Algebraic and Coalgebraic Methods in Software Development

MAP-i — 2013-14

Summary

This document describes a proposal for a course on Fundations of Computing *to be offered in the 2013-14 MAP-i edition. The proposal is supported by a team from Aveiro University (Dep. of Mathematics) and Minho University (Dep. of Informatics).*

1 Context and Objectives

An increasing number of computer based systems rely on the cooperation of distributed, heterogeneous components or services organised into open software architectures that, moreover, can survive in loosely-coupled environments and be easily adapted to changing application requirements. Such is the case, for example, of applications designed to take advantage of the increased computational power provided by massively parallel systems or of the whole business of Internet-based software development. In order to develop such systems in a systematic way, the focus of development methods has switched, along the last decade, from functional to architectural issues: both data and processes are encapsulated into software units which are connected into large systems resorting to a number of techniques intended to support reusability and modifiability. This encapsulation principle is essential to both the *object-oriented* and the more recent *component-based* software engineering paradigms.

This entails the need for semantic techniques able to cope either with *date structuring* and *prescription of functionality*, as well as with specification and analysis of (externally observable) *behaviour*. Both *qualitative* and *quantitative*, i.e. QoS-related, aspects o software have to be taken into consideration.

If on data-intensive applications the main element to tackle is the *structure* of information and its transformations, in dynamic, reactive computing the focus is placed on system's behaviours and their interactions. Quoting Robin Milner, in his Turing Award Lecture, computing science has become a *structural theory of interaction*: *Thus software, from being a prescription for how to do something — in Turing's terms a "list of instructions" — becomes much more akin to a description of behaviour, not only programmed on a computer, but occurring by hap or design inside or outside it.*

Both *initial* algebras and *final* coalgebras provide abstract descriptions of a variety of phenomena in programming, in particular of *data* and *behavioural* structures, respectively. As universal properties, they both entail definitional and proof principles, *i.e.*, a basis for the development of program calculi directly based on (actually driven by) type specifications. Moreover, such properties can be turned into programming *combinators* and used, not only to calculate programs, but also to program with. In functional programming the role of such universals has been fundamental to a whole discipline of algorithm derivation and transformation. On the oher hand, *coalgebraic modelling* of dynamical systems and reasoning by *coinduction* has recently emerged as active area of research.

This course explores the role of such algebraic and coalgebraic structures, and corresponding logics, in program development. As expected, *initial* algebras turn out to be *inductive data types*, *i.e.*, abstract descriptions of data structures. Dually, *final* coalgebras entail a notion of *coinductive*, *behaviour types*, representing the dynamics of systems. Therefore, the course will cover the core ideas, techniques and results in

- Algebraic specification, induction and equational logic
- Coalgebraic specification, coinduction and modal logic for coalgebras

In both cases exposition will resort to suitable tool support (namely, OBJ, BOBJ and Circus).

In software development, relations nicely capture nondeterminism and vagueness of requirements. Probabilistic functions go a step further by quantifying the likelihood of each possible, expected or faulty, behaviour. Such a shift from *qualitative* to *quantitative reasoning*, which is becoming pervasive in Computer Science, calls for a language able to accommodate both these aspects, while preserving the polymorphic and calculational style of functional and relational program derivation. Recently, *typed linear algebra* has been suggested as a suitable candidate for such a unifying role, when restricting to discrete probability spaces. A new module in the course, first offered in the 2013-14 edition, will explore this perspective and its applications, namely to the study of *fault-propagation* in software systems. It builds on very recent and exciting results on

- Linear algebra of programming and applications

To provide a common background to formulate and discuss the topics above, the course will also include a brief

- Introduction to category theory

The course will build a *roadmap* to the broad area of algebraic and coalgebraic methods in software development, not only by providing an introductory survey, but also by exposing students to cutting-edge research topics and open problems, object of three research projects currently coordinated by the proponent team:

- Mondrian (`PTDC/EIA-CCO/108302/2008`) on *Foundations for architectural design: Service certification, dynamic reconfiguration and self-adaptability*,
- Qais (`PTDC/EIA-CCO/122240/2010`) on *Quantitative analysis of interacting systems: foundations and algorithms*,
- Nasoni (`PTDC/EEI-CTP/2341/2012`) on *Heterogenous software coordination: Foundations, methods, tools.*

Motivated students will have the opportunity to discuss possible PhD topics in Foundations of Computing within the themes of this course. Funding opportunities may be available.

2 Learning outcomes

- Familiarity with the main topics, research questions and scientific challenges in the covered area (algebraic and coalgebraic methods);
- Ability to apply them to building and reasoning about, abstract models for software, its functionality, behaviour and composition.
- Ability to extract information from scientific papers in the area.
- Enhanced technical writing and presentation skills.

3 Pre-requisites

The course is almost self-contained, assuming only familiarity with elementary discrete mathematics at undergraduate level. Some previous experience on semantics of programming languages will help.

4 Format

Tutorial module, supported with demos and experimental lab work.

5 Grading

Assessment on base of an individual report on a research paper and a set of written exercises.

