A New Paradox in Type Theory

Thierry Coquand
Department of Computer Science, University of Göteborg/Chalmers
S-412 96 Göteborg, Sweden

April 13, 1994

Introduction

The aim of this paper is to present a new paradox for Type Theory, which is a type-theoretic refinement of Reynolds' result [24] that there is no set-theoretic model of polymorphism. We discuss then one application of this paradox, which shows unexpected connections between the principle of excluded middle and the axiom of description in impredicative Type Theories.

1 Minimal and Polymorphic Higher-Order Logic

1.1 Minimal Higher-Order Logic

1.1.1 A presentation of the system

We assume known simply typed lambda calculus (see for instance [3].) The lambda-terms will always be considered up to β-conversion. The **types** of minimal higher-order logic consist of one basic type o and function types of the form $\alpha \rightarrow \beta$. The **terms** of minimal higher-order logic are those of simply typed-lambda terms - constants, variables, abstractions - with the usual type constraints. Write $a : \tau$ to mean "a is of type τ." The only constants are the constant $\Rightarrow$ of type $o \rightarrow o \rightarrow o$, and for each type τ, the constant $\forall_\tau : (\tau \rightarrow o) \rightarrow o$. A **proposition** is a term of type o.

We write $\phi \Rightarrow \psi$ for $\Rightarrow (\phi, \psi)$, and $\forall x^\alpha.\phi$ for $\forall_\alpha(\lambda x.\phi)$. The application of a to the successive arguments $b_1, \ldots, b_n$ is written $a(b_1, \ldots, b_n)$. The notation $[a/x]b$ stands for the substitution of the term a for the variable x in b.

We define inductively when a proposition ϕ is **entailed** by a finite set Γ of propositions, notation $\Gamma \vdash \phi$. A proposition is **provable** or **true** iff it is entailed by the empty set of propositions. This is given by the rules.

$$\frac{\phi \in \Gamma}{\Gamma \vdash \phi} \qquad (HYP)$$

$$\frac{\Gamma \cup \{\phi\} \vdash \psi}{\Gamma \vdash \phi \Rightarrow \psi} \qquad (ABS)$$

$$\frac{\Gamma \vdash \phi \Rightarrow \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi} \qquad (MP)$$

$$\frac{\Gamma \vdash \forall x^\alpha.\phi}{\Gamma \vdash [x/t]\phi} \qquad (INST)$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \forall x^\alpha . \phi} \qquad (GEN)$$

In the rule $(INST)$, t is a term of type α, and in the rule (GEN), it has to be assumed that x^α does not appear free in any proposition of Γ.

1.1.2 Definition of other logical connectives

It is possible to define other logical connectives. This fact was in essence already known to Russell [25], at least for negation and conjunction.

$$\begin{array}{lcll}
\bot & = & \forall \phi^o . \phi & : o \\
\neg & = & \lambda \phi^o . \phi \Rightarrow \bot & : o \to o \\
\top & = & \forall \phi^o . \phi \Rightarrow \phi & : o \\
\wedge & = & \lambda \phi^o, \psi^o . \forall \delta^o . (\phi \Rightarrow \psi \Rightarrow \delta) \Rightarrow \delta & : o \to o \to o \\
\vee & = & \lambda \phi^o, \psi^o . \forall \delta^o . (\phi \Rightarrow \delta) \Rightarrow (\psi \Rightarrow \delta) \Rightarrow \delta & : o \to o \to o \\
\exists_\tau & = & \lambda P^{\tau \to o} . \forall \delta^o . (\forall x^\tau . P(x) \Rightarrow \delta) \Rightarrow \delta & : (\tau \to o) \to o
\end{array}$$

Often, we shall not write explicitely the type of a bound variable when it can be infered. For instance, the definition of $\exists_\tau$ will also be written $\lambda P . \forall \delta . (\forall x . P(x) \Rightarrow \delta) \Rightarrow \delta : (\tau \to o) \to o$.

1.1.3 Church's Higher-Order Logic

The original logic of Church [3] was formulated for classical logic and had a ground type of individuals. Yet another difference was the introduction of a description operator (another version contains an extensionality axiom and the axiom of choice). It is possible to interpret classical higher-order propositional logic in minimal higher-order logic (see [11]).

1.1.4 Semantics

Minimal higher-order logic has a direct set-theoretic semantics. Each type denotes a finite set: the type o a set with two elements { T,F }, and the function type operator is interpreted as set-theoretic exponentiation. The constants $\Rightarrow$, $\forall_\tau$ are then interpreted following the usual truth-table laws of boolean logic. By induction, it is seen that a provable proposition gets the value T under this semantics. This insures the **consistency** of minimal higher-order logic, that is, there are propositions that are not provable. For instance, the proposition $\bot = \forall \phi^o . \phi$ gets the value F, and hence cannot get a proof.

Such a semantics is not faithful to the intuitionistic character of minimal higher-order logic. Topos theory provides various intuitionistic interpretations [13], which fail however to reflect the definitional equality on propositions.

