Generalised type setups for dependently sorted logic

Peter Aczel
Schools of Mathematics and Computer Science
The University of Manchester
petera@cs.man.ac.uk

June 10, 2011

1 Introduction

Dependently sorted logic is intended to be a generalisation of many sorted logic that allows sorts to depend on variables in the kind of way that types can depend on variables in dependent type theories. The notion of a *generalised Type Setup (gTS)* is a generalisation of the notion of a *type setup*. The latter notion was previously introduced by me, in about 2004, as a suitable abstract notion of type theory for specifying the sorts and terms dependent on contexts of variable declarations, needed to formulate a dependently sorted logic. The aim of my paper will be to review the notion of a type setup and introduce the more general notion.

2 Some earlier examples of notions of dependently sorted logics

A seminal example of a logic having dependent sorts is the logic for equational reasoning in Cartmell's *Generalised Algebraic (GA) theories*, [Cartmell, 1978, Cartmell, 1986]. A typical example of a GA theory is the axiom system for the notion of a category which has a sort Obj of objects and a sort constructor Hom for forming a sort $Hom(x, y)$ of maps $x \to y$, for $x, y : Obj$ and has function symbols, i,e, term constructors, id and $comp$ for forming identity terms $id(x)$, for $x : Obj$ and composition maps $comp(x, y, z, u, v) : Hom(x, z)$ for $x, y, z : Obj$, $u : Hom(x, y)$, $v : Hom(y, z)$. The notion of a GA theory, being purely equational, does not capture the full generality of the idea of a dependently sorted logic, which allows first order formulae which are not equations. Nevertheless the formulation of the general notion of a signature for the language of a GA theory and the definition of the syntactic categories of contexts of variable declarations, sorts and terms is already rather complicated.

Another significant example of a logic having dependent sorts is Makkai's *first order logic with dependent sorts (FOLDS)*,[Makkai, 1995]. While $FOLDS$ has the usual logical constants for first order logic it avoids having individual constants or function symbols and so can get away with a simple notion of sort because the only terms are the variables so that each sort must either be a constant sort or else must have the form $s(x_1, \ldots, x_n)$ where s is an n-place sort constructor, for $n > 0$, and $x_1, \ldots, x_n$ is a list of variables.

The notion of a *Logic-enriched type theory*, [Aczel and Gambino, 2002], [Gambino and Aczel, 2006], provides another kind of example of a notion of logic with dependent sorts, the sorts being the dependent types of a type theory. Here the sorts and terms are expressions of the type theory, the type theory itself having a rather complicated structure. In order to give a precise definition of the idea of a logic-enrichment of a type theory it is necessary to have a precise definition of the notion of a type theory. In [Gambino and Aczel, 2006] a concrete syntactic notion of *standard pure type theory* was presented that had certain forms of judgement and allowed an arbitrary system of rules for deriving judgements.

3 Type Setups

It has seemed desirable to formulate a more general notion of dependently sorted logic which would include the above examples. What seems to be needed is a more general, more abstract, notion of *dependent type theory*. Already, in [Cartmell, 1978, Cartmell, 1986], Cartmell introduced the notions of *category with attributes and contextual category*, claiming that, for his GA theories, contextual categories are the algebraic structures

> *'that structurally correspond exactly to the syntactically defined theories'*

After Cartmell's work category theorists have come up with many variations on the notions of category with attributes and contextual category; e.g. category with display maps, [Taylor, 1986], comprehension category, [Jacobs, 1991], category with families, [Dybjer, 1996]. These are intended to capture at various levels of abstraction the syntax and/or semantics of dependent type theories.

But none of these notions have seemed to me to be quite right as an abstract notion of 'type theory' suitable for a sufficiently general formulation of the notion of a dependently sorted logic over a 'type theory' that would give a smooth generalisation of the usual way first order logic is presented. I have introduced the notion of a *type setup* for that purpose, see [Aczel, 2009], and the notion has been investigated by Joao Belo, [Belo, 2007, Belo, 2009].

In contrast to the above earlier notions, type setups make use of an explicit notion of variable with contexts as finite sequences of variable declarations, $x : A$, and substitutions as finite sequences of variable assignments, $x := a$. From a purely category theoretic point of view the explicit use of variables is not essential. Nevertheless, even in our abstract setting we prefer to keep to the

logically familiar use of variables so as to have a smooth generalisation of the traditional presentation of many-sorted logic.