6 Course Contents

Plan

1. Introduction to category theory for computer science
    (a) Universal properties; categories; isomorphism; monomorphisms and epimorphisms.
    (b) Constructions in categories: duality, products, sums, limits and colimits.
    (c) Functors and natural transformations
    (d) The Yoneda lemma
    (e) Adjoint functors and adjoint functor theorems
    (f) Cartesian closed categories and λ-calculus
    (g) Monoidal categories
2. Algebras and algebraic specification
    (a) Signatures, models
    (b) Equational logic
    (c) Signature morphisms
    (d) Refinements
    (e) Introduction to the theory of institutions
    (f) Behavioural specifications
3. Coalgebras and coalgebraic specification
    (a) Coalgebras
    (b) Bisimulation
    (c) Coinduction, final coalgebras
    (d) Logics for coalgebras
    (e) Applications
4. Linear algebra of programming
    (a) Categories of matrices.
    (b) Probabilities and probabilistic functions.
    (c) Adjunction: probabilistic functions represented by column stochastic matrices. The ‘matrix-transform’.
    (d) Elementary probability theory encoded in linear algebra. Examples.
    (e) The distribution monad.
    (f) Applied linear algebra of programming: calculating fault propagation in mutually recursive functions.
    (g) Weighted automata and weighted coalgebraic systems.
    (h) Weighted bisimulation. Probabilistic behaviour.
    (i) LAoP for QoS calculation.

Textbooks and Reading Material

On category theory : [3, 34, 17, 2] [12, 11, 8, 33]

On algebraic specification : [12, 11, 8, 33]

On coalgebraic modelling and coinduction : [32, 15, 16, 1]

On linear algebra of programming : [28, 18, 29]

7 Team

Luis Soares Barbosa (Coordinator) is Associate Professor, with tenure, at the Department of Informatics of Minho University, and a researcher at Cctc (area of *Theory and Formal Methods*). His research interests are related to program semantics and calculi applied to systems understanding and rigorous software construction. A particular application area concerns the development of formal models and calculi for software components, services and architectures. On this topic he has published over the past 4 years more than 15 papers in several journals and conferences. He has supervised 2 PhD thesis (1 in the area of the current proposal) and is currently supervising 5 PhD projects (2 in the area of the current proposal). *Selected relevant publications* on coalgebraic modelling and coinductive reasoning: [4, 5, 25, 31, 6, 26, 7, 23, 22].

Dirk Hofmann is Assistant Professor at Department of Mathematics at the University of Aveiro, and researcher at the Center for Research and Development in Mathematics and Applications. His main interests of research focus on the development and application of categorical methods in Mathematics, more specifically in algebra, topology and domain theory. On this topic he has published more than ten papers in several journals over the past 4 years. He has supervised 4 Msc thesis and is currently supervising 1 PhD project. *Selected relevant publications* on category theory: [14, 9, 10, 13].

Manuel António Martins is Assistant Professor at the Department of Mathematics of Aveiro University, and a researcher at the Center for Research and Development in Mathematics and Applications. His research interests are related to Abstract Algebraic Logic (AAL) and Algebraic Specification of abstract data types; namely on the application of tools and results of AAL to the specification and verification of software systems. On this topic he has published 6 papers in international journals. He has supervised 4 MSc thesis (2 in the area of the current proposal) and is currently supervising 2 PhD projects (1 in the area of the current proposal). *Selected relevant publications* on specification and verification of software systems: [19, 24, 20, 21, 23, 22].

José N. Oliveira graduated in electrical engineering in 1978 from the University of Porto in Portugal and received the MSc and PhD degrees in computer science in 1980 and 1984, respectively, from the University of Manchester, United Kingdom. Since 2010 he has been an associate professor with habilitation at the Computer Science Department of the University of Minho, Portugal, and a member of the High Assurance Software Laboratory (HASLab) of INESC TEC/University of Minho. He is also a member of IFIP WG2.1 (Algorithmic Languages and Calculi) and of the Formal Methods Europe (FME) association. He has been working on formal methods since his PhD, with a recent interest in quantitative formal techniques relying on linear algebra and category theory. In the last 3 years (2011-2013) he published 6 journal papers and supervised 4 PhD theses on these topics. *Selected relevant publications*: [30, 28, 27, 18, 6, 29]

References

[1] J. Adamek. An introduction to coalgebra. *Theory and Applications of Categories*, 14(8):157–199, 2005.

[2] J. Adamek, H. Herrlich, and G. E. Strecker. *Abstract and Concrete Categories.* John Wiley & Sons, Inc (revised electronic edition in 2004), 1990.

[3] S. Awodey. *Category Theory.* Oxford Logic Guides. Oxford University Press, 2006.

[4] L. S. Barbosa. Process calculi *à la* Bird-Meertens. In *CMCS'01*, volume 44.4, pages 47–66, Genova, April 2001. Elect. Notes in Theor. Comp. Sci., Elsevier.

