To next papers[1]

Vladimir Voevodsky[2,3]

Started July 2014

Abstract

???

Conjecture [**2009.12.27.prop1**] Let $\mathcal{C}$ be a category, CC be a C-system and $M : CC \to \mathcal{C}$ a functor such that $M(pt_{CC})$ is a final object of $\mathcal{C}$ and M maps distinguished squares of CC to pull-back squares of $\mathcal{C}$. Then there exists a universe $p_M : \widetilde{U}_M \to U_M$ in $PreShv(\mathcal{C})$ and a C-system homomorphism $M' : CC \to CC(PreShv(\mathcal{C}), p_M)$ such that the square

$$\begin{tikzcd}
CC \arrow[r, "M"] \arrow[d, "M'"] & \mathcal{C} \arrow[d] \\
CC(PreShv(\mathcal{C}), p_M) \arrow[r, "int"] & PreShv(C)
\end{tikzcd}$$

where the right hand side vertical arrow is the Yoneda embedding, commutes up to a functor isomorphism.

Idea: use $U_M = M^*(Ob_1)$ and $\widetilde{U}_M = M^*(\widetilde{Ob}_1)$.

??? "Contextual category" was thought by Cartmell (cf. [**?**]) as a generalization of a category, not as category with structure. Similar to generalized algebraic theory versus algebraic theory. ???

??? But that means that we do not have a notion of contextual category yet since we do not have a nation of *equivalence* of contextual categories. ???

??? In a type theory every bound variable and every "boundable" variable must have a type. In one approach this might be already seen on the level of the syntactic signature - an operation which binds a variable must have one of its arguments distinguished as the type of the variable. On the other hand the variables on which a type depends need not be boundable. ???

Note: important thing for the development of type theory is to be able, for $E \in M(X)$ to say whether E is the result of application of a given operation O and if yes, then what is the argument of this operation. This "match" ability is present (? in what precise sense) in the description through nominal signatures and initial algebras of nominal polynomial functors and absent in the description using LF.

Note: about GAT and how, based on the result of Cartmell, one can see that any C-system can be obtained through the construction of Proposition ... applied to the pair of the form (M, M) where M is defined by a single sorted algebraic signature.

??? *** Question: can we give an example of a C-system which is finitely presented using a nominal signature but can only be infinitely presented as a GAT. *** ???

[1] *2000 Mathematical Subject Classification: 03B15, 03B22, 03F50, 03G25*

[2] School of Mathematics, Institute for Advanced Study, Princeton NJ, USA. e-mail: vladimir@ias.edu

[3] Work on this paper was supported by NSF grant 1100938.

(Aug. 15, 2014) We want to produce a definition of a type system with which we will be able to state and prove a theorem stating that the associated C-system is an initial model of the theory obtained by adding to the theory of C-systems additional operations and equations whose format is directly defined by the format of the derivation rules which define the type system.

We choose to consider only the case of type systems built from expressions over a nominal signature with one name-sort Var, two distinct data-sorts $Term$ and $Type$ and an operation $v : Var \to Term$.

Find an example of a constructor with an argument with several bound variables.

(Aug. 17, 2014) Let us say that "typed sorts" are formed from the base sorts:

$TermT$ - "term of type"

$Type$ - "type"

Type' := Type — (var:Type').Type' term' := term — (term', term') — term':Type' — (var:Type').term'

(var:Type).((term:Type),Type)

app((f:T),(o:T'),(x:T")(T"'))

—- f:T —- o:T' x:T" —- T"' type

—- T = Prop((x:T")(T"')) —- T' = T"