PhD thesis proposal
”Pointfree Program Calculation - Theory and Aplications”

Depto. Informática, Universidad Nacional de San Luis

1 Introduction

There is clearly a need for more reliable methods of program construction than the traditional *ad hoc* methods in use today. What is needed is a science of programming, instead of today's craft (or perhaps "black art"). As Jeremy Gunawardena points out [6], computation is inherently more mathematical than most engineering artifacts; hence, practising software engineers should be at least as familiar with the mathematical foundations of software engineering as other engineers are with the foundations of their own branches of engineering.

By "mathematical foundations", we are referring to simple properties and laws of computer programs: equivalences between programming constructs, relationships between well-known algorithms, and so on. In particular, we are interested in calculating with programs, in the same way that we calculate with numeric quantities in algebra at school.

One particularly appropriate framework for program calculation is functional programming, simply because the absence of side-effects ensures referential transparency — all that matters of any expression is the value it denotes, not any other characteristic such as the method by which it computed, the time taken to evaluate it, the number of characters used to express it, and so on.

As is well-known, functions are special cases of (binary) relations which, expressed in pointfree notation, are likewise amenable to calculation. The advantage is that the range of mathematics which is involved is far more reaching, ranging from pure functions (which can be regarded as *implementations*) to their specification and the statement (and proof) of properties about them, which is traditionally carried out in (pointwise) predicate logic.

This project aims at developing and extending the well-known technique of encoding predicate logic and set-theory in the binary relation calculus, for generic program calculation purposes. This algebrization of logic has a long history since De Morgan pioneered the approach in the 1860s. More recently, the works of the MPC school [1], stemming from Eindhoven, Oxford [2] and other research places have shown the approach to be applicable to a wide range of computing problems, including dynamic programming and temporal logic. This includes, as a special case, the rich sub-calculus of (inductive) functions which has become known as the "Bird-Meertens" formalism.

In this thesis, such a pointfree calculation theory will be challenged via its application to practical problems as, for instance, the existing theory of relational databases,

which we expect to ”refactor” and (hopefully) generalize (read: add “genericity” to) in the pointfree style.

2 Context and Details

The context of this work is the relational theory of datatypes (rather than a calculus of total functions) which can be framed into allegory theory [2], a generalization of category theory.

When software designers refer to the relational calculus, by default what is understood is the calculus of n-ary relations studied in logics and database theory, and not the calculus of binary relations which was initiated by De Morgan in the 1980s [13] and eventually became the core of the algebra of programming [3], [2], [1].

According to [7], it was Quine, in this 1932 Ph.D dissertation, who showed how to develop the theory of n-ary relations for all n simultaneously