathcal{U} \mid (\tau \circ \sigma)^{\leftarrow}(X) \text{ is inductive}\} \end{aligned} \tag{3.6}$$

We thus need to show that $(\tau \circ \sigma)^{\leftarrow}(X)$ is inductive. So, let $x \in \mathcal{U}$ with $(\tau \circ \sigma)^{\leftarrow}(X) \in \sigma(x)$. Since $(\mathcal{U}, \sigma, \tau)$ is powerful, we have:

$$\sigma \circ \tau \circ \sigma(x) = \{Y \subseteq \mathcal{U} \mid (\tau \circ \sigma)^{\leftarrow}(Y) \in \sigma(x)\} \tag{3.7}$$

Thus, $X \in \sigma \circ \tau \circ \sigma(x)$. Since X is inductive, this means that $\tau \circ \sigma(x) \in X$, or equivalently $x \in (\tau \circ \sigma)^{\leftarrow}(X)$. Therefore, $(\tau \circ \sigma)^{\leftarrow}(X)$ is indeed inductive.

By equation (3.6) we now see $X \in \sigma(\Omega)$. Since X is inductive, this means that $\Omega \in X$. Therefore, Ω is indeed decent.

We first show that $\tau \circ \sigma(\Omega) < \Omega$. Therefore, let $X \in \sigma(\Omega)$. By equation (3.6) this means that $(\tau \circ \sigma)^{\leftarrow}(X)$ is inductive. Since Ω is decent, we see that $\Omega \in (\tau \circ \sigma)^{\leftarrow}(X)$, i.e. $\tau \circ \sigma(\Omega) \in X$. We thus see that $\tau \circ \sigma(\Omega) < \Omega$. On the other hand, the set $Z := \{y \in \mathcal{U} \mid \tau \circ \sigma(y) \not< y\}$ is inductive:

Let $x \in \mathcal{U}$ be such that $Z \in \sigma(x)$. Suppose that $\tau \circ \sigma(x) < x$. Then (by taking $y = \tau \circ \sigma(x)$) we see that $\tau \circ \sigma(\tau \circ \sigma(x)) \not< \tau \circ \sigma(x)$. However, we claim that we also have $\tau \circ \sigma(\tau \circ \sigma(x)) < \tau \circ \sigma(x)$:

We let $X \in \sigma \circ \tau \circ \sigma(x)$. As seen in (3.7), this is the same as $(\tau \circ \sigma)^{\leftarrow}(X) \in \sigma(x)$. Since we assumed that $\tau \circ \sigma(x) < x$ we thus know that $\tau \circ \sigma(x) \in (\tau \circ \sigma)^{\leftarrow}(X)$, i.e. $\tau \circ \sigma \circ \tau \circ \sigma(x) \in X$. Therefore, we indeed have $\tau \circ \sigma(\tau \circ \sigma(x)) < \tau \circ \sigma(x)$.

From this contradiction we see that $\tau \circ \sigma(x) \not< x$, which is equivalent to $x \in Z$. Thus, Z is indeed inductive. Since Ω is decent, we see that $\Omega \in Z$, which means that $\tau \circ \sigma(\Omega) \not< \Omega$. This contradicts our earlier observation that $\tau \circ \sigma(\Omega) < \Omega$. □

3.5 A Powerful Universe in λU^-

The motivation here is mostly the same as for our paradoxical universe in section 3.3. The entire argument can still be formalized in λHOL, but we once again need more expressivity to construct a powerful universe.

This time, we study pairs (X, r) with $X : \mathcal{T}\mathit{ype}$ and $r : \wp\wp X \to X$. Such pairs can be seen as powerful universes $(\mathcal{U}, \sigma, \tau)$ in which we left out σ, not unlike what we did in the construction of our paradoxical universe.

Furthermore, we can once again see r as a kind of 'reflection', this time from $\wp\wp X$ into X. Just as for our paradoxical universe, we construct a powerful universe by looking at reflections of reflections. Therefore, we define:

$$\mathcal{U} := \Pi X{:}\mathcal{T}\mathit{ype}.(\wp\wp X \to X) \to \wp\wp X$$

We first construct $\tau : \wp\wp\mathcal{U} \to \mathcal{U}$. Looking at the definition of powerful universes and our paradoxical universe in section 3.3, the following is a natural candidate:

$$\tau : \wp\wp\mathcal{U} \to \mathcal{U} := \lambda V{:}\wp\wp\mathcal{U}.\lambda X{:}\mathcal{T}\mathit{ype}.\lambda r{:}\wp\wp X \to X.(\varphi^{\leftarrow}_{X,r})^{\leftarrow}\, V$$

The construction of σ is essentially the same as for the paradoxical universe:

$$\sigma : \mathcal{U} \to \wp\wp\mathcal{U} := \lambda f{:}\mathcal{U}.f\ \mathcal{U}\,\tau$$

It is once again easy to check that $(\mathcal{U}, \tau, \sigma)$ forms a powerful universe. First observe that for every $f : \mathcal{U}$ we still have $\varphi_{U,\tau}\, f =_\beta (\tau \circ \sigma)\, f$.

From this we obtain for every $V \in \wp\wp\mathcal{U}$:

$$(\sigma \circ \tau)\, V =_\beta (\tau\, V)\, \mathcal{U}\, \tau =_\beta (\varphi^{\leftarrow}_{\mathcal{U},\tau})^{\leftarrow}\, V =_\beta ((\tau \circ \sigma)^{\leftarrow})^{\leftarrow}\, V$$

Since also this powerful universe can be formalized in λU^-, we have another proof of the inconsistency of λU^-. This gives a term of type $\bot$, which can be seen below.

$$\Delta := \lambda y{:}\mathcal{U}.\neg\forall p{:}\wp\mathcal{U}((\sigma\, y\, p) \to (p\, \tau\, (\sigma\, y)))$$
$$\Omega := \text{the normal form of } \tau\, \lambda p{:}\wp\mathcal{U}.\forall x{:}\wp U.((\sigma\, x\, p) \to (p\, x))$$

$$(\lambda 0{:}\forall p{:}\wp\mathcal{U}.(\forall x{:}\mathcal{U}.((\sigma\, x\, p) \to (p\, x)) \to (p\, \Omega)).$$
$$(((0\, \Delta)\, \lambda x{:}\mathcal{U}.\lambda 2{:}(\sigma\, x\, \Delta).\lambda 3{:}\forall p{:}\wp\mathcal{U}.((\sigma\, x\, p) \to (p\, \tau\, (\sigma\, x))).$$
$$(((3\, \Delta)\, 2)\, \lambda p{:}\wp U.(3\, \lambda y{:}\mathcal{U}.(p\, \tau\, (\sigma\, y)))))$$
$$\lambda p{:}\wp U.(0\, \lambda y{:}\mathcal{U}.(0\, \lambda y{:}\mathcal{U}.(p\, \tau\, (\sigma\, y))))$$
$$\lambda p{:}\wp\mathcal{U}.\lambda 1{:}\forall x{:}\mathcal{U}.((\sigma\, x\, p) \to (p\, x)).((1\, \Omega)\, \lambda x{:}\mathcal{U}.(1\, \tau\, (\sigma\, x)))) : \bot$$

If one completely spells out the proof term of section 3.3, one can compare it with the proof term above. This way, we see that this proof term is indeed simpler in the sense that it contains fewer applications corresponding to *modus ponens*: the proof term corresponding to section 3.3 has 12 of such applications, while this term only has 6.

3.6 Reduction Behavior

The reduction behavior of Hurkens's paradox has been studied extensively. We briefly present the most important results.

Hurkens already observed that the term as presented above *almost* reduces to itself. The reason it does not exactly reduce to itself is that $\tau\, (\sigma\, x) \neq_\beta x$, which causes the types in the lambda-abstractions to explode. More details can be found in [Hurkens, 1995].

Because the term does not exactly reduce to itself, it is hard to turn this term into a fixed point combinator. One could try the approach as suggested in [Meyer and Reinhold, 1986]. This gives us a *looping combinator*, which can be roughly described as a combinator that is almost a fixed point combinator. Unfortunately, in [Barthe and Coquand, 2006] it is shown that this does not give us a fixed point combinator.

4

Encoding Naive Set Theory in λU and λU^-

An obvious method to prove the inconsistency of λU and λU^- would be to encode naive set theory in it, and then look at the term corresponding to a set-theoretic paradox (like Russell's paradox or the Burali-Forti paradox discussed above).

To this end, we will set up a correspondence between so-called *non-well-founded sets* and *pointed graphs* and encode these graphs in λU. Afterwards, we will show the same can be done in λU^-, be it with some minor changes.