1.2 Polymorphic Higher-Order Logic

1.2.1 Second-Order Lambda Calculus

Second-order lambda-calculus has been introduced independently by Girard [10] and Reynolds [22]. One motivation is to provide a syntax for polymorphic (or generic, or uniform) procedure. Typically, the identity operation is of type $\alpha \to \alpha$, where α is arbitrary, and such an operation behaves "uniformly" in α. It is quite difficult however to describe precisely this notion of uniformity, as it is shown by the paradox we will present.

The types of second-order calculus are either type variables, written α, $\beta, \ldots$, or function types $\sigma \to \tau$, or product types $\Pi\alpha.\tau$. For instance, the type of the polymorphic identity is $\Pi\alpha.\alpha \to \alpha$.

A **closed** type is a type without free type variables.

The syntax of the terms of second-order lambda calculus is the one of simply typed lambda-calculus, extended with **type instantiation** $a\{\tau\} : [\alpha/\tau]\sigma$, where a is a term of type $\Pi\alpha.\sigma$, and **type abstraction** $\Lambda\alpha.a : \Pi\alpha.\sigma$, where a is a term of type σ. For this rule, the type variable α should not appear free in the type of the term variables of a.

For instance, the polymorphic identity $\mathsf{id} = \Lambda\alpha.\lambda x^\alpha.x$ is a term of type $\mathsf{A} = \Pi\alpha.\alpha \to \alpha$. Notice that it is possible to intantiate id on its own type $\mathsf{id}\{\mathsf{A}\} : \mathsf{A} \to \mathsf{A}$, and to apply the result to id, getting $\mathsf{id}\{\mathsf{A}\}(\mathsf{id}) : \mathsf{A}$.

The β-conversion of typed lambda-calculus is extended with **type β-conversion**

$$(\Lambda\alpha.a)\{\tau\} = [\alpha/\tau]a : [\alpha/\tau]\sigma.$$

For instance, the term $\mathsf{id}\{\mathsf{A}\}$ is convertible to the term $\lambda x^{\mathsf{A}}.x$, and hence id is convertible to $\mathsf{id}\{\mathsf{A}\}(\mathsf{id})$.

1.2.2 A presentation of the system

We consider second-order lambda calculus with one ground type o, one constant $\Rightarrow : o \to o \to o$, and, for each closed type τ, one constant $\forall_\tau : (\tau \to o) \to o$. Terms are always considered up to conversion. As above, we write $\phi \Rightarrow \psi$ for $\Rightarrow (\phi, \psi)$, and $\forall x^\alpha.\phi$ for $\forall_\alpha(\lambda x.\phi)$. A **proposition** is a term of type o.

We define exactly as in minimal higher-order logic when a proposition is entailed from a finite set of proposition, with the inductive clauses (HYP), (ABS), (MP), $(INST)$ and (GEN), and when a proposition is provable. We get an extension of minimal higher-order logic, called **polymorphic higher-order logic**.

1.2.3 An example of a derivation

Here is a simple example that shows the expressive power of polymorphic higher-order logic. We define:

$$\begin{array}{lcll}
\mathsf{N} & = & \Pi\alpha.\alpha \to (\alpha \to \alpha) \to \alpha & \\
\mathsf{O} & = & \Lambda\alpha.\lambda x^\alpha.\lambda f^{\alpha\to\alpha}.x & : \mathsf{N} \\
\mathsf{S} & = & \lambda n^{\mathsf{N}}.\Lambda\alpha.\lambda x^\alpha.\lambda f^{\alpha\to\alpha}.f(n\{\alpha\}(x,f)) & : \mathsf{N} \to \mathsf{N} \\
\mathsf{E}_\tau & = & \lambda x^\tau, y^\tau.\forall P^{\tau\to o}.P(y) \Rightarrow P(x) & : \tau \to \tau \to o
\end{array}$$

The term E_τ is called **Leibniz's equality** over the closed type τ. The propositions expressing that E_τ is an equivalence relation over τ are directly provable [26]. Notice next that if we define

$$\mathsf{P} = \lambda n^{\mathsf{N}}.n\{o\}(\bot, \lambda\phi^o.\mathsf{T}) : \mathsf{N} \to o$$

then we have the conversion

$$\begin{array}{rcll}
\mathsf{P}(\mathsf{O}) & = & \bot & : o \\
\mathsf{P}(\mathsf{S}(x)) & = & \mathsf{T} & : o
\end{array}$$

and, from this, it follows that the proposition $\forall x^{\mathsf{N}}.\neg\mathsf{E}_{\mathsf{N}}(\mathsf{O}, \mathsf{S}(x))$ is provable. This proposition expresses the fourth Peano axiom for arithmetic.

2 A Type-Theoretic Refinement of Reynolds' Theorem

2.1 An heuristic presentation of Reynolds' Theorem

Reynolds' theorem [24, 21] states that there is no set-theoretic model of second-order lambda-calculus. We do not need here to detail the notion of "set-theoretic model" required in order to make this statement precise. But we will however give some comments in order to motivate the argument of the next section.

