Notes on inconsistency

Vladimir Voevodsky

Started August 6, 2010

Contents

These notes contain a description of a possible approach to the proof of inconsistency of set theory. The basic idea of this approach is to construct a term $A : Prop$ in Coq (actually in a "lighter" version of it based on CIC0, see [**?**]) and a proof that $A = \neg A$. Since the type system of Coq admits set-theoretic models which map $Prop$ to the disjoint union of two points and negation to the permutation of these points this will imply inconsistency of set theory.

The main idea behind the construction of A is a variant of the argument used in the famous Goedel's proof of incompleteness theorem (see []). We will write $\equiv$ for the definitional equality between terms, i.e. the equality based on β, ι etc. reductions and $=$ for the propositional equality. One proceeds as follows:

1. One constructs a function $enum : nat- > (nat- > Prop)$ such that for any term X of type $nat- > Prop$, or more precisely, for any valid judgement of the form $(\vdash X : nat- > Prop)$, one can, using the concrete syntax of X, find a natural number n_X such that $enum\, n_X$ is definitionally equal to X.

2. Consider the term $(fun\, n : nat \; => \; \neg(enum\, n\, n))$. It is of type $nat- > Prop$, therefore there exists $m : nat$ such that $enum\, m \equiv (fun\, n : nat \; => \; \neg(enum\, n\, n))$. Then

$$enum\, m\, m \equiv (fun\, n : nat \; => \; \neg(enum\, n\, n))\, m \equiv \neg(enum\, m\, m)$$

3. Since definitional equality implies propositional one we get $enum\, m\, m = \neg(enum\, m\, m)$.

It is intuitively obvious that $enum$ with properties given above can be constructed since there are only countably many strings of symbols used in Coq and since any checker for Coq contains in particular an algorithm which will verify whether or not a given string of symbols followed by $: nat- > Prop$ is a valid judgement of the form $(\vdash X : nat- > Prop)$.

Formalization of this intuitive idea however takes quite a bit of work. In part it is related to the fact that the only known proof of the termination of the checker algorithm uses consistency of set theory as an essential assumption. While it is clear that one may assume consistency in a proof of inconsistency this may create some technical complications.