4.1 Non-Well-Founded Sets

In this chapter we will make heavy use of pointed graphs. Therefore, we start with their definition.

Definition 4.1.1. A **graph** is a tuple (V, E) with V a set (called the **vertices**) and $E \subseteq V \times V$ (called the **edges**).

A **pointed graph** $((V, E), p)$ consists of a graph (V, E) and a distinguished vertex $c \in V$ (called the **point**).

We will draw edges (a, b) as an arrow from a to b, and when drawing a pointed graph we will draw the point as a square.

To simplify our further discussion, we introduce the concept of *children* of a vertex. Intuitively speaking, these are the elements directly 'below' a vertex.

Definition 4.1.2. Let (V, E) be a graph and let $v \in V$. Then $E_v := \{w \in V \mid (v, w) \in E\}$ is called the set of **children** of v.

The main idea of this chapter is that we want to represent sets using pointed graphs. Intuitively, one can represent a set x as a pointed graph

by letting the point be x itself, then drawing all the elements $u \in x$ and connecting them to x by an edge, then drawing all the elements of those elements and connecting them to the sets of which they are elements, and so on. An example can be seen below.

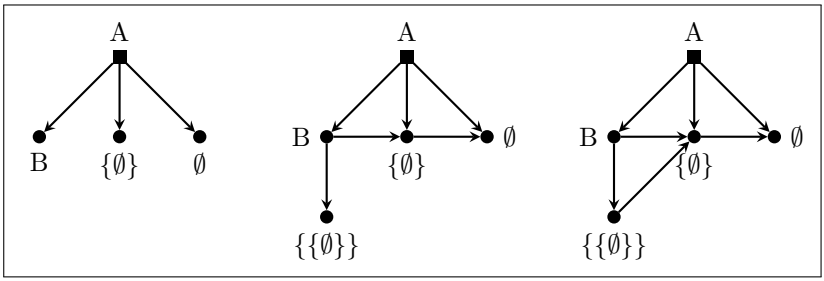


Figure 4.1: *Building a picture of* $A := \{\{\{\emptyset\}\}, \{\emptyset\}\}, \{\emptyset\}, \emptyset\}$ *where* $B := \{\{\{\emptyset\}\}, \{\emptyset\}\}$.

Thus, given a set we have constructed a graph and an assignment of sets to the vertices such that the set assigned to a vertex is the same as the set of the sets assigned to its children. We will call such an assignment a *decoration* of the graph and such a graph a *picture* of the set.

Definition 4.1.3. A **decoration** of a graph (V, E) is a collection $(x_v)_{v \in V}$ such that for every $v \in V$:

$$x_v = \{x_c \mid c \in E_v\}$$

A pointed graph $((V, E), p)$ is called a **picture** of a set x iff there exists a decoration $(x_v)_{v \in V}$ of (V, E) such that $x = x_p$.

This definition captures exactly what we wanted: all assignments in figure 4.1 are decorations and the last pointed graph is a picture of $\{\{\{\{\emptyset\}\}, \{\emptyset\}\}, \{\emptyset\}, \emptyset\}$. We can now formalize our intuitive argument as given above and show that we can build a picture for every set.

Proposition 4.1.4. $ZF^- \models$ *Every set has a picture.*

Proof. Let x be a set. We define $V := \mathrm{TC}(x)$ (where TC is as given in lemma 1.3.2) and $E := \{(v, w) \mid v, w \in V \mid w \in v\}$. It is then easy to see that the collection $(v)_{v \in V}$ is a decoration of (V, E), from which follows that $((V, E), x)$ is a picture of x. □

However, there are pointed graphs which, assuming ZF, are not a picture of any set. For example, consider the following pointed graph:

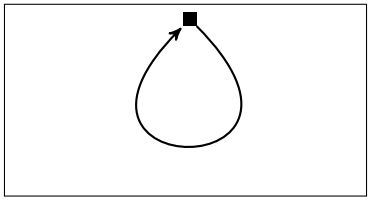


Figure 4.2: *A pointed graph which is not a picture in ZF.*

This pointed graph is a picture of the 'set' $\Omega = \{\Omega\}$. However, if one assumes ZF we see from corollary 1.3.1 that Ω is not a set. Nonetheless, it is not intuitively clear why we would not allow a set like this to exist. So, why don't we postulate that every pointed graph is the picture of a set, or equivalently that every graph has a decoration, instead of assuming the axiom scheme of foundation? We refer the unconvinced reader to the end of this section for some more (philosophical) discussion of this question.

Furthermore, we wonder if we should require the decorations of a graph to be unique. To this end, we first take a look what happens if one assumes ZF. In this case, the proposition below shows that every graph can have at most one decoration.

Proposition 4.1.5. *ZF* $\models$ *Every graph has at most one decoration.*

Proof. Let (V, E) be a graph and let $(x_v)_{v\in V}$, $(y_v)_{v\in V}$ be two decorations of (V, E). Let $\varphi(x) := \forall v \in V[x = x_v \to x = y_v]$. We prove, using $\in$-induction: $\forall x[\varphi(x)]$.

So, let x be a set and let $v \in V$ such that $x = x_v$. Since $(x_w)_{w\in V}$ is a decoration, we know that $x_v = \{x_c \mid c \in E_v\}$. Therefore, by induction hypothesis we know that $\varphi(x_c)$ holds for all $c \in E_w$, which in particular implies $x_c = y_c$. From this we see, using extensionality:

$$x = x_v = \{x_c \mid c \in E_v\} = \{y_c \mid c \in E_v\} = y_v$$

Thus, we obtain using $\in$-induction: $\forall x[\varphi(x)]$. In particular, we see that for all $v \in V$ we have that $\varphi(x_v)$ holds, which implies $x_v = y_v$. Therefore the graph can have at most one decoration. □

In spirit of this proposition, it makes sense to not only require each graph to have a decoration, but to also require this decoration to be

unique. Therefore, we replace the axiom scheme of foundation by the following axiom, which is due to Peter Aczel in [Aczel, 1988].

Axiom 4.1 (Anti-Foundation Axiom (AFA)). Every graph has a unique decoration.

The first question which arises from this axiom is the question if ZF^-+AFA is consistent relative to ZF^-. We postpone this question until the end of this section, where we will prove the relative consistency in theorem 4.1.10.

To simplify our use of the anti-foundation axiom, we introduce a notation for the unique set to which a pointed graph corresponds

Definition 4.1.6. Let $((V, E), p)$ be a pointed graph and let $(x_v)_{v \in V}$ be its unique decoration. Then we define $d((V, E), p) := x_p$, i.e. $d((V, E), p)$ is the unique set of which $((V, E), p)$ is a picture.

So, we now know that we can assign a unique set to each pointed graph. However, we wonder if the converse is true: does each set have a unique picture? Existence is guaranteed by proposition 4.1.4, however, figure 4.3 shows that uniqueness in general does not hold.

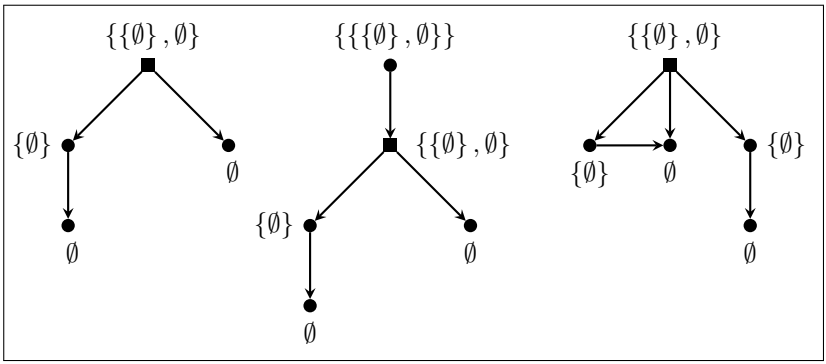


Figure 4.3: *Three pictures (with their unique decoration) of the Von Neumann natural number* 2*.*

Nonetheless, we do have a weaker result: we will show below that each set has a unique picture up to *bisimulation*. Intuitively, a bisimulation is a relation which respects the pointed graph-structure, i.e. it relates the points and respects arrows. This is made precise in the definition below.

Definition 4.1.7. Let $((V, E), p)$ and $((W, F), q)$ be pointed graphs. Then a **bisimulation** from $((V, E), p)$ to $((W, F), q)$ is a relation $R \subseteq V \times W$ such that:

- pRq;
- For all $v \in V$, $w \in W$ such that vRw:
  - For all $a \in E_v$ there exists a $b \in F_w$ such that aRb.
  - For all $b \in F_w$ there exists an $a \in E_v$ such that aRb.

