Using Category Theory to Design Programming Languages

John C. Reynolds*

Computer Science Department, Carnegie Mellon University,
Pittsburgh, U.S.A.
john.reynolds@cs.cmu.edu

In a 1980 paper entitled "Using Category Theory to Design Conversions and Generic Operators", the author showed how the concepts of category theory can guide the design of a programming language to avoid anomalies in the interaction of implicit conversions and generic operators. He wrote:

> ... Our intention is not to use any deep theorems of category theory, but merely to employ the basic concepts of this field as organizing principles. This might appear as a desire to be concise at the expense of being esoteric. But in designing a programming language, the central problem is to organize a variety of concepts in a way which exhibits uniformity and generality. Substantial leverage can be gained in attacking this problem if these concepts can be defined concisely within a framework which has already proven its ability to impose uniformity and generality upon a wide variety of mathematics.

In this talk, we will revisit these ideas and generalize them to other aspects of language design. We intend to demonstrate that language design is an unusual form of applied mathematics, where one uses, rather than theorems, the underlying structural principles of a field such as category theory.

- We will review the treatment of implicit conversions and generic operators, in which the conversions are specified by a functor on the preorder of subtypes, and the operators are natural transformations.
- We will describe the treatment of side-effects as monads, where each side-effect-free type is mapped into the corresponding type with effects by a monadic functor.
- We will present the description of block structure by functor categories, which insures that the values of local variables do not escape from the block in which the variables are declared.

* Research supported in part by National Science Foundation Grant CCF-0541021.

G. Castagna (Ed.): ESOP 2009, LNCS 5502, pp. 62–63, 2009.
© Springer-Verlag Berlin Heidelberg 2009

– If time permits, we will describe the treatment of type variables and polymorphism by PL-categories, where type expressions are described by a base category, and ordinary expressions by a functor from the base category to a category of (cartesian closed) categories.

In each case, we will show how desirable properties of a programming language can be enforced by using an appropriate categorial definition.

No prior knowledge of category theory will be assumed.