Since there is no set of all sets, there is a problem in interpreting set-theoretically second-order lambda-calculus. However, in [23], Reynolds conjectured that there is a non trivial set-theoretic model where the function operator is interpreted as set-theoretic exponentiation. The idea was that, in interpreting a product of a family of sets (A_X), indexed over all sets, we consider only family $a_X \in A_X$ that are "uniform", with a strong enough notion of uniformity so that the collection of uniform families is small enough to be considered as a set.

Let us motivate this conjecture by some concrete examples. For the type $\mathsf{A} = \Pi\alpha.\alpha \rightarrow \alpha$, the idea is to consider only "parametric" families (t_X), with $t_X \in X^X$. One definition of parametricity expresses the notion of "representation independence" (cf. [23]): for all sets X and Y, if $R \subseteq X \times Y$, and $R(x, y)$ then we shall have $R(t_X(x), t_Y(y))$. It is then the case that there is only one "parametric" family, which corresponds to the polymorphic identity. Indeed, given a set Y, and $y \in Y$, we can always take for X the singleton set $\{0\}$, and R the relation holding only between 0 and y. If (t_X) is parametric, we shall have $R(0, t_Y(y))$ which implies $t_Y(y) = y$. In this way, the type $\Pi\alpha.\alpha \rightarrow \alpha$ gets interpreted by a singleton.

For the type $\mathsf{N} = \Pi\alpha.\alpha \rightarrow (\alpha \rightarrow \alpha) \rightarrow \alpha$, the condition of uniformity of a family (t_X) becomes: for all sets X and Y, if $R \subseteq X \times Y$, if $R(a, b)$ and for all $x \in X$, $y \in Y$, $R(x, y)$ implies $R(f(x), g(y))$, then we shall have $R(t_X(a, f), t_Y(b, g))$. It is then the case that if (t_X) is parametric, there exists a fixed integer n_0 such that $t_X(x, f) = f^n(x)$ for all set X, $x \in X$ and $f \in X^X$. This integer n_0 is $t_\omega(0, S)$ where S is the successor function. Indeed, given a set Y, and $b \in Y$, $g \in Y^Y$, we let $R \subseteq \omega \times Y$ be the relation holding between n and $y \in Y$ only if $y = g^n(b)$. If (t_X) is parametric, we shall have $R(n_0, t_Y(b, g))$ which implies $t_Y(b, g) = g^{n_0}(y)$. In this way, N gets interpreted essentially by ω.

Such an argument is directly generalised to any type of second-order lambda-calculus determined by any algebraic signature: the type gets interpreted essentially by the initial algebra of this signature. This is shown in [23].

It is then natural to look for the case of a signature that has no set-theoretic initial algebra, and the simplest example is the signature with only one constructor that maps elements of $B^{(B^X)}$ into elements of X, where B is a fixed set. This leads to the consideration of parametric families for the type $\Pi\alpha.(((\alpha \rightarrow \tau) \rightarrow \tau) \rightarrow \alpha) \rightarrow \alpha$, where τ is a fixed type, and to Reynolds' proof in [24].

2.2 A Type-Theoretic Formulation

The intuitive arguments of the previous section cannot be formulated in polymorphic higher-order logic. Indeed, the uniformity condition involves in general a quantification over all sets, and we have no quantification over type variables in polymorphic higher-order logic. Instead, we will express it as a kind of "induction principle" over a given type.

We first consider the following expressions. Given a type expression α, we let $\Phi(\alpha)$ be $(\alpha \rightarrow o) \rightarrow o$.

$$
\begin{array}{rcll}
\mathsf{A}_0 & = & \Pi\alpha.(\Phi(\alpha)\to\alpha)\to\alpha & \\
\phi & = & \Lambda\alpha,\beta.\lambda f.\lambda z.\lambda u.z(\lambda x.u(f(x))) & :\Pi\alpha,\beta.(\alpha\to\beta)\to\Phi(\alpha)\to\Phi(\beta) \\
\mathsf{iter} & = & \Lambda\alpha.\lambda f.\lambda u.u\{\alpha\}(f) & :\Pi\alpha.(\Phi(\alpha)\to\alpha)\to\mathsf{A}_0\to\alpha \\
\mathsf{intro} & = & \lambda z.\Lambda\alpha.\lambda f.f(\phi\{\mathsf{A}_0,\alpha\}(\mathsf{iter}\{\alpha\}(f),z)) & :\Phi(\mathsf{A}_0)\to\mathsf{A}_0 \\
\mathsf{match} & = & \mathsf{iter}\{\Phi(\mathsf{A}_0)\}(\phi\{\Phi(\mathsf{A}_0),\mathsf{A}_0\}(\mathsf{intro})) & :\mathsf{A}_0\to\Phi(\mathsf{A}_0)
\end{array}
$$

All these definitions can be done in second-order lambda-calculus. The term ϕ expresses that the map $\alpha \longmapsto \Phi(\alpha)$ can be seen as a "functor", and the term iter expresses some kind of "weak initiality" of A_0 w.r.t. this "functor". This corresponds to the functor $T(X) = 2^{(2^X)}$ in set theory, and we are going to build in polymorphic higher-order logic what would be an initial T-algebra (see [21]).