Two pointed graphs $((V, E), p)$ and $((W, F), q)$ are called **bisimular** (notation: $((V, E), p) \simeq ((W, F), q)$) iff there exists a bisimulation from $((V, E), p)$ to $((W, F), q)$.

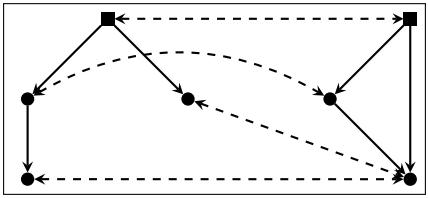


Figure 4.4: *Example of a bisimulation.*

We will now proceed with the main result of this section: each set has a unique picture up to bisimulation.

Theorem 4.1.8. *ZF^{-}+AFA $\models$ Let $((V, E), p)$ and $((W, F), q)$ be two pointed graphs. Then:*

$$d((V, E), p) = d((W, F), q) \Leftrightarrow ((V, E), p) \simeq ((W, F), q)$$

Proof. Let $(x_v)_{v \in V}$ be the unique decoration of $((V, E), p)$ and let $(y_w)_{w \in W}$ be the unique decoration of $((W, F), q)$.

First, let $d((V, E), p) = d((W, F), q)$. Define the relation $R \subseteq V \times W$ through $vRw \Leftrightarrow x_v = y_w$. Then it is easily checked that R is a bisimulation from $((V, E), p)$ to $((W, F), q)$. Thus, $(V, E), p) \simeq ((W, F), q)$.

Conversely, let $((V, E), p) \simeq ((W, F), q)$. So, let S be a bisimulation from $((V, E), p)$ to $((W, F), q)$.

Define:

$$G := \{((v_1, w_1), (v_2, w_2)) \mid (v_1, w_1), (v_2, w_2) \in S \mid v_2 \in E_{v_1} \wedge w_2 \in F_{w_1}\}$$

If we let $X_{(a,b)} := x_a$ and $Y_{(a,b)} := y_b$, then $(X_s)_{s\in S}$ and $(Y_s)_{s\in S}$ are both decorations of (S, G):

Let $(v, w) \in S$. Since G is a bisimulation we know that for all $a \in E_v$ there exists a $b \in F_w$ such that $(a, b) \in S$, thus $(a, b) \in G_{(v,w)}$. Conversely, if $(a, b) \in G_{(v,w)}$ then in particular $a \in E_v$. Using this we see:

$$\begin{aligned} X_{(v,w)} = x_v = \{x_a \mid a \in E_v\} &= \left\{x_a \mid (a, b) \in G_{(v,w)}\right\} \\ &= \left\{X_{(a,b)} \mid (a, b) \in G_{(v,w)}\right\}. \end{aligned}$$

Therefore, $(X_s)_{s\in S}$ is indeed a decoration of (S, G). Analogously we find that $(Y_s)_{s\in S}$ is also a decoration of (S, G).

From this we see that $((S, G), (p, q))$ is a picture of both $d((V, E), p)$ and $d((W, F), q)$. Thus, by the anti-foundation axiom: $d((V, E), p) = d((W, F), q)$. □

Together with proposition 4.1.4 this tells us that a set is characterized by an equivalence class of bisimular pointed graphs. The next corollary tells us how the $\in$-relation on sets translates to these equivalence classes.

Corollary 4.1.9. *ZF⁻+AFA* $\models$ *Let* $((V, E), p)$ *and* $((W, F), q)$ *be two pointed graphs. Then* $d((V, E), p) \in d((W, F), q) \Leftrightarrow \exists q' \in W[(q, q') \in F \wedge ((V, E), p) \simeq ((W, F), q')$.

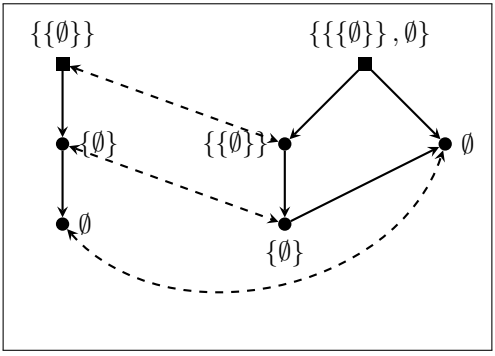


Figure 4.5: *Example of the lifted $\in$-relation.*

We now exhibit an important theoretical result, which we promised when we introduced the anti-foundation axiom: the consistency of ZF⁻+AFA relative to ZF⁻.

Theorem 4.1.10. *If ZF^- is consistent, then so is ZF^-+AFA.*

Proof. The proof is quite technical, so we only sketch the main idea. The details can be found in [Barwise and Moss, 1996, Chapter 9].

We translate each formula φ to a formula $\tilde{\varphi}$ as follows:

- We replace each quantification over sets by a quantification over pointed graphs.
- We replace set equality by bisimulation.
- We replace the $\in$ relation by the relation as given in corollary 4.1.9.

We claim that, if φ is provable using ZF^-+AFA, then $\tilde{\varphi}$ is provable using ZF^-. To prove this, it is enough to show that the translation of each axiom of ZF^-+AFA is provable using ZF^-; this is shown in [Barwise and Moss, 1996, Chapter 9].

The relative consistency now follows by considering $\varphi := \bot$. □

Further discussion on the axiom scheme of foundation and AFA

The axiom scheme of foundation is often justified through the *cumulative hierarchy* of sets. In this motivation, the universe of sets is built in stages, with one stage for every ordinal number. At stage 0, we only have the empty set. At every later stage, we form all the sets which we can make by using sets from the previous stage. The universe of sets is then the collective union of all these stages.

It is easy to check that all sets made this way satisfy the axiom scheme of foundation (by transfinite induction over the stages). Still, since the cumulative argument does not capture the way we actually form sets, the argument fails to convince.

Furthermore, the axiom scheme of foundation is not used in our 'daily' mathematical activity. It is more of a set-theoretical invention which can be used to shape some order in the chaotical world of set theory. Or, as Paul Cohen stated in [Cohen, 1966]: "This axiom is a somewhat artificial one and we include it for technical reasons only".

Even worse, recently non-well-founded sets (that is, sets which defy the axiom scheme of foundation) appeared in practice. For example, Aczel's original reason to study non-well-founded sets grew out of a problem in computer science on the theory of *concurrent processes*.

This shows that the axiom scheme of foundation is not as innocent as some people make it out to be, and opens the discussion for replacing the axiom scheme of foundation with the anti-foundation axiom.

We further remark that the anti-foundation axiom we have introduced is not the only way to introduce non-well-founded sets. Two other (non-equivalent) anti-foundation axioms are due to P. Finsler and Dana Scott. More on them can be found in e.g. [Aczel, 1988].

4.2 Interpretation in λU

As mentioned above, we want to prove the inconsistency of λU by interpreting naive set theory in it. In spirit of theorem 4.1.8, we can see a set as an equivalence class of bisimular pointed graphs. Therefore, we will be formalizing our sets as pointed graphs, as done by Alexandre Miquel in [Miquel, 2001, chapter 8].

Definition 4.2.1. A **pointed graph** in λU is a tuple $((V, E), p)$ with $V : \mathcal{Type}$, $E : V \to V \to \mathcal{Prop}$ and $p : V$.

To simplify our notation, we introduce abbreviations as follows.

Definition 4.2.2.

$$\begin{aligned}
\mathcal{PGraph} \to M &:= \Pi V{:}\mathcal{Type}.(V \to V \to \mathcal{Prop}) \to V \to M \\
\lambda((V, E), p).M &:= \lambda V{:}\mathcal{Type}.\lambda E{:}V \to V \to \mathcal{Prop}.\lambda p{:}V.M \\
\forall((V, E), p).M &:= \forall V{:}\mathcal{Type}.\forall E{:}V \to V \to \mathcal{Prop}.\forall p{:}V.M \\
\exists((V, E), p).M &:= \exists V : \mathcal{Type}.\exists E : V \to V \to \mathcal{Prop}.\exists p : V.M
\end{aligned}$$

We now define bisimularity in λU and characterize $\in$ using corollary 4.1.9.

