# Inconsistency of "Automath powersets" in impredicative type theory

Herman Geuvers

November 2013

## 1 Introduction

In this note, we discuss the way that set theory was done in Automath, e.g. by van Benthem Jutting in his formalization of Landau's "Grundlagen der Arithmetik". The approach taken in Automath is to axiomatize the power-set of a type $A : \mathsf{Prop}$ as a new type $\mathcal{P}\,A : \mathsf{Prop}$, and add constants and hypotheses to make sure that one can move between $A \to \mathsf{Prop}$ and $\mathcal{P}\,A$. In first order dependent type theory, this is a viable approach. In the setting of an impredicative type theory, however, this is not: we show that it leads to an inconsistency.

One way to show the inconsistency is to add the rule that $A \to \mathsf{Prop} : \mathsf{Prop}$ for $A : \mathsf{Prop}$. This rule should somehow enable the $\mathsf{Prop} : \mathsf{Prop}$ axiom and therefore inconsistency. We do something seemingly weaker which is to introduce a context $\Gamma_0$, consisting of axioms that ensure that there is a type $\mathcal{P}\,A : \mathsf{Prop}$ that is equivalent to $A \to \mathsf{Prop}$.

The way we do this is by showing that the axioms of $\Gamma_0$ imply a context $\Gamma_1$ that has been shown inconsistent by Coquand [2]. The context $\Gamma_1$ of Coquand states that there is a "retract" of $\mathsf{Prop}$ inside some $B : \mathsf{Prop}$. Coquand indicates how the inconsistent system $\lambda U$ can be encoded in $\Gamma_1$. In the paper, Coquand also asks whether there is a direct way of showing inconsistency of $\Gamma_1$. That is what we do here, by interpreting Hurkens' paradox inside $\Gamma_1$. The latter is done inside the Coq system.

## 2 A context for set theory

We place ourselves in the Calculus of Constructions. We use Coq syntax.

**Definition 2.1** *The context $\Gamma_0$ in Coq notation is the following. (We use "Implicit Arguments".)*

```
Variable ps     : Prop->Prop.
Variable eps    : forall S:Prop, S -> ps S -> Prop.
Variable subs   : forall S:Prop, (S->Prop) -> ps S.
Variable eps_in : forall S:Prop, forall P:S->Prop, forall x:S,
                  P x -> eps x (subs P).
Variable eps_el : forall S:Prop, forall P:S->Prop, forall x:S,
                  eps x (subs P) -> P x.
```

In addition one could assume some extensionality axiom, like

```
Variable eps_eq : forall S:Prop, forall P:S->Prop,
                  P = fun x => eps x (subs P).
```

or alternatively

```
Variable EXT : forall S:Prop, forall P Q :S->Prop,
               (forall x:S, P x -> Q x) -> P = Q.
```

The latter implies the first, but we need neither of them.

The idea is that `ps A` is the type of subsets of `A` and that `eps` is the $\epsilon$ relation and `subs` turns a predicate into a subset. The equivalence of `eps x (subs P)` and `P x` is ensured by `eps_in` and `eps_el`.

**Definition 2.2** *The context* $\Gamma_1$ *of Coquand [2] is the following, again in Coq syntax.*

```
Variable B    : Prop.
Variable IN   : Prop -> B.
Variable OUT  : B -> Prop.
Variable Hin  : forall A : Prop, A -> OUT(IN A).
Variable Hout : forall A : Prop, OUT(IN A) -> A.
```

So the type `B` is a retract of `Prop` via `IN` and `OUT`, which is ensured by `Hin` and `Hout`.