If α is a type, we write $\mathsf{Rel}(\alpha)$ the type $\alpha\to\alpha\to o$. If $E:\mathsf{Rel}(\alpha)$, we say that E is a **relation** on α. Let us say that a relation is a **partial equivalence relation** iff it is provably symmetric and transitive.

If we have $f:\alpha\to\beta$, E relation on α and F relation on β, let us write $\mathsf{morphism}(E,F,f)$ the proposition $\forall x,y^{\alpha}.E(x,y)\Rightarrow F(f(x),f(y))$. We say that f is a **morphism** between E and F if, and only if, the proposition $\mathsf{morphism}(E,F,f)$ is provable. If furthermore $g:\beta\to\alpha$ is a morphism between F and E, we say that the pair (f,g) is an **isomorphism** between E and F if, and only if, both propositions $\forall x,y.E(x,y)\Rightarrow E(x,g(f(y)))$ and $\forall x,y.F(x,y)\Rightarrow F(x,f(g(y)))$ are provable.

The next definitions associate to the types o and A_0 a relation that is provably partial equivalence relation.

$$
\begin{array}{rcll}
\equiv & = & \lambda\phi,\psi.(\phi\Rightarrow\psi)\wedge(\psi\Rightarrow\phi) & :\mathsf{Rel}(o) \\
\mathsf{sym} & = & \Lambda\alpha.\lambda E.\forall x,y.E(x,y)\Rightarrow E(y,x) & :\Pi\alpha.\mathsf{Rel}(\alpha)\to o \\
\mathsf{trans} & = & \Lambda\alpha.\lambda E.\forall x,y,z.E(x,y)\Rightarrow E(y,z)\Rightarrow E(x,z) & :\Pi\alpha.\mathsf{Rel}(\alpha)\to o \\
\mathsf{per} & = & \Lambda\alpha.\lambda E.\mathsf{sym}\{\alpha\}(E)\wedge\mathsf{trans}\{\alpha\}(E) & :\Pi\alpha.\mathsf{Rel}(\alpha)\to o \\
\mathsf{power} & = & \Lambda\alpha.\lambda E.\lambda f,g.\forall x,y.E(x,y)\Rightarrow f(x)\equiv g(y) & :\Pi\alpha.\mathsf{Rel}(\alpha)\to\mathsf{Rel}(\alpha\to o) \\
\phi_2 & = & \Lambda\alpha.\lambda E.\mathsf{power}\{\alpha\to o\}(\mathsf{power}\{\alpha\}(E)) & :\Pi\alpha.\mathsf{Rel}(\alpha)\to\mathsf{Rel}(\Phi(\alpha))
\end{array}
$$

The term ϕ_2 extends the action of $\alpha \longmapsto \Phi(\alpha)$ to relations over types. The term $\mathsf{per}\{\alpha\}(E)$ represents the fact that E is a relation symmetric and transitive on the type α.

It is direct to show:

Lemma: if E is a partial equivalence on the type τ, then $\mathsf{power}\{\tau\}(E)$ is a partial equivalence relation on the type $\tau\to o$.

We introduce next a term that represents the intersection of all relations E on A_0 that are partial equivalence relations and such that intro is a morphism between $\phi_2\{\mathsf{A}_0\}(E)$ and E.

$$\mathsf{E}_0 = \lambda x,y.\forall E.\mathsf{per}\{\mathsf{A}_0\}(E)\Rightarrow\mathsf{morphism}(\phi_2\{\mathsf{A}_0\}(E),E,\mathsf{intro})\Rightarrow E(x,y) \;:\; \mathsf{Rel}(\mathsf{A}_0)$$

Since these two properties of a relation on A_0 are closed under intersection, we have:

Lemma: the relation E_0 is a partial equivalence relation on A_0, and intro is a morphism between $\phi_2\{\mathsf{A}_0\}(E_0)$ and E_0.

The relation E_0 can be seen as a construction in polymorphic higher-order logic of the initial T-algebra of the functor $T(X) = 2^{(2^X)}$.

Lemma: the term match is a morphism between $\phi_2\{\mathsf{A}_0\}(E_0)$ and E_0; furthermore $(\mathsf{intro}, \mathsf{match})$ is an isomorphism between E_0 and $\phi_2\{\mathsf{A}_0\}(E_0)$.

For this, we essentially follow the usual argument that the morphism parts of initial T-algebra are isomorphisms (see [21] and the references given there).

Theorem: Polymorphic higher-order logic is inconsistent.

That is, all propositions are provable, or alternatively, $\perp$ is provable. This follows directly from the lemmas, and the usual intuitionistic proof of Cantor's theorem, that there cannot be onto maps from a set to its power set (see for instance [19]).

This argument has been checked and found using a computer, and the formal proof is presented in [7].

2.3 Connection with Girard's paradox

In [11], Girard considers essentially the extension of polymorphic higher-order logic with quantification over type variables (called "system U") and proves that a form of Burali-Forti paradox holds for this extension. The question of the consistency of polymorphic higher-order logic (called "system U^-") is then raised and left open. The theorem above solves this question negatively.