Definition 4.2.3.

$$\begin{aligned}
&\mathcal{EQV} : \mathcal{PGraph} \to \mathcal{PGraph} \to \mathcal{Prop} := \\
&\quad \lambda((V, E), p).\lambda((W, F), q).\exists R : (V \to W \to \mathcal{Prop}). \\
&\quad (R\,p\,q) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge \\
&\quad (\forall v{:}V.\forall a{:}V.\forall w{:}W.(E\,v\,a) \to (R\,v\,w) \\
&\qquad\quad \to \exists b : W.(R\,a\,b) \wedge (F\,w\,b)) \qquad\qquad\qquad\qquad \wedge \\
&\quad (\forall w{:}W.\forall b{:}W.\forall v{:}V.(F\,w\,b) \to (R\,v\,w) \\
&\qquad\quad \to \exists a V.(R\,a\,b) \wedge (E\,v\,a))
\end{aligned}$$

We will write $((V,E),p) \simeq ((W,F),q)$ for $\mathcal{EQV}\, V\, E\, p\, W\, F\, q$.

$$\begin{aligned}&\mathcal{ELT} : \mathcal{PGraph} \to \mathcal{PGraph} \to \mathcal{Prop} := \\ &\quad \lambda((V,E),p).\lambda((W,F),q).\exists q' : W. \\ &\quad (F\, q\, q') \wedge (((V,E),p) \simeq ((W,F),q'))\end{aligned}$$

We will write $((V,E),p) \in ((W,F),q)$ for $\mathcal{ELT}\, V\, E\, p\, W\, F\, q$.

From theorem 4.1.8 we know, amongst other things, that $\simeq$ is an equivalence relation. However, we cannot formalize this argument in λU, since we do not have any set theory in λU (yet). Since we do however want to use this fact, we prove it directly in λU.

Proposition 4.2.4. *Bisimilarity $\simeq$ is an equivalence relation on the universe of pointed graphs.*

Proof. Let $((V,E))p)$, $((W,F),q)$ and $((X,G),r)$ be pointed graphs.
It is easy to check that:

- The relation R given by $\lambda v{:}V.\lambda w{:}V.v =_V w$, where $=_V$ is Leibniz-equality on V, is a bisimulation from $((V,E),p)$ to $((V,E),p)$.
- If S is a bisimulation from $((V,E),p)$ to $((W,F),q)$, then R defined by $\lambda w{:}W.\lambda v{:}V.S\, w\, v$ is a bisimulation from $((W,F),q)$ to $((V,E),p)$.
- If S is a bisimulation from $((V,E),p)$ to $((W,F),q)$ and T is a bisimulation from $((W,F),q)$ to $((X,G),r)$, then R defined by $\lambda v{:}V.\lambda x{:}X.\exists w{:}W.S\, v\, w \wedge T\, w\, x$ is a bisimulation from $((V,E),p)$ to $((X,G),r)$.

Thus, $\simeq$ is indeed an equivalence relation. □

Since we want to use $\simeq$ as our equality, we show that it obeys the substitution property for $\in$.

Lemma 4.2.5. *If $((V,E),p) \simeq ((W,F),q)$, $((X,G),r) \simeq ((Y,H),s)$ and $((V,E),p) \in ((X,G),r)$, then $((W,F),q) \in ((Y,H),s)$.*

Proof. Since $((V,E),p) \in ((X,G),r)$, there exists an $r' : X$ with $G\, r\, r'$ and $((V,E),p) \simeq ((X,G),r')$.

Let R be a bisimulation from $((X,G),r)$ to $((Y,H),s)$. Since we have $G\, r\, r'$ there exists an $s' : Y$ with $H\, s\, s'$ and $R\, r'\, s'$. But then

it is easily checked that R is also a bisimulation from $((X,G),r')$ to $((Y,H),s')$, thus $((X,G),r') \simeq ((Y,H),s')$.

This gives us $((W,F),q) \simeq ((V,E),p) \simeq ((X,G),r') \simeq ((Y,H),s')$ and we have $H\, s\, s'$, so indeed $((W,F),q) \in ((Y,H),s')$. □

We now have a means of interpreting sets in λU, through pointed graphs. Since we want to form impredicative set-theoretic paradoxes, we would like to have a type corresponding to the collection $\mathcal{PGraph}$ of all pointed graphs. Because of the lack of Σ-types in λU, this is not directly possible.

However, we can find a type for $\wp\wp\mathcal{PGraph}$, and we can easily embed all pointed graphs in this type by sending each pointed graph $((V,E,p)$ to $\{Y \in \wp\mathcal{PGraph} \mid ((V,E),p) \in Y\}$. We formalize this as follows.

Definition 4.2.6.

$$\begin{aligned} \mathcal{U} : \mathit{Type} \quad &:= (\mathcal{PGraph} \to \mathit{Prop}) \to \mathit{Prop} \\ i : \mathcal{PGraph} \to \mathcal{U} \quad &:= \lambda((V,E),p).\lambda P{:}\mathcal{PGraph} \to \mathit{Prop}.P\, V\, E\, p \end{aligned}$$

Observe that we need the $(\mathit{Kind}, \mathit{Type})$ quantification of λU and λU^- to define $\mathcal{U}$. Thus, the above cannot be done in λHOL.

The i given above is injective, in the sense as described in the lemma below.

Lemma 4.2.7. *For all pointed graphs* $((V,E),p),((W,F),q)$ *satisfying* $(i\, V\, E\, p) =_{\mathcal{U}} (i\, W\, F\, q)$ *we have* $((V,E),p) \simeq ((W,F),q)$.

Proof. Consider the predicate $P : \mathcal{U} \to \mathit{Prop}$ with $P(u)$ expressing 'the singleton-set $\{((V,E),p)\}$ is in u', i.e.:

$$P := \lambda u{:}\mathcal{U}.u\,(\lambda((X,G),r).((V,E),p) \simeq ((X,G),r))$$

It is easily checked that $P\,(i\, V\, E\, p)$ and $((V,E),p) \simeq ((V,E),p)$ are β-convertible, and similarly that $P\,(i\, W\, F\, q)$ and $((V,E),p) \simeq ((W,F),q)$ are convertible. The first holds by reflexivity of $\simeq$, so we find $((V,E),p) \simeq ((W,F),q)$, as desired. □

Observe that i is not surjective. As should be intuitively clear, $\emptyset \in \wp\wp\mathcal{PGraph}$ is not in the image of i. We will call the elements in the image of i *sets*, since we are using i to embed pointed graphs, which in turn represent sets.

Definition 4.2.8. We call $u : \mathcal{U}$ a **set** iff u is in the image of $\boldsymbol{i}$, i.e. iff $\boldsymbol{set}\, u$ holds, where $\boldsymbol{set}$ is defined as:

$$\boldsymbol{set} : \mathcal{U}{\to}\mathcal{P}\boldsymbol{rop} := \lambda u{:}\mathcal{U}.\exists((V,E),p).u =_{\mathcal{U}} (\boldsymbol{i}\, V\, E\, p)$$

We want to use the fact that $\boldsymbol{i}$ is not surjective later on. Therefore, we let $\boldsymbol{out}$ be $\emptyset \in \wp\wp\mathcal{PG}\boldsymbol{raph}$ as described above and we prove that it is indeed not in the image of $\boldsymbol{i}$.

Definition 4.2.9.

$$\boldsymbol{out} : \mathcal{U} := \lambda P{:}(\mathcal{PG}\boldsymbol{raph}{\to}\mathcal{P}\boldsymbol{rop}).\bot$$

Lemma 4.2.10. *$\boldsymbol{out}$ is not a set.*

Proof. Assume there exists a pointed graph $((V,E),p)$ such that we have $(\boldsymbol{i}\, V\, E\, p) =_{\mathcal{U}} \boldsymbol{out}$. Let $P : \mathcal{U}{\to}\mathcal{P}\boldsymbol{rop}$ be the predicate with $P(u)$ expressing 'u contains $\mathcal{PG}\boldsymbol{raph}$', that is:

$$P : \mathcal{U}{\to}\mathcal{P}\boldsymbol{rop} := \lambda u{:}\mathcal{U}.u\,(\lambda((V,E),p).\top)$$

It is easily checked that $P(\boldsymbol{i}\, V\, E\, p)$ and $\top$ are convertible, and similarly that $P\,\boldsymbol{out}$ and $\bot$ are convertible. Since the first clearly holds, we thus reach $\bot$. Therefore, $\boldsymbol{out}$ is not a set. □

In order to finish our interpretation of sets in $\mathcal{U}$, we need to lift $\mathcal{EQV}$ and $\mathcal{ELT}$ to those elements of $\mathcal{U}$ which are a set, which we do below.

