Abstract. We present the paradigm of *categories-as-syntax*. We briefly recall the even stronger paradigm *categories-as-machine-language* which led from $\lambda$-calculus to categorical combinators viewed as basic instructions of the Categorical Abstract Machine. We extend the categorical combinators so as to describe the proof theory of first order logic and higher order logic. We do not prove new results: the use of indexed categories and the description of quantifiers as adjoints goes back to Lawvere and has been developed in detail in works of R. Seely. We rather propose a syntactic, equational presentation of those ideas. We sketch the (quasi-equational) categorical structures for dependent types, following ideas of J. Cartmell (contextual categories). All these theories of categorical combinators, together with the translations from $\lambda$-calculi into them, are introduced smoothly, thanks to the systematic use of
- an abstract variable-free notation for $\lambda$-calculus, going back to N. De Bruijn,
- a sequent formulation of the natural deduction.

Introduction

We present the paradigm of *categories-as-syntax*. We first briefly review the even stronger paradigm of *categories-as-machine-language*, which led from $\lambda$-calculus to categorical combinators viewed as basic instructions of the Categorical Abstract Machine (CAM), thus offering a new approach to the implementation of functional languages based on $\lambda$-calculus (like ML or Miranda); $\lambda$-expressions are translated into categorical combinators, each of which is interpreted as a very simple machine language instruction in the CAM. The categorical combinators of cartesian closed categories and the CAM have been described at length elsewhere ([11, 14, 12]) so we shall limit the presentation here to the essence of the approach.

We extend the categorical combinators so as to describe the proof theory of (intuitionistic) first order logic and higher order logic. We do not prove new results: the use of indexed categories and the description of quantifiers as adjoints goes back to Lawvere (hyperdoctrines) [28, 29] and has been developed in detail in works of R. Seely [35, 37]. We rather propose a syntactic presentation of those ideas, which enhances an equational presentation of proof theory.

We end the paper by sketching the (quasi-equational) categorical structures for dependent types, using ideas of J. Cartmell [6] (contextual categories). Actually Cartmell’s structures were rediscovered by T. Ehrhard and the author [15]. Subsequently, Ehrhard [16, 17] has worked on a more abstract framework (D-categories), based on fibered categories rather than (exactly) indexed categories (i.e. split fibrations). This more general framework encompasses
another approach to the categorical semantics of type dependency, due to Seely, and based on locally cartesian closed categories [36]. Ehrhard also enriches his categorical structures in a very simple way to capture (a version of) Coquand-Huet's calculus of constructions [7]. Here we shall take an intermediate step, by sketching Ehrhard's structures in the special case of split fibrations.

All these theories of categorical combinators are introduced smoothly from \(\lambda\)-calculus or natural deduction style, thanks to the systematic use of

- an abstract variable-free notation, going back to N. De Bruijn,
- a sequent formulation of the natural deduction.

Finally we shall define very simple realizability models, based on the notion of \(\omega\)-set (due to E. Moggi).

The paper does not require a previous knowledge of the literature on categorical logic, but supposes a basic acquaintance with \(\lambda\)-calculus and categories (not including fibrations).

In Section 1 we discuss what we mean by abstract notation. Section 2 is a rapid description of the CAM. Section 3 introduces the categorical combinators of cartesian closed categories. Section 4 defends the idea that abstract notation is not good only at evaluating, but is also helpful when typing and giving meaning. We attack on quantifiers from Section 5. Sections 5,6 deal with predicate logic, more specifically Section 5 with universal, section 6 with existential quantification. Section 7 discusses the extension to higher-order logic. Section 8 is a discussion of type dependency (including a quick "flash" on the theory of constructions). Section 9 discusses how to turn Moggi's \(\omega\)-sets into (higher order) D-categories and hyperdoctrines.

1. What is \(\alpha\)-conversion about?

Let us look at the following session in your favorite functional language:

\[
\begin{align*}
\text{I:} & \quad \text{let } x = 3 \\
& \quad \text{let } y = 4 \\
& \quad x + y.
\end{align*}
\]

Basic observation: What is important is not that the variables have such or such name, but an indication on how to fetch their value (or their type). Thus the latter session is more akin to

\[
\begin{align*}
\text{II:} & \quad \text{let } x = 3 \\
& \quad \text{let } z = 4 \\
& \quad x + z
\end{align*}
\]

\[
\begin{align*}
\text{III:} & \quad \text{let } y = 4 \\
& \quad \text{let } x = 3 \\
& \quad x + y.
\end{align*}
\]

A way of turning this "more akin" into a rigorous certitude is to have an abstract notation on which to map all the sessions above, and to consider as "akin" the sessions mapped onto the same abstract one. Here is the abstract