**Lemma 2.3** *the context* $\Gamma_1$ *can be interpreted in* $\Gamma_0$ *and therefore* $\Gamma_0$ *is inconsistent.*

**Proof** Use the following Coq definitions.

```
Definition T   := forall A: Prop, A -> A.
Definition t   := fun A: Prop => fun x:A => x.
Definition D   := ps T.
Definition IN  := fun A:Prop => subs(fun x:T => A).
Definition OUT := fun x:ps T => eps t x.
```

Then one can prove

```
Lemma Hin  : forall A : Prop, A -> OUT(IN A).
```

using `eps_in` and

```
Lemma Hout : forall A : Prop, OUT(IN A) -> A.
```

using `eps_el`.

We conclude that $\Gamma_0$ is inconsistent, because $\Gamma_1$ is inconsistent, as was shown by Coquand in [2] by interpreting $\lambda U$ inside it.

# 3 A direct proof of inconsistency of $\Gamma_1$

Here we give a direct proof of inconsistency of the context $\Gamma_1$ of [2]. We translate the paradox of Hurkens in it and as it is so short we give the code completely. The first 5 declarations are $\Gamma_1$. Then we construct a term of type `B` for an arbitrary `B : Prop` using the definitions and Lemmas from Hurkens paradox, with small modifications to make it all type check: in comparison to the paradox presented in [4, 3, 1], we use `D` in place of `Prop` at various places.

```
Section Paradox.

Set Implicit Arguments.

Variable D : Prop.
Variable IN : Prop -> D.
Variable OUT : D -> Prop.
Variable Hin : forall A : Prop, A -> OUT(IN A).
Variable Hout : forall A : Prop, OUT(IN A) -> A.

Variable B:Prop.
Definition V := forall A:Prop, ((A -> Prop) -> A->Prop) -> A -> Prop.
```

```
Definition U := V -> D.
Definition sb := fun (A:Prop) (r:(A-> Prop) -> A -> Prop)(a:A)(z:V) =>(r (z A r) a).
Definition le (i : U->Prop)(x:U) :=
           OUT(x (fun A r => fun a => i (fun v:V => IN(sb r a v)))).
Definition induct := fun i : U->Prop => forall x:U, le i x -> i x.
Definition WF : U := (fun z :V => IN(induct (z U le))).
Definition I (x:U) : Prop :=
           (forall i: U->Prop, le i x -> i (fun v:V => IN(sb le x v))) -> B.

Lemma Omega : forall i: U -> Prop, induct i -> i WF.
Proof.
intros i y.
apply y.
unfold le, WF, induct in |- *.
apply Hin.
intros x H0.
apply y.
unfold le.
apply Hin.
exact H0.
Qed.

Lemma lemma1 : induct (fun u => I u).
Proof.
unfold induct in |- *.
intros x p.
unfold I.
intro q.
apply (q (fun u => (I u)) p).
intros i Haux.
apply q.
apply (Hout Haux).
Qed.

Lemma lemma2 : (forall i:U -> Prop, induct i -> (i WF)) -> B.
Proof.
intro x.
apply (x (fun u => (I u)) lemma1).
intros i H0.
apply (x (fun y => i(fun v => IN(sb le y v)))).
unfold le, WF in H0.
apply (Hout H0).
Qed.


Theorem paradox : B.
Proof.
exact (lemma2 Omega).
Qed.

End Paradox.
```

# References


[1] Barthe, G. and Coquand, Th., Remarks on the equational theory of non-normalizing Pure Type Systems. *Journal of functional programming*, **16**(2), pages 137–155, 2006.

[2] Coquand, Th., A new paradox in type theory. *Logic, methodology and philosophy of science ix: Proc. ninth int. congress of logic, methodology, and philosophy of science*, pages 7–14, Elsevier, 1994.

[3] Geuvers, H., Inconsistency of classical logic in type theory, Short Note (Version of November 27, 2001)
`http://www.cs.ru.nl/~herman/PUBS/newnote.ps.gz`

[4] Hurkens A. 1995, A simplification of Girard's paradox, in Dezani-Ciancaglini, M. Plotkin, G., eds, *Second International Conference on Typed Lambda Calculi and Applications*, TLCA'95, Vol. 902 of LNCS, pp. 266–278.