Definition 4.2.11.

$$\begin{aligned}&\boldsymbol{eqv} : \mathcal{U}{\to}\mathcal{U}{\to}\mathcal{P}\boldsymbol{rop} :=\\ &\quad \lambda u{:}\mathcal{U}.\lambda v{:}\mathcal{U}.\exists((V,E),p).\exists((W,F),q).\\ &\quad u =_{\mathcal{U}} (\boldsymbol{i}\, V\, E\, p) \wedge v =_{\mathcal{U}} (\boldsymbol{i}\, W\, F\, q) \wedge ((V,E),p) \simeq ((W,F),q)\end{aligned}$$

We will write $u \cong v$ for $\boldsymbol{eqv}\, u\, v$.

$$\begin{aligned}&\boldsymbol{elt} : \mathcal{U}{\to}\mathcal{U}{\to}\mathcal{P}\boldsymbol{rop} :=\\ &\quad \lambda u{:}\mathcal{U}.\lambda v{:}\mathcal{U}.\exists((V,E),p).\exists((W,F),q).\\ &\quad u =_{\mathcal{U}} (\boldsymbol{i}\, V\, E\, p) \wedge v =_{\mathcal{U}} (\boldsymbol{i}\, W\, F\, q) \wedge ((V,E),p) \in ((W,F),q)\end{aligned}$$

We will write $u \sqsubset v$ for $\boldsymbol{elt}\, u\, v$.

The lifted $\cong$ inhabits most of the properties of $\simeq$: it is a partial equivalence relation that is reflexive exactly on the sets, and it it obeys the substitution property for $\in$.

Proposition 4.2.12. *$\cong$ is a partial equivalence relation on $\mathcal{U}$, which is reflexive exactly on the sets.*

Proof. Proposition 4.2.4 tells us that $\simeq$ is an equivalence relation. From this, the reflexivity on sets and symmetry of $\cong$ directly follows. The transitivity easily follows by additionally using the injectivity from lemma 4.2.7.

Furthermore, it is easily seen from the definition of $\cong$ that it $u \cong v$ implies that u and v are both sets. Thus, in particular, $\cong$ cannot be reflexive on elements which are not sets. □

Lemma 4.2.13. *If $u \cong u'$, $v \cong v'$ and $u \in v$, then $u' \in v'$.*

Proof. Directly follows from lemma 4.2.5 and the injectivity in lemma 4.2.7. □

So, we have finished our embedding of pointed graphs in $\mathcal{U}$, and thus have an interpretation of sets, through pointed graphs, as the things we called sets in $\mathcal{U}$. Since $\mathcal{U} : \mathcal{T}ype$, we can use $\mathcal{U}$ to form new pointed graphs, and then use i to turn these pointed graphs back into sets in $\mathcal{U}$. We will use this to demonstrate we can perform comprehension on our sets in $\mathcal{U}$.

By comprehension, we usually mean that for every predicate Q on sets we can form the set consisting of exactly those sets x for which $Q\,x$ holds. However, sets are only determined by terms of type $\mathcal{U}$ up to $\cong$. Therefore, we will only look at predicates $P : U \to \mathcal{P}rop$ which are *compatible*; that is, predicates $P : U \to \mathcal{P}rop$ such that, if $P\,u$ holds and $u \cong v$, then also $Q\,v$ holds.

Definition 4.2.14.

$$\begin{aligned} &\mathit{compat} : (\mathcal{U} \to \mathcal{P}rop) \to \mathcal{P}rop := \\ &\qquad \lambda P{:}(\mathcal{U} \to \mathcal{P}rop).\forall u{:}\mathcal{U}.\forall v{:}\mathcal{U}.(P\,u) \to (u \cong v) \to (P\,v) \end{aligned}$$

We will call $P : \mathcal{U} \to \mathcal{P}rop$ **compatible** iff $\mathit{compat}\ P$ holds.

One can easily check, with the help of lemma 4.2.13, that the predicate $\lambda u{:}\mathcal{U}.\varphi$ is compatible if the formula φ is built by only using:

- The binary relations $u \cong v$ and $u \in v$;

- The logical connectives $\bot, \top, \neg, \wedge, \vee, \rightarrow$;
- The quantifications $\forall u{:}\mathcal{U}.(\mathit{set}\, u) \rightarrow \psi$ and $\exists u : \mathcal{U}.(\mathit{set}\, u) \wedge \psi$.

We now turn towards the construction that we will use to perform comprehension on the sets in $\mathcal{U}$. To this end, we first observe that we have a 'universal graph' which contains bisimilar copies of all graphs.

Proposition 4.2.15. *For every graph $((V, E), p)$ we have:*

$$((V, E), p) \simeq ((\mathcal{U}, \mathit{elt}^{\delta}), (i\, V\, E\, p))$$

Proof. It is easily checked that the relation

$$R : V{\rightarrow}U{\rightarrow}\mathcal{P}\mathit{rop} := \lambda v{:}V.\lambda u{:}\mathcal{U}.(i\, V\, E\, v) \cong u$$

is a bisimulation from $((V, E), p)$ to $((\mathcal{U}, \mathit{elt}^{\delta}), (i\, V\, E\, p))$. □

Now, let $P : \mathcal{U}{\rightarrow}\mathcal{P}\mathit{rop}$ be a predicate. We extend this universal graph by connecting something which is not a set, e.g. out, to all sets $u : \mathcal{U}$ for which $P\, u$ holds. If we call the new edge-relation $\mathcal{FOLD}\, P$, this leads to a pointed graph $((\mathcal{U}, \mathcal{FOLD}\, P), \mathit{out})$, which is a picture of the set we are looking for. This is formalized below.

Definition 4.2.16.

$$\mathcal{FOLD} : (U{\rightarrow}Prop){\rightarrow}U{\rightarrow}U{\rightarrow}\mathcal{P}\mathit{rop} := \lambda P{:}(\mathcal{U}{\rightarrow}\mathcal{P}\mathit{rop}).\lambda u{:}\mathcal{U}.\lambda v{:}\mathcal{U}.$$
$$\mathit{elt}^{\delta}\, u\, v \vee ((u =_{\mathcal{U}} \mathit{out}) \wedge (\mathit{set}\, v) \wedge (P\, v))$$

Observe that we only added edges starting from out, which is not a set. Therefore, for sets $u : \mathcal{U}$, the pointed graph $((\mathcal{U}, \mathcal{FOLD}\, P), u)$ should not have changed 'too much' from $((\mathcal{U}, \mathit{elt}^{\delta}), u)$. This is indeed the case, as can be seen in the next lemma.

Lemma 4.2.17. *For every predicate $P : \mathcal{U}{\rightarrow}\mathcal{P}\mathit{rop}$ and every $u : \mathcal{U}$ with $\mathit{set}\, u$, $((\mathcal{U}, \mathit{elt}^{\delta}), u) \simeq ((\mathcal{U}, \mathcal{FOLD}\, P), u)$.*

Proof. We define the relation R as Leibniz equality on $\mathcal{U}$ restricted to sets, i.e.:

$$R : \mathcal{U}{\rightarrow}\mathcal{U}{\rightarrow}\mathcal{P}\mathit{rop} := \lambda v{:}\mathcal{U}.\lambda w{:}\mathcal{U}.(\mathit{set}\, v) \wedge v =_{\mathcal{U}} w$$

Since out is not a set by lemma 4.2.10, it is easily checked that R is a bisimulation from $((\mathcal{U}, \mathit{elt}^{\delta}), u)$ to $((\mathcal{U}, \mathcal{FOLD}\, P), u)$. □

As seen above, $((\mathcal{U}, \mathcal{FOLD}\, P), \mathit{out})$ is a picture of the set we are looking for. Therefore, we use i to pull this pointed graph back into a set in $\mathcal{U}$.

Definition 4.2.18.

$$\mathit{fold} : (\mathcal{U}{\to}\mathcal{P}\mathit{rop}){\to}\mathcal{U} := \lambda P{:}(\mathcal{U}{\to}\mathcal{P}\mathit{rop}).i\ \mathcal{U}\ (\mathcal{FOLD}\, P)\ \mathit{out}$$

We will prove that $\mathit{fold}\ P$ is indeed the set we are looking for by proving introduction and elimination rules for it. Observe that we only need P to be compatible for the elimination rule; as can be seen below, it is not necessary for the introduction rule.