The contexts, Γ, of a type setup are the objects of a category $\mathcal{C}$ whose maps $\gamma : \Gamma' \to \Gamma$, play the role of substitutions. Associated with each context Γ is a set of Γ-types and, for each Γ-type A there is a set of Γ-terms of type A. The substitutions $\gamma : \Gamma' \to \Gamma$ act, functorially on types and terms so that, for each Γ-type A, $A[\gamma]$ is a Γ'-type and $a[\gamma]$ is a Γ'-term of type $A[\gamma]$ for each Γ-term a of type A. As with the earlier category theoretic notions for dependent types, a fundamental ingredient of the notion of a type setup are axioms for the 'comprehension extension' of a context. In a type setup a given a Γ-type A can be extended by adding a new variable declaration $x : A$ to obtain the context $(\Gamma, x : A)$, provided that the variable x is Γ-free; i.e. has not been already declared in Γ. In addition, given a substitution $\gamma : \Delta \to \Gamma$ and a Γ-term a of type A, the substitution γ can be extended to a substitution $(\gamma, x := a) : \Delta \to (\Gamma, x : A)$ such that $x[(\gamma, x := a)] = a$.

Given a type setup and a signature of sorted predicate symbols we have all the ingredients needed to formulate the syntax of formulae of a dependently sorted logic, with a notion of Γ-formula inductively generated from the atomic Γ-formulae using the usual connectives and quantifiers, the quantifiers having the forms $(\forall x : A)$, $(\exists x : A)$ where A is a Γ-type and the variable x is Γ-free. The action of the substitutions of the type setup on formulae can be defined by structural recursion on formulae in the usal way and a natural deduction style axiomatisation of intuitionistic logic using sequents can be formulated.

4 The notion of a Generalised Type Setup (gTS)

Concrete dependent type theories generally use untyped variables, but use contexts which are finite sequences of variable declarations, that associate a type with each declared variable. So the contexts form a tree structure of finite sequences, with the empty context at the root and each context having as its children its extensions by adding a new variable declaration. The notion of a contextual category also has such a tree structure. Other notions of category for type dependency, such as the notions of category with attributes and category with families, are more aimed at the semantics of type dependency and do not impose the tree structure.

As with concrete dependent type theories the notion of a type setup also uses a tree structure of finite sequences of variable declarations. I now think that this extra structure, which only complicates the presentation, is unnecessary for the purpose of the formulation of dependently sorted logic and the new notion of generalised type setup avoids the extra structure. Of course the notion of a gTS keeps the fundamental notion of context extension, as do all the various competing notions.

In my talk I hope to explain the notion of a gTS, describe intuitionistic predicate logic with equality over a gTS and, if there is time, outline an application of a logic-enriched type theory.

References

[Aczel, 2009] P. Aczel, *Predicate Logic over a Type Setup*, slides in `http://www.cs.man.ac.uk/ petera/papers.html`, of a talk given at the meeting "Philosophy and Foundations of Mathematics: Epistemological and Ontological Aspects", a conference dedicated to Per Martin-Lof on the occasion of his retirement, Uppsala, May 2009.

[Aczel and Gambino, 2002] P. Aczel and N. Gambino, *Collection Principles in Dependent Type Theory*, **Types for Proofs and Programs** (P. Callaghan, Z. Luo,, J. McKinna, and R. Pollack, editors), Lecture Notes in Computer Science, vol 2277, Springer, pp. 1-23, 2002.

[Belo, 2007] J. Belo, *Dependently Sorted Logic*, **TYPES'07**, Proceedings of the 2007 international conference on Types for proofs and programs, Springer-Verlag Berlin, Heidelberg, 2008

[Belo, 2009] J. Belo, Ph.D. thesis, Manchester University, 2009.

[Cartmell, 1978] J. Cartmell, *Generalised Algebraic theories and Contextual Categories*, Ph.D. thesis, Univ. Oxford, 1978.

[Cartmell, 1986] J. Cartmell, *Generalised Algebraic theories and Contextual Categories*, Annals of Pure and Applied Logic, 32:209-243, 1986.

[Dybjer, 1996] P. Dybjer, *Internal Type Theory*, **Types for Proofs and Programs**, (S. Berardi and M. Coppo, editors), Lecture Notes in Computer Science, vol 1158, Springer, pp. 120-134, 1996.

[Gambino and Aczel, 2006] N. Gambino and P. Aczel, *The Generalised Type-Theoretic Interpretation of Constructive Set Theory*, Journal of Symbolic Logic, vol.71, pp. 67-103, 2006.

[Jacobs, 1991] B. Jacobs, *Categorical Type Theory*, Ph.D. thesis, Nijmegen, 1991.

[Makkai, 1995] M. Makkai, *First Order Logic with Dependent Sorts, with Applications to Category Theory*, preprint, McGill University, 1995.

[Taylor, 1986] P. Taylor, *Recursive domains, indexed category theory and polymorphism*, Ph.D. thesis, Cambridge University, 1986