[5] L. S. Barbosa and J. N. Oliveira. Coinductive interpreters for process calculi. In *Proc. of FLOPS'02*, pages 183–197. Springer Lect. Notes Comp. Sci. (2441), 2002.

[6] L. S. Barbosa and J. N. Oliveira. Transposing partial components: an exercise on coalgebraic refinement. *Theor. Comp. Sci.*, 365(1-2):2–22, 2006.

[7] L. S. Barbosa, J. N. Oliveira, and A. M. Silva. Calculating invariants as coreflexive bisimulations. In J. Meseguer and G. Rosu, editors, *Algebraic Methodology and Software Technology, 12th International Conference, AMAST 2008, Urbana, IL, USA, July 28-31, 2008, Proceedings*, pages 83–99. Springer Lect. Notes Comp. Sci. (5140), 2008.

[8] M. Bidoit and R. Hennicker. Proving behavioral refinements of col-specifications. In *Essays Dedicated to Joseph A. Goguen*, pages 333–354, 2006.

[9] Maria Manuel Clementino and Dirk Hofmann. Lawvere completeness in Topology. *Appl. Categ. Structures*, 17:175–210, 2009.

[10] Maria Manuel Clementino and Dirk Hofmann. Relative injectivity as cocompleteness for a class of distributors. *Theory Appl. Categ.*, 21(12):210–230, 2009.

[11] J. Goguen and R. Burstall. Institutions: abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.

[12] J. Goguen and G. Malcolm. *Algebraic semantics of imperative programs.* MIT Press Series in the Foundations of Computing. Cambridge, 1996.

[13] Dirk Hofmann. Topological theories and closed objects. *Adv. Math.*, 215(2):789–824, 2007.

[14] Dirk Hofmann. Injective spaces via adjunction. *J. Pure Appl. Algebra*, 2010 (accepted).

[15] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–159, 1997.

[16] A. Kurz. Coalgebras and modal logic. Technical report, Lecture Notes for ESSLLII'2001, Helsinki, 2001.

[17] S. Mac Lane. *Categories for the Working Mathematician.* Springer Verlag, 1971.

[18] H.D. Macedo and J.N. Oliveira. Typing linear algebra: A biproduct-oriented approach. *Science of Computer Programming*, 2012. In press. DOI:10.1016/j.scico.2012.07.012.

[19] M. A. Martins. Behavioral institutions and refinements in generalized hidden logics. *Journal of Universal Computer Science*, 12(8):1020–1049, 2006.

[20] M. A. Martins. Closure properties for the class of behavioral models. *Theor. Comput. Sci.*, 379(1-2):53–83, 2007.

[21] M. A. Martins. On the behavioral equivalence between k-data structures. *Comp. J.*, 50(3):181–191, 2008.

[22] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement by interpretation. In Dang Van Hung and Padmanabhan Krishnan, editors, *7th IEEE International Conference on Software Engineering and Formal Methods (SEFM'09)*, pages 250–259. IEEE Computer Society Press, 2009.

[23] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement by interpretation in a general setting. In J. Derrick E. Boiten and S. Reeves, editors, *Proc. Refinement Workshop 2009, Electr. Notes Theor. Comput. Sci. (256)*, pages 105–121. Elsevier, 2009.

[24] M. A. Martins and D. Pigozzi. Behavioural reasoning for conditional equations. *Math. Struct. Comput. Sci.*, 17(5):1075–1113, 2007.

[25] Sun Meng and L. S. Barbosa. Components as coalgebras: The refinement dimension. *Theor. Comp. Sci.*, 351:276–294, 2005.

[26] Sun Meng and L. S. Barbosa. A coalgebraic semantic framework for reasoning about UML sequence diagrams. In Hong Zhu, editor, *Proceedings of the Eighth International Conference on Quality of Software, QSIC 2008, 12-13 August 2008, Oxford, UK*, pages 17–26. IEEE Computer Society, 2008.

[27] S.-C. Mu and J.N. Oliveira. Programming from Galois connections. *Journal of Log. Algebraic Programming*, 81(6):680–704, 2012.

[28] J.N. Oliveira. Towards a linear algebra of programming. *Formal Aspects of Computing*, 24(4-6):433–458, 2012.

[29] J.N. Oliveira. Weighted automata as coalgebras in categories of matrices. *International Journal of Foundations of Computer Science*, 2013. Accepted for publication.

[30] J.N. Oliveira and M.A. Ferreira. Alloy meets the algebra of programming: A case study. *IEEE Transactions on Software Engineering*, 39(3):305–326, 2013.

[31] P. Ribeiro, M. A. Barbosa, and L. S. Barbosa. Generic process algebra: A programming challenge. *Journal of Universal Computer Science*, 12(7):922–937, 2006.

[32] J. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. (Revised version of CWI Techn. Rep. CS-R9652, 1996).

[33] D. Sannella and A. Tarlecki. *Foundations of Algebraic Specifications and Formal Program Development.* Cambridge University Press, 2011.

[34] R. F. C. Walters. *Categories and Computer Science*, volume 28 of *Cambridge Computer Science Texts*. Cambridge University Press, 1991.