Proposition 4.2.19 (Introduction rule for fold). *Let $P : \mathcal{U}{\to}\mathcal{P}\mathit{rop}$ be a predicate. If $u : \mathcal{U}$ is a set such that $P\, u$ holds, then $u \in \mathit{fold}\ P$.*

Proof. Since u is a set, there exists a pointed graph $((V, E), p)$ such that $u =_{\mathcal{U}} (i\, V\, E\, p)$. Therefore, from proposition 4.2.15 and lemma 4.2.17 we see:

$$((V, E), p) \simeq ((U, \mathit{elt}^{\delta}), u) \simeq ((\mathcal{U}, \mathcal{FOLD}\, P), u)$$

Since, by hypothesis, we have $P\, u$, $\mathcal{FOLD}\, P\ \mathit{out}\ u$ holds. Therefore, we have:

$$u =_{\mathcal{U}} (i\, V\, E\, p) \in (i\, \mathcal{U}\, (\mathcal{FOLD}\, P)\, \mathit{out}) \equiv \mathit{fold}\ P \qquad \square$$

Proposition 4.2.20 (Elimination rule for fold). *Let $P : \mathcal{U}{\to}\mathcal{P}\mathit{rop}$ be a compatible predicate. If $u \in (\mathit{fold}\, P)$, then $P\, u$ holds.*

Proof. Since $u \in (\mathit{fold}\ P)$, we know that there exist pointed graphs $((V, E), p), ((X, G), r)$ such that $u =_{\mathcal{U}} (i\, V\, E\, P)$, $(\mathit{fold}\ P) =_{\mathcal{U}} (i\, X\, G\, r)$ and $((V, E), p) \in ((X, G), r)$.

By lemma 4.2.7, we see that $((\mathcal{U}, \mathcal{FOLD}\, P), \mathit{out}) \simeq ((X, G), r)$. Therefore, by lemma 4.2.5 we have $((V, E), p) \in ((\mathcal{U}, \mathcal{FOLD}\, P), \mathit{out})$. So, by definition of $\in$, there exists a $v : \mathcal{U}$ such that $((V, E), p) \simeq ((\mathcal{U}, \mathcal{FOLD}\, P), v)$ and $\mathcal{FOLD}\, P\ \mathit{out}\ v$ holds. Since $v \in \mathit{out}$ would imply that out is a set, which it is not by lemma 4.2.10, we therefore see from the definition of $\mathcal{FOLD}$ that $\mathit{set}\, v$ and $P\, v$ hold.

So, let $((W, F), q)$ be a pointed graph such that $v =_{\mathcal{U}} (i\, W\, F\, q)$. From proposition 4.2.15 and lemma 4.2.17 we now see:

$$((W, F), q) \simeq ((\mathcal{U}, \mathit{elt}^{\delta}), v) \simeq ((\mathcal{U}, \mathcal{FOLD}\, P), v) \simeq ((V, E), p)$$

Therefore, $v \cong u$. Since P is compatible we thus find that $P\, u$ holds. $\square$

From this, we can easily find a contradiction in λU, by interpreting Russell's paradox.

Theorem 4.2.21. *λU is inconsistent.*

Proof. Let $\Omega : \mathcal{U} := \mathit{fold}\ (\lambda u{:}\mathcal{U}.\neg(u \in u))$. From proposition 4.2.19 we find a proof $p : \neg(\Omega \in \Omega) \to (\Omega \in \Omega)$ and from proposition 4.2.20 we find a proof $q : (\Omega \in \Omega) \to \neg(\Omega \in \Omega)$.

Now: $(\lambda\zeta{:}\Omega \in \Omega.q\,\zeta\,\zeta)\,(p\,(\lambda\zeta{:}\Omega \in \Omega.q\,\zeta\,\zeta)) : \bot$. □

4.3 Interpretation in λU^-

The interpretation of set theory in λU as given above cannot be transferred to λU^- without any changes, since we use the $(\mathit{Kind}, \mathit{Prop})$ quantification on multiple occasions (e.g. to formalize proposition 4.2.4).

This can, however, be fixed by only looking at pointed graphs which have $\mathcal{U}$ as the type of its vertices. This still allows us to define $\mathcal{FOLD}$, while removing the need for the $(\mathit{Kind}, \mathit{Prop})$ quantification.

The definitions and proofs of the previous section can then largely be translated into λU^- by dropping all quantifications over types and instantiating them with $\mathcal{U}$. However, there are two exceptions, namely lemma 4.2.7 and lemma 4.2.10. In the proofs of these two lemmas, we explicitly used the definition of $\mathcal{U}$. So, in these proofs the predicate P should not drop the quantification over a type. It is easily checked that this still gives a typable term in λU^-.

Thus, we also find a contradiction in λU^-.

Theorem 4.3.1. *λU^- is inconsistent.*

4.4 Reduction Behavior

By formalizing the proof above in e.g. Coq, one can study the reduction behavior of the paradox and try to construct a fixed point combinator out of it. Since Coq uses a different type system (the *Calculus of Inductive Constructions*), this is not directly possible; however, by a slight modification to the source code we can turn the type system into λU or λU^-. This modification is discussed in appendix A, where we also include a (heavily inlined and optimized) formalization of the inconsistency proof in λU^-.

A logical attempt to turn this proof into a fixed point combinator would be to do the following:

- Change the Russell-predicate $\lambda u{:}\mathcal{U}.\neg(u \in u)$ into $P := \lambda u{:}\mathcal{U}.(u \in u) \to A$.
- Change the paradox into:

$$\lambda A{:}\mathcal{Type}.\lambda f{:}A{\to}A.(\lambda\zeta{:}\Omega \in \Omega.f\,(q\,\zeta\,\zeta))(p\,(\lambda\zeta{:}\Omega \in \Omega.f\,(q\,\zeta\,\zeta)))$$

Now observe that:

$$\begin{aligned}&(\lambda\zeta{:}\Omega \in \Omega.f\,(q\,\zeta\,\zeta))\,(p\,(\lambda\zeta{:}\Omega \in \Omega.f\,(q\,\zeta\,\zeta)))\\ &\to_\beta f\,(q\,(p\,(\lambda\zeta{:}\Omega \in \Omega.f\,(q\,\zeta\,\zeta)))\,(p\,(\lambda\zeta{:}\Omega \in \Omega.f\,(q\,\zeta\,\zeta))))\end{aligned}$$

Thus, this would give a fixed point combinator if $q\,(p\,M) =_\beta M$ for all proofs $M : (u \in u) \to A$. Since p is an introduction rule and q is an elimination rule for $\mathcal{FOLD}$, one would expect this to happen. Unfortunately, this is not the case.

The main reason why $q\,(p\,M) \neq_\beta M$ is most easily illustrated using a simple example. Let (E, p) and (F, q) be two pointed graphs and let $a : (E, p) \simeq (F, q)$. In lemma 4.2.4, we proved that $\simeq$ is an equivalence relation; from this we find a proof EQV_refl of the reflexivity of $\mathcal{EQV}$ and a proof EQV_trans of the transitivity of $\mathcal{EQV}$.

Now, what happens if we attach a reflexivity proof to a; i.e. what can we say about

$$b := \text{EQV\_trans}\,E\,p\,E\,p\,F\,q\,(\text{EQV\_refl}\,E\,p)\,a?$$

Intuitively, we would like this to be β-equal to a. Unfortunately, this is not the case. If the orginal bisimulation contained in a is A, then this bisimulation changes into:

$$\lambda v{:}V.\lambda w{:}W.\exists x{:}V.v =_{\mathcal{U}} x \wedge A\,x\,w$$

While this bisimulation is logically equivalent to the original one, it is not β-equal to it. In fact, the only thing that changes between the terms a and b is the bisimulation, not the proofs that the bisimulation is a bisimulation. Thus, b falls just short of being β-equal to a.

It is because of this problem that $q\,(p\,M) \neq_\beta M$: the transitivity is used in the compatibility of $\in$, which is in turn used in q. Aside from this compatibility rule, the term reduces properly.

Our next attempt tries to eliminate the compatibility from the proof. We inline the definition of $\mathcal{ELT}\;E\,p\,E\,p$ in P by directly specifying a witness of the $\exists$-quantifier. By studying the proof of proposition

4.2.19, we find a good candidate: $i\,E\,p$. Thus, we change $\mathcal{ELT}$ in the predicate P into:

$$Q := \lambda E{:}\mathcal{U}{\to}\mathcal{U}{\to}\mathcal{P}\mathit{rop}.\lambda p{:}\mathcal{U}.E\,p\,(i\,E\,p) \wedge \mathcal{EQV}\,E\,p\,E\,(i\,E\,p)$$

This eliminates the compatibility from our elimination rule. However, we now obtain a different problem: we need to show that, if $i\,E\,p =_{\mathcal{U}} i\,F\,q$ and $Q\,F\,q$ hold, then $Q\,E\,p$ holds. First we remark that we can strengthen lemma 4.2.7: in fact, $i\,E\,p =_{\mathcal{U}} i\,F\,q$ implies that $((\mathcal{U}, E), p) =_{\mathcal{PG}\mathit{raph}} ((\mathcal{U}, F), q)$. Unfortunately, even though it feels counter-intuitive, this equality is not strong enough to prove that $Q\,E\,p$ holds. The problem is that we can only apply this equality to predicates defined on all of $\mathcal{PG}\mathit{raph}$; and since we always have that $i\,X\,G\,r$ is of type $\mathcal{U}$ instead of of type X, we cannot define this general predicate. Thus, this approach seems like it leads to a dead end.

