On an approach to conveniently formalize mathematics

Vladimir Voevodsky

Started June 4, 2005

As was mentioned above our goal is to find a way to encode mathematical knowledge in a format which can be used as an input by proof verification programs and by programs aimed at the creation of structured databases of mathematics. This format should be be intuitive enough such that human readers can easily verify that definitions and theorems are encoded correctly. On the other hand it should be accessible for formal proof verification. This later condition means that we should be able to develop certification programs which would take "papers" written in this format as an input and return as an output either "accept" or "not accept" in such a way that the following two conditions are satisfied.

1. The program never "accepts" a paper which contains an assertion which is recognized by the mathematical community as being false.
2. In most cases when the mathematical community believes that it knows proofs of all the assertions in a paper it can provide the certification program with enough supplementary material ("details") to make it to "accept" the paper.

Clearly the first of these conditions can never be fully guaranteed and the goal should be not to provide such a guarantee but to create programs which would satisfy this condition in practice. There may be some unorthodox ways of creating such programs but at the moment the only approach that I can see looks as follows. One chooses a formal system (language and axioms) which is generally believed to be consistent. Then, one develops the certification program keeping in mind that if it accepts a paper then it should be possible to translate all the assertions of the paper into formulas provable in the system. If this certification program accepts an assertion which is generally believed to be false then implementing such a translation we should be able to show that this outcome is due to the inconsistency of the underlying formal system and not to some software glitch. If the foundational system is well chosen this in itself would be an interesting result.

Note that the users of the format need not know anything about the formal system which is used by a certification program. In fact, different certification programs can use different formal systems as the basis of their certification algorithms. It might be possible to consider the pair consisting of the format and a certification program as some kind of a formal system in its own right. However, this does not seem to be a very productive approach since in order to satisfy the "readability" requirement it will probably have to be hopelessly complex from the logical point of view.

Here we are going to suggest a draft version of such a format. Our description will proceed as follows. First we are going to describe the intuition behind our approach to "formating" mathematics. Since it is crucial that one should be able to format the existing mathematical definitions and statements with relative ease this informal part of the description is very important. Then, instead of giving a formal syntactic description of the format, we present drafts of several shorts papers written in this format. These papers contain formatted definitions and theorems from several simple areas of

mathematics. Finally we will discuss how to construct a certification program for our format. To support our approach to the certification we will show that the inconsistency of our certification process would imply inconsistency of the ZFG-theory – the Zermelo-Fraenkel "set" theory with the added axiom stating the existence of a Grothendieck universe.

The forest base Here we introduce

1. **Class Def.** $set := \bar{S}(element)$
2. **Class Def.** $set\_correspondence := (source : set, target : set, graph : \bar{S}(element, element) | \forall u1, u2 : element\ ((u1, u2) \in graph) \Rightarrow (u1 \in source) \wedge (u2 \in target))$
3. **Class Def.** $set\_map := (f : set\_correspondence | \forall x : element(x \in f.source) \Rightarrow (\exists y : element((x, y) \in f.graph)) \wedge (\forall y1, y2 : element((x, y1) \in f.graph) \wedge ((x, y2) \in f.graph) \Rightarrow (y1 = y2)))$
4. **Class Def.** $set\_surjection := (f : set\_map | \forall y : element(y \in f.target) \Rightarrow (\exists x : element((x, y) \in f.graph)))$
5. **Class Def.** $set\_injection := (f : set\_map | \forall x1, x2, y : element((x1, y) \in f.graph) \wedge ((x2, y) \in f.graph) \Rightarrow (x1 = x2))$
6. **Class Def.** $set\_bijection := (f : set\_map | (f\ is\ set\_surjection) \wedge (f\ is\ set\_injection))$
7. **Class Def.** $finite\_set := (X : set | \forall f : set\_injection(f.source = X) \wedge (f.target = X) \Rightarrow (f\ is\ set\_bijection)$
8. **Class Def.** $infinite\_set := (X : set | \neg(X\ is\ finite\_set))$
9. **Constr.Def.** $set\_union := (X, Y : set; Z : set | \forall x : element(x \in Z) \Leftrightarrow (x \in X) \vee (x \in Y))$
10. **Constr.Def.** $singleton\_set := (x : element; X : set | \forall y : element(y \in X) \Leftrightarrow (y = x))$
11. **Class Def.** $singleton\_set := (X : set | \exists x : element(X = singleton\_set(x)))$
12. **Const.Def.** $empty\_set := (.; X : set | \forall x : element \neg(x \in X))$ (???)
13. **Class Def.** $set\_and\_subset := (X, Y : set | \forall x : element(x \in X) \Rightarrow (x \in Y))$
14. **Notation:** we write $X \subset Y$ instead of $(X, Y)\ is\ set\_and\_subset$
15. **Notation:** we write $f : X \to Y$ instead of $(f.source = X) \wedge (f.target = Y)$
16. **Notation:** we write $X \cup Y$ instead of $set\_union(X, Y)$
17. **Class Dec.** $small\_set : \bar{S}(set)$ (notation???)
18. **Constr. Dec.** $\nu : small\_set \to element$
19. **Constr.Def.** $set\_of\_subsets := (X : small\_set; Y : set | \forall x : element(x \in Y) \Leftrightarrow (\exists A : set(A \subset X) \wedge (x = \nu(A))))$

20. **Ax.** $\forall X, Y : small\_set(\nu(X) = \nu(Y)) \Rightarrow (X = Y)$

21. **Ax.** $\forall X, Y : set(X\ is\ small\_set) \wedge (Y\ is\ small\_set) \Rightarrow (set\_union(X, Y)\ is\ small\_set)$

22. **Ax.** $\forall x : element(sigleton\_set(x)\ is\ small\_set)$

23. **Ax.** $empty\_set\ is\ small\_set$

24. **Ax.** $\forall X : small\_set(set\_of\_subsets(X)\ is\ small\_set)$