Reynolds' argument, as it is presented in [24] can be directly formulated in the system U, but not in polymorphic higher-order logic, because the notion of parametricity used there is defined with a quantification over set variables. The idea of replacing this quantification by an "induction principle" appears also, independently, in the framework of topos theory in a paper of A. Pitts [19].

In [5], a slight simplification of Girard's argument is presented. We have not been able however to formulate a "Burali-Forti" like paradox in polymorphic higher-order logic, that is, we have not seen if it was possible to avoid the quantification over type variables used in [11, 5].

3 Application to Impredicative Type Theory

3.1 Impredicative Type Theory

Impredicative type theory has been introduced in [4] and is analysed in [6]. We will not present in detail this type theory, but limit ourselves to a short description.

Impredicative type theory is a direct expression of the principle of "propositions-as-types" and "proof-as-objects" for minimal higher-order logic. In order to stress this aspect, we represent by Set the type of propositions, that are now thought of as intuitionistic sets (the set of their proofs). The objets of type Set are themselves considered as types. We let a "small type", or "set" be a type that is also an object of type Set. The basic operation is the dependent product, written $(\Pi x : A)B(x)$ of a dependent family of types $B(x)$ $(x : A)$ over a type A. The basic feature of impredicative Type Theory is that small types are closed by product. If $B(x) : \mathsf{Set}$ $(x : A)$, then $(\Pi x : A)B(x) : \mathsf{Set}$. The theorem of Reynolds shows that it is impossible to think of the present sets as sets in the sense of Zermelo-Skolem-Fraenkel.

This basic feature is the main difference with Martin-Löf's logical framework, as presented in [18]. Otherwise, these systems are quite similar. In particular, a fundamental role is played by the notion of **context**, which is a finite set of typed variables declaration. This notion is also a basic notion of Automath, and we refer to the article [2] for an intuitive description of contexts.

If A and B are types, we let $A \rightarrow B$ be the product of the constant type family B over the type A. In the case where A and B are small types or sets, we write it also $A \Rightarrow B$. Minimal higher-order logic has a direct interpretation in impredicative type theory: o gets interpreted by Set, and a proposition gets interpreted by a small type, which represents the type of its proofs. For instance, the proposition $\mathsf{T} = \forall\phi.\phi \Rightarrow \phi$ is interpreted by $(\Pi X : \mathsf{Set})X \Rightarrow X$. The rules of inference (HYP), (ABS), (MP), $(INST)$ and (GEN) are then a consequence of the general principle that a proposition is true if, and only if, its corresponding type of proofs is inhabited. For instance, the usual proof of T is the polymorphic identity $(\lambda X : \mathsf{Set})(\lambda x : X)x$ over intuitionistic sets. We will use the same notations for logical connectives introduced in minimal higher-order logic, suitably reinterpreted in the framework of impredicative Type Theory.

The "truth table" semantics of minimal higher-order logic described above is directly extended to a model of impredicative Type Theory where a type is interpreted as a finite set, and a small type as a set that has at most one element. Let this model be the **proof irrelevance model**, so called because it forgets proof objects. This terminology is inspired by [2].

3.2 Definite descriptions and Excluded Middle

3.2.1 Proof Irrelevance

The principle of **proof irrelevance** is

$$(\Pi A : \mathsf{Set})(\Pi x, y : A)\mathsf{E}_A(x, y).$$

It states that any set (or intuitionistic proposition) has at most one element w.r.t. Leibniz's equality. Since Leibniz's equality is the weakest possible notion of equality, in the sense that if E is an equivalence relation on A, then $\mathsf{E}_A(x, y)$ implies $E(x, y)$, the principle of proof irrelevance implies that any set has at most one element w.r.t. any notion of equality over this set.

3.2.2 The principle of definite description

Let A be a set, and $\phi(x) : \mathsf{Set}\ (x : A)$. As in minimal higher-order logic, we let $(\exists x : A)\phi(x)$ be $(\Pi X : \mathsf{Set})[(\Pi x : A)[\phi(x) \Rightarrow X]] \Rightarrow X$ and we let $(\exists! x : A)\ \phi(x) : \mathsf{Set}$ be the term

$$(\exists x : A)[\phi(x) \wedge (\Pi y : A)[\phi(y) \Rightarrow \mathsf{E}_A(x, y)]].$$

This expresses that there exists one and only one element satisfying ϕ, where the equality on A is Leibniz's equality. The principle of **definite descriptions** is

$$(\Pi A, B : \mathsf{Set})(\Pi R : A \rightarrow B \rightarrow \mathsf{Set})\ [(\Pi x : A)(\exists! y : B)R(x, y)] \Rightarrow [(\exists f : A \rightarrow B)(\Pi x : A)R(x, f(x))]$$

This principle appears in the system of Church [3], in the form of a description operator ι. The motivation comes from Russell's work on denoting (see [27, 26]).

3.2.3 Excluded Middle

The last principle we shall consider is the principle of **excluded middle**.

$$(\Pi A : \mathsf{Set}) A \vee \neg(A).$$

The extension of Martin-Löf's set theory with this principle has been considered by J. Smith in [29]. It is direct to check that this principle is equivalent to

$$(\Pi A : \mathsf{Set}) \neg(\neg(A)) \Rightarrow A.$$

3.3 An application

We can now state the application of the inconsistency of polymorphic higher-order logic.