It might very well be possible that some further inlining and optimization leads to a fixed point combinator; unfortunately, we cannot give a decisive answer at the current time.

A

Coq Formalization

```
(* Formalization of Miquel's paradox.
   We follow the proof as outlined in the thesis, except
   we consider the version in Lambda U-, as discussed in
   section 3.3; that is, we only consider graphs over the
   type U. *)

(* This formalization is in Lambda U-, and therefore some
   modifications need to be made to the Coq source code.
   It is (more than) enough to change the following line
   in kernel/typeops.ml:
    | (Type u1, Type u2) -> Type (sup u1 u2)
   into:
    | (Type u1, Type u2) -> Type u1

   This changes the PTS-rule
   (Type i, Type j, Type max(i,j))
   into (Type i, Type j, Type i), which is strong enough
   for our purpose. *)

(* The type U. *)
Definition U :=
  (forall T: Type, (T -> T -> Prop) -> T -> Prop) -> Prop.

(* Definition of False *)
Definition False : Prop := forall P: Prop, P.

(* Definition of Leibniz-equality on U and proofs that it
   is an  equivalence relation *)
Definition leib_eq (u v: U) :=
```

```
  forall P: U -> Prop, P u -> P v.
Infix "==" :=
  leib_eq (at level 70, no associativity) : type_scope.

Lemma leib_eq_refl : forall u : U, u == u.
unfold leib_eq. intros. exact H.
Defined.

Lemma leib_eq_sym : forall u v : U, u == v -> v == u.
intros. apply H. apply leib_eq_refl.
Defined.

Lemma leib_eq_trans : forall u v w : U,
  u == v -> v == w -> u == w.
intros. apply H0. exact H.
Defined.

(* Notational shorthand for graphs *)
Definition Graph := (U -> U -> Prop).

(* Equivalence (bisimulation) on pointed graphs *)
Definition EQV :=
  fun (E: Graph) (p: U) (F: Graph) (q: U) =>
  forall Q: Prop,
  (forall R : U -> U -> Prop,(R p q) ->
  (forall Q1: Prop, forall v a w : U,(E v a) -> (R v w)
    -> (forall b : U,(R a b) -> (F w b) -> Q1) -> Q1)
  -> (forall Q1: Prop, forall w b v : U,(F w b) -> (R v w)
    -> (forall a : U,(R a b) -> (E v a) -> Q1) -> Q1)
-> Q) -> Q.

(* Proofs that EQV is an equivalence relation *)
Lemma EQV_refl : forall (E: Graph) (p: U), EQV E p E p.
unfold EQV. intros.
apply H with (fun v:U => fun w:U => v == w).
apply leib_eq_refl. intros. apply H2 with a.
apply leib_eq_refl. apply H1. exact H0. intros.
apply H2 with b. apply leib_eq_refl.
apply leib_eq_sym in H1. apply H1. exact H0.
Defined.
```

```
Lemma EQV_sym : forall (E: Graph) (p : U) (F: Graph)
  (q : U), (EQV E p F q) -> (EQV F q E p).
unfold EQV. intros. apply H. intros.
apply H0 with (fun v w : U => R w v).
exact H1. exact H3. exact H2.
Defined.

Lemma EQV_trans :
  forall (E: Graph) (p: U) (F: Graph) (q: U) (G: Graph)
  (r: U), (EQV E p F q) -> (EQV F q G r) -> (EQV E p G r).
unfold EQV. intros. apply H. intro S. intros. apply H0.
intro T. intros.
apply H1 with (fun v x : U => forall P : Prop,
  (forall w : U, (S v w) -> (T w x) -> P) -> P).
intros. apply H8 with q. exact H2. exact H5. intros.
apply H9. intros. apply H3 with v a w0. exact H8.
exact H11. intros. apply H6 with w0 b w. exact H14.
exact H12. intros. apply H10 with b0. intros.
apply H17 with b. exact H13. exact H15. exact H16.
intros. apply H9. intros. apply H7 with w b w0. exact H8.
exact H12. intros. apply H4 with w0 a v. exact H14.
exact H11. intros. apply H10 with a0. intros.
apply H17 with a. exact H15. exact H13. exact H16.
Defined.


(* Definition of the ELT (subgraph) relation *)
Definition ELT := fun (E : Graph) (p : U) (F: Graph)
  (q : U) => forall Q: Prop,
  (forall q' : U, (F q q') -> (EQV E p F q') -> Q) -> Q.

(* EQV is compatible with respect to ELT *)
Lemma EQV_compat : forall (E : Graph) (p : U) (F : Graph)
  (q : U) (G : Graph) (r : U) (H : Graph) (s : U),
  (EQV E p F q) -> (EQV G r H s) -> (ELT F q G r)
    -> (ELT E p H s).
unfold ELT. intros. apply H2. intro r'. intros. apply H1.
intros. apply H7 with r r' s. exact H4. exact H6.
intro s'. intros. apply H3 with s'. exact H10.
```

```
assert (EQV G r' H s'). unfold EQV. intros.
apply H11 with R. exact H9. exact H7. exact H8.
assert (EQV F q H s'). apply EQV_trans with G r'.
exact H5. exact H11. apply EQV_trans with F q.
exact H0. exact H12.
Defined.

(* Definition of i: we will write i2 for the version from
   the thesis and for the sake of clarity we will write i
   for the version curried with U. *)
Definition i2 (T: Type) (E: T -> T -> Prop) (p: T)
  (u: forall S: Type, (S -> S -> Prop) -> S -> Prop) :=
    u T E p.

Definition i (E: Graph) (p: U) := i2 U E p.

(* For the injectivity of i, we need the general
   definition of EQV. *)

Definition EQVg :=
  fun (S: Type) (E: S -> S -> Prop) (p: S) (T: Type)
  (F: T -> T -> Prop) (q: T) => forall Q: Prop,
  (forall R : S -> T -> Prop,(R p q) ->
  (forall Q1: Prop, forall v a: S, forall w: T,
    (E v a) -> (R v w)
      -> (forall b : T,(R a b) -> (F w b) -> Q1) -> Q1)
  -> (forall Q1: Prop, forall w b: T, forall v: S,
    (F w b) -> (R v w)
      -> (forall a : S,(R a b) -> (E v a) -> Q1) -> Q1)
  -> Q) -> Q.

(* The critical predicate in the proof that i is
   injective, including introduction and elimination
   rules. *)
Definition contains_singleton (E: Graph) (p: U) (u: U) :=
  u (fun (T: Type) (F: T -> T -> Prop) (q: T) =>
    EQVg U E p T F q).

Lemma contains_singleton_intro: forall (E: Graph) (p: U)
  (F: Graph) (q :U),
```

```
    EQV E p F q -> contains_singleton E p (i F q).
intros. unfold contains_singleton. unfold i. exact H.
Defined.

Lemma contains_singleton_elim: forall (E: Graph) (p: U)
  (F: Graph) (q :U),
    contains_singleton E p (i F q) -> EQV E p F q.
intros. unfold contains_singleton in H. unfold i in H.
exact H.
Defined.

(* Proof of the injectivity of i. *)
Lemma iinj: forall (E: Graph) (p : U) (F : Graph) (q : U),
  (i E p) == (i F q) -> (EQV E p F q).
intros. apply contains_singleton_elim. apply H.
apply contains_singleton_intro. apply EQV_refl.
Defined.

(* We define when an element of U is a set (i.e. when it
   is in the image of i). *)
Definition set (u : U) :=
  forall Q: Prop, (forall (E : Graph) (p : U),
    (u == (i E p)) -> Q) -> Q.

(* We state the following trivial lemma, to avoid having
   to prove it over and over. *)
Lemma i_set : forall (E : Graph) (p : U), set (i E p).
unfold set. intros. apply H with E p. apply leib_eq_refl.
Defined.

(* Definition of out. *)
Definition out : U := fun _ => False.

(* Definition of True. *)
Definition True := forall P: Prop, P -> P.

Lemma True_intro: True.
unfold True. intros. exact H.
Defined.
```

```
(* The critical predicate in the proof that out is not a
   set, including an introduction rule. *)
Definition contains_universe (u: U): Prop :=
 u (fun _ _ _ => True).

Lemma contains_universe_intro: forall (E: Graph) (p: U),
  contains_universe (i E p).
intros. unfold contains_universe. unfold i. unfold i2.
apply True_intro.
Defined.

Lemma out_not_set : (set out) -> False.
unfold set. intros. apply H. intros.
assert (contains_universe out). apply leib_eq_sym in H0.
apply H0. apply contains_universe_intro. apply H1.
Defined.

(* We skip the definition of eqv and the lemmas regarding
   it, since we will inline eqv to simplify our proofs.
   By inlining eqv, we can directly pass the witnesses of
   elements u of U being a set (i.e. the E and p such that
   u = i E p) instead of having to redetermine them every
   time. The main advantage of this is that we can isolate
   the usage of iinj as much as possible, thus making the
   reduction easier to trace (since iinj is the only proof
   whose reduction we cannot properly trace in CIC, and it
   is therefore the main term responsible for the
   non-normalisation of the term). *)

(* We do, however, need elt for our universal graph below.
   We directly give the dual version. *)
Definition eltd (u v: U) :=
  forall Q: Prop, (forall (E: Graph) (p: U) (F: Graph)
  (q: U),
  u == (i E p) -> v == (i F q) -> ELT F q E p -> Q) -> Q.

(* The next lemma proves the existence of an 'universal
   graph' *)
Lemma univ_graph : forall (E : Graph) (p : U),
  (EQV E p eltd (i E p)).
```

```
unfold EQV. intros.
apply H with
  (fun v u: U => forall P : Prop, (forall (F: Graph)
    (q: U), u == i F q -> EQV E v F q -> P) -> P).
intros. apply H0 with E p. apply leib_eq_refl.
apply EQV_refl. intros. apply H1. intros. apply H4.
intros. apply H6 with v a q. exact H0. exact H5.
intros. apply H2 with (i F b). intros. apply H10 with F b.
apply leib_eq_refl. unfold EQV. intros. apply H11 with R.
exact H8. exact H6. exact H7. unfold eltd. intros.
apply H10 with F q F b. exact H3. apply leib_eq_refl.
unfold ELT. intros. apply H11 with b. exact H9.
apply EQV_refl. intros. apply H1. intros. apply H4.
intros. apply H0. intros. assert (ELT F0 q0 F q).
apply EQV_compat with F0 q0 E0 p0. apply EQV_refl.
apply iinj. apply leib_eq_trans with w. apply leib_eq_sym.
exact H8. exact H3. exact H10. apply H11. intros.
apply H7 with q q' v. exact H12. exact H5. intros.
apply H2 with a. intros. apply H16 with F0 q0. exact H9.
apply EQV_trans with F q'. unfold EQV. intros.
apply H17 with R. exact H14. exact H6. exact H7.
apply EQV_sym. exact H13. exact H15.
Defined.

(* For the sake of optimalization, we will only define
   FOLD for our 'Russell-style' predicate. *)
Parameter A: Prop.
Definition P (E: Graph) (p: U) := (ELT E p E p) -> A.

Definition FOLD (u v : U) := forall Q: Prop,
  (eltd u v -> Q) -> (forall (E: Graph) (p: U),
    u == out -> v == (i E p) -> (P E p) -> Q) -> Q.

(* Lemma 3.2.17 *)
Lemma fold_small : forall (u : U), (set u)
  -> (EQV eltd u FOLD u).
unfold EQV. intros.
apply H0 with
  (fun v w : U => forall Q1: Prop,
    ((set v) -> (v == w) -> Q1) -> Q1).
```

```
intros. apply H1. exact H. apply leib_eq_refl. intros.
apply H1. intros. apply H2. intros. apply H3 with a.
intros. apply H9. apply leib_eq_sym in H5. apply H5.
apply i_set. apply leib_eq_refl. unfold FOLD. intros.
apply H9. apply H8. exact H1. intros. apply H1. intros.
apply H2. intros. apply H3 with b. intros. apply H4.
intros. apply H7. apply leib_eq_sym in H9. apply H9.
apply i_set. apply leib_eq_refl. apply leib_eq_sym in H6.
apply H6. exact H4. intros. apply H2. intros.
apply out_not_set. apply H4. apply H8. exact H7.
Defined.

(* The introduction rule for fold *)
Lemma fold_intro : forall (E: Graph) (p: U), (P E p)
  -> (ELT E p (FOLD) out).
intros. unfold ELT. intros. apply H0 with (i E p).
unfold FOLD. intros. apply H2 with E p.
apply leib_eq_refl. apply leib_eq_refl. exact H.
apply EQV_trans with eltd (i E p). apply univ_graph.
apply fold_small. apply i_set.
Defined.

(* The compatibility of our predicate, necessary for the
   elimination rule *)
Lemma P_compat : forall (E: Graph) (p: U) (F: Graph)
  (q: U), P E p -> EQV E p F q -> P F q.
unfold P. intros. apply H. apply EQV_compat with F q F q.
exact H0. apply EQV_sym. exact H0. exact H1.
Defined.

(* The elimination rule for fold *)
Lemma fold_elim : forall (E: Graph) (p: U),
  (ELT E p (FOLD) out) -> (P E p).
unfold P. intros. apply H. intro v. intros. apply H1.
intro. apply H3. intros. apply out_not_set.
apply leib_eq_sym in H4. apply H4. apply i_set.
intros. apply H2. intros. apply P_compat with E0 p0 E p.
exact H5. apply leib_eq_sym in H4.
apply EQV_trans with eltd v. apply H4. apply univ_graph.
apply EQV_trans with (FOLD) v. apply fold_small.
```

```
apply H4. apply i_set. apply EQV_sym. exact H2. exact H0.
Defined.

(* Our terms p and q *)
Lemma p : P FOLD out -> ELT FOLD out FOLD out.
intro. apply fold_intro. exact H.
Defined.

Lemma q : ELT FOLD out FOLD out -> P FOLD out.
intro. apply fold_elim. exact H.
Defined.

(* Our attempt at a fixed point combinator *)
Parameter f: A -> A.

Definition Fix: A :=
  (fun z : (ELT FOLD out FOLD out) => f (q z z))
  (p (fun z : (ELT FOLD out FOLD out) => f (q z z))).
```

