Algebraic Specification and Symbolic Computation[*]

Laureano Lambán, Vico Pascual and Julio Rubio

Universidad de La Rioja, Luis de Ulloa s/n
Edificio Vives, 26004 Logroño (La Rioja), España
{lalamban,mvico,jurubio}@dmc.unirioja.es

This work deals with a rather unusual application of a Computer Algebra system. Looking for an algebraic specification of the symbolic computation system called EAT (for *Effective Algebraic Topology*), we have found several interesting relations in the algebraic specification field. In this way, the "application arrow" has been, in a sense, reversed, and an application of Computer Algebra to Algebraic Specification has been obtained.

The system under observation is a Common Lisp program, designed by Sergeraert and called EAT [3]. The aim of EAT is to provide a tool for the mechanized computation of homology groups of infinite topological spaces (namely, iterated loop spaces). In order to undertake a formal analysis of the system, an operation on Abstract Data Types was introduced in [2]. The main tool for this construction is a syntactic operation between signatures, which was denoted by $()_{imp}$. The operation $()_{imp}$ intends to capture the way of working in EAT, and, in particular, to model the handling of infinite data structures (which is essential to implement algorithms in Algebraic Topology).

In this work, interpretations of the $()_{imp}$ operation are presented from four points of view:

- The original one, the programming perspective.
- As a particular case of certain Category Theory constructions.
- In the hidden algebraic specification setting [1].
- From a coalgebraic point of view [4].

[*]Partially supported by DGES, project PB98-1621-C02-01.

Then these four perspectives are suitably related. As a consequence of it, a generalization of the results presented in [2] is obtained. Namely, the final object of [2] is now expressed as a coproduct. This result gives a more accurate model of the EAT way of programming.

As a by-product, some previously known results in the algebraic specification field are re-found, enlightening the relationship between the hidden and the coalgebraic methodologies. In particular, the final objects described, for instance, in [1] and [4] are easily expressed in our case, illustrating the very nature of these algebraic specification constructs.

References

[1] J. Goguen, G. Malcolm. A hidden agenda. *Theoretical Computer Science,* **245** (2000) 55–101.

[2] L. Lambán, V. Pascual, J. Rubio. Specifying implementations. In *Proceedings ISSAC'99,* ACM Press (1999) 245–251.

[3] J. Rubio, F. Sergeraert, Y. Siret. *EAT: Symbolic software for effective homology computation.* `ftp://fourier.ujf-grenoble.fr/pub/EAT`, Institut Fourier, Grenoble, 1997.

[4] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science,* **249** (2000) 3–80.