Lemma: The set

$$(\exists f : o \rightarrow o \rightarrow o)(\Pi x, y : \mathsf{B})\ T(f(x,y)) \equiv [T(x) \Rightarrow T(y)] \qquad (IMP).$$

and, for each set A, the set

$$(\exists f : (A \rightarrow \mathsf{B}) \rightarrow \mathsf{B})(\Pi P : A \rightarrow \mathsf{B})\ T(f(P)) \equiv [(\Pi x : A) T(P(x))] \qquad (UNIV_A).$$

are inhabited.

Proof: We show only how to build a proof of (IMP); the case of $(UNIV_A)$ can be solved in a similar way.

If $x : \mathsf{B}$, we let $T(x) : \mathsf{Set}$ be $E_{\mathsf{B}}(x, \mathsf{true})$, $F(x) : \mathsf{Set}$ be $E_{\mathsf{B}}(x, \mathsf{false})$, and $B(x) : \mathsf{Set}$ be $T(x) \vee F(x)$.

Notice that if we have $B(z_1)$, $B(z_2)$ and $T(z_1) \equiv T(z_2)$, then we have also $\mathsf{E}_{\mathsf{B}}(z_1, z_2)$. Indeed, the axiom $\neg(\mathsf{E}_{\mathsf{B}}(\mathsf{true}, \mathsf{false}))$ rules out the case $T(z_1)$, $F(z_2)$ and the case $F(z_1)$, $T(z_2)$. If $T(z_1)$ and $T(z_2)$, then $\mathsf{E}_{\mathsf{B}}(z_1, z_2)$, because Leibniz's equality is symmetric and transitive. Similarly, if $F(z_1)$ and $F(z_2)$, then $\mathsf{E}_{\mathsf{B}}(z_1, z_2)$.

This can be expressed in intuitive terms as the fact that the operator $T(z)$ $(z : \mathsf{B})$ is "one-to-one" on elements of B that satisfies the predicate B.

For getting a proof of (IMP), we build a proof of a stronger statement

$$(\exists f : o \rightarrow o \rightarrow o)(\Pi x, y : \mathsf{B})\ B(f(x,y)) \wedge [T(f(x,y)) \equiv [T(x) \Rightarrow T(y)]].$$

This follows from the principle of definite description and

$$(\Pi x, y : \mathsf{B})(\exists! z : \mathsf{B})\ B(z) \wedge [T(z) \equiv [T(x) \Rightarrow T(y)]].$$

This is a direct consequence of the axiom $\neg(\mathsf{E}_{\mathsf{B}}(\mathsf{true}, \mathsf{false}))$, and of the principle of excluded middle. Indeed, by the principle of excluded middle, we have $T(x) \Rightarrow T(y)$ or $\neg(T(x) \Rightarrow T(y))$. In the first case, we can choose $z = \mathsf{true}$, and in the second case $z = \mathsf{false}$. Furthermore, we have seen that the axiom $\neg(\mathsf{E}_{\mathsf{B}}(\mathsf{true}, \mathsf{false}))$ implies that the operator $T(z)$ $(z : \mathsf{B})$ is one-to-one on elements of type B that satisfies the predicate B. $\square$

Theorem: In impredicative type theory extended with excluded middle, the principle of definite description implies the principle of proof irrelevance.

Proof: We place ourselves in the context

$$\Gamma = \mathsf{B} : \mathsf{Set}, \mathsf{true} : \mathsf{B}, \mathsf{false} : \mathsf{B}, h : \neg(\mathsf{E}_{\mathsf{B}}(\mathsf{true}, \mathsf{false})),$$

and we build a proof of $\perp$ in this context.

It will then follow from the principle of excluded middle that $\mathsf{E}_{\mathsf{B}}(\mathsf{true}, \mathsf{false})$ is derivable in the context

$$\mathsf{B} : \mathsf{Set}, \mathsf{true} : \mathsf{B}, \mathsf{false} : \mathsf{B}.$$

Hence, the principle of proof irrelevance is derivable in the empty context.

First, we give a way to interpret each closed types of polymorphic higher-order logic by a set of impredicative Type Theory. We interpret the type of propositions o by the set B, and in general a type of polymorphic higher-order logic will be interpreted as a set, interpreting the function type operator as exponentiation on sets and the product over type variables as the product over set variables. For instance, the type $\Pi\alpha.\alpha \to \alpha$ is interpreted as the set $(\Pi X : \mathsf{Set})X \to X$.

Next, we consider a fixed derivation of the absurd proposition in polymorphic higher-order logic. In this derivation, we have used only a finite number of universal quantification over a finite number of closed types. Let $A_1, \ldots, A_n : \mathsf{Set}$ be an enumeration of the translation of those types in impredicative Type Theory. Consider then the context Γ extended by

$$f_0 : \mathsf{B} \to \mathsf{B} \to \mathsf{B}, h_0 : (\Pi x, y : \mathsf{B})\ T(f_0(x, y)) \equiv [T(x) \Rightarrow T(y)],$$

and for each set A_i,

$$f_i : (A_i \to \mathsf{B}) \to \mathsf{B}, h_i : (\Pi P : A_i \to \mathsf{B})\ T(f_i(P)) \equiv [(\Pi x : A_i)T(P(x))].$$

In this extended context Δ, We can translate the given proof of the absurd proposition into a construction of a term of type $\perp$. For this, we interpret $\Rightarrow$ as f_0, and each universal quantification by one of the term f_i.

By this way, we get a construction of type $\perp$ in the extended context Δ.

Using the lemma, we get a proof of (IMP), and of $(UNIV_{A_1}), \ldots, (UNIV_{A_n})$. This allows us to transform this derivation of $\perp$ in the extented context Δ into a derivation of $\perp$ in the context Γ. $\square$

4 Related Results and Problems

4.1 Looping combinators

The inconsistency of polymorphic higher-order logic, or even of the system U of [11], entails, by direct translation, the existence of a non normalisable term in a type system with a type of all types (see [16, 5, 12]). The existence of a fixed-point combinator in such a type system is an open problem since [16]. The article [12] contains a proof, using computers in an essential way, that shows the existence of a family of looping combinators, that is, a family of terms $Y_n : (X : \mathsf{Type})X \to X$ such that $Y_n(X, f) = f(Y_{n+1}(X, f))$ $(X : \mathsf{Type}, f : X \to X)$. The fact that we get a family of looping combinators, and not a fixed point combinator seems to be closely connected to the well-known "mismatch" in the representation of destructors for recursively defined types represented in second-order lambda-calculus (as presented for instance in [17]). But the author has not been able to make this connection precise.

The existence of a family of looping combinators entails the undecidability of type checking for a type system with a type of all types. In [9], the existence of a family of looping combinators is derived from A-translation in polymorphic higher-order logic.

In [8], it is shown that it is possible to build such a fixed-point operator in the presence of a the well-founded type operator of Martin-Löf [15].

4.2 Strong existence

The results about excluded middle in impredicative theory were first expressed as consequence of the inconsistency of the system U of Girard, which extends polymorphic higher-order logic with quantification over type variables [6]. It was then shown that it is possible to interpret system U in the context

$$\mathsf{B} : \mathsf{Set},\ E : \mathsf{B} \rightarrow \mathsf{Set},\ \epsilon : \mathsf{Set} \rightarrow \mathsf{B},\ H : (X : \mathsf{Set})\ X \equiv E(\epsilon(X)).$$

Hence, it is possible to derive $\perp$ in this context. The author does not know any "direct" derivation of $\perp$ in this context.

A consequence of this is the fact that, in presence of a strong existence operator [15] added to impredicative Type Theory, the principle of excluded middle implies the principle of proof irrelevance. A different proof, somewhat more direct and based in a different idea than Reynolds', has been given by S. Berardi, and checked in the proof checker LEGO of R. Pollack.

The present result, which concerns the principle of definite descriptions, generalises and was motivated by a result of G. Pottinger [20].

4.3 Consistency and Independence Results

S. Berardi has shown by a model theoretic argument that the axiom of description, and hence the axiom of choice, is not provable in impredicative Type Theory (personal communication.) A "syntactic" version of this model is described in [1]. It is similar to the proof irrelevance model, but the inhabited sets are interpreted instead by the set of all untyped lambda terms. This also models the principle of excluded middle, but not the principle of proof irrelevance. It shows that the principle of proof irrelevance is independent of excluded middle.

In [28], a purely proof theoretic argument shows the consistency of a context implying classical arithmetic, where the set of integers is interpreted as a small type.

4.4 Related results in Category Theory

The results about excluded middle seem to have some connections with the two following results in category theory. Both are described in [13].

The first one is Diaconescu's theorem, that in a topos, the axiom of choice implies the principle of excluded middle. The analysis of the proof given in [13] reveals an essential use of the extensional equality, and this result does not seem to be easily interpretable in Type Theory, based in an essential way in the fact that the equality between propositions is definitional [18].

The second one is Joyal's result, that says that any "boolean category" is trivial (see [13], page 67). In this case also, this result does not seem to be easily interpretable in Type Theory, because the equality on proofs is definitional [18]. For instance, it is not the case in general that a set $\perp \rightarrow A$ has only one element w.r.t. definitional equality, but the fact that it has only one element for the extensional equality is used in an essential way in the proof presented in [13].

Conclusion

We hope to have shown that the study of paradoxes in Type Theory is a rich topic. Quite characteristic is the use of computers in the process of checking, and analysing such paradoxes [1, 5, 12]. However, the feeling of the author is that we have only superficially yet explored this question, and it is clear that a more basic understanding of the nature of paradoxes connected to impredicativity is missing.

Acknowledgement

The inconsistency of polymorphic higher-order logic was suggested first by a discussion with Ch. Paulin-Mohring about her representation of inductive types in impredicative Type Theory (see [17]), and then by a discussion with H. Barendregt about the representation of Girard's paradox in polymorphic higher-order logic. Studying the status of excluded middle in impredicative Type Theory was suggested by a discussion with S. Yoccoz. S. Berardi and G. Pottinger provided helpful comments on a draft version of this paper. I want also to thank G. Huet, who urged me to analyse paradoxes in type theories.

References

[1] Berardi S. "Type dependence and constructive mathematics." Ph. D. thesis, Dipartimento Matematica, Università di Torino, 1990.

[2] De Bruijn N.G. "A survey of the project Automath." In: To H.B. Curry: Essays in combinatory logic, lambda calculus and formalism, ed. J.P. Seldin and J.R. Hindley, Academic Press 1980, pp. 579 - 606.

[3] Church A. "A formulation of the simple theory of types." Journal of Symbolic Logic (1940), 56–68.

[4] Coquand Th. and Huet G. "Constructions: A Higher Order Proof System for Mechanizing Mathematics." EUROCAL85, Linz, Springer-Verlag LNCS 203 (1985).

[5] Coquand Th. "An Analysis of Girard's Paradox." Proceedings of the first Logic in Computer Science, Boston, 1986, pp. 227 - 236.

[6] Coquand Th. "Metamathematical Investigations of a Calculus of Constructions." Logic and Computer Science, P. Odifreddi editor, Academic Press 1990.

[7] Coquand Th. "Reynolds' paradox, with the Type:Type axiom." in The Calculus of Constructions, Documentation and user's guide, G. Huet ed., projet FORMEL, 1989.

[8] Coquand Th. "The paradox of trees in Type Theory." to appear in B.I.T., 1991.

[9] Coquand Th. and Herbelin H. "An application of A-translation to the existence of families of looping combinators." Submitted to the Journal of Functional Programing, 1991.

[10] Girard J.Y. "Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et dans la théorie des types." Proc. 2nd. Scand. Log. Symp., North Holland, 1971.

[11] Girard J.Y. “Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieur.” These d'Etat, Paris VII (1972).

[12] Howe D.J. “The Computational Behaviour of Girard's Paradox.” Proceedings of the 2nd Logic in Computer Science 87, Ithaca, 1987.

[13] Lambek J. and Scott P.J. (1986) *Introduction to Higher Order Categorical Logic.* Cambridge University Press.

[14] Martin-Löf P. “An intuitionistic theory of types: predicative part.” Logic Colloquium, North-Holland (1975).

[15] Martin-Löf P. (1984) *Intuitionistic Type Theory.* Bibliopolis.

[16] Meyer A. R. and Reinhold M. B. “type” is not a type.” In Conference record of the thirteenth annual ACM symposium on principles of programming languages, Association for Computing Machinery, SIGACT, SIGPLAN, 1986.

[17] Ch. Paulin-Mohring. “Inductive definitions in the Calculus of Constructions.” in The Calculus of Constructions, Documentation and user's guide, G. Huet ed., projet FORMEL, 1989.

[18] Nordström B., Petersson K., Smith. J. M. (1990), *Programming in Martin-Löf Type Theory.* Oxford Science Publications, Clarendon Press, Oxford.

[19] Pitts A. “Non-trivial Power Types can't be Subtypes of Polymorphic Types.” Proceedings of the fourth Logic in Computer Science, 1989, pp. 6 - 13.

[20] Pottinger G. “Definite Descriptions and Excluded Middle in the Theory of Constructions.” Communications in the TYPES electronic forum. October 1, 1989.

[21] Reynolds J.C. and Plotkin G.D. “On Functors Expressible in the Polymorphic Typed Lambda Calculus.” Logical Foundations of Functional Programming, edited by G. Huet, Addison Wesley, 1989.

[22] Reynolds J. C. “Towards a Theory of Type Structure.” Programming Symposium, Paris. Springer Verlag LNCS **19** (1974) 408–425.

[23] Reynolds J.C. “Types, abstraction and parametric polymorphism.” In: Information Processing 83, edired by R.E.A. Mason. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1983, pp. 513 - 523.

[24] Reynolds J.C. “Polymorphism is not Set-Theoretic.” Semantics of Data Types, edited by G. Kahn, D.B. MacQueen, and G.D. Plotkin, LNCS 173, Springer-Verlag, 1984, pp. 145 - 156.

[25] Russell B. (1903) *The Principles of Mathematics.* Cambridge University Press.

[26] Russell B. and Whitehead A.N. (1912) *Principia Mathematica.* Volume 1,2,3 Cambridge University Press.

[27] Russell B. “On Denoting.” Mind, 1905. Reprinted in Robert C. Marsh, Logic and Knowledge, George Allen & Unwin, London, 1956, pp. 41-56.

[28] Seldin J.P. “Excluded Middle without Definite Description in the Theory of Constructions.” MWPLT91 Spring, Proceedings of The First Montreal Workshop on Proglramming Language Theory, M. Okada and P.J. Scot eds, 1991, pages 74-83.

[29] Smith J. “On a Nonconstructive Type Theory and Program Derivation.” The Proceedings of Conference on Logic and its Applications, Bulgaria, Plenum Press, 1986.