Bibliography

P. Aczel. *Non-Well-Founded Sets*, volume 14 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford, 1988.

H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North Holland, Amsterdam, 1984.

H.P. Barendregt and E. Barendsen. Introduction to lambda calculus, 1998.

G. Barthe and T. Coquand. Remarks on the equational theory of non-normalizing pure type systems. *Journal of Functional Programming*, 16(2):137–155, 2006.

J. Barwise and L.S. Moss. *Vicious circles*, volume 60 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford, 1996.

C. Burali-Forti. Una questione sui numeri transfiniti. *Rendiconti del Circolo Matematico di Palermo*, 11:154–164, 1897a.

C. Burali-Forti. Sulle classi ben ordinate. *Rendiconti del Circolo Matematico di Palermo*, 11:260, 1897b.

G. Cantor. Beitrage zur Begründung der transfiniten Mengenlehre, II. *Mathematische Annalen*, 49:207–246, 1897.

P.J. Cohen. *Set Theory and The Continuum Hypothesis*. W. A. Benjamin, New York, 1966.

I.M. Copi. The Burali-Forti Paradox. *Philosophy of Science*, 25(4): 281–286, October 1958.

J.H. Geuvers and M.J. Nederhof. A modular proof of strong normalisation for the calculus of constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

K.G. Hagström. Note sur l'antinomie Burali-Forti. *Arkiv for Matematik, Astronomi och Fysik*, 10:1–4, 1914.

A.J.C. Hurkens. A Simplification of Girard's Paradox. In *Second International Conference on Typed Lambda Calculi and Applications, TLCA'95*, volume 902 of *Lecture Notes in Computer Science*, pages 266–278. Springer Berlin / Heidelberg, 1995.

A.R. Meyer and M.B. Reinhold. "Type" is not a type. *Proceedings POPL'86*, pages 287–295, 1986.

A. Miquel. *Le Calcul des Constructions Implicite: Syntaxe et Semantique*. PhD thesis, Université Paris 7, 2001.

F. van Raamsdonk. Logical verification, course notes, 2008.

W. Veldman. Axiomatische verzamelingenleer.

(λζ:Ω ϵ Ω.q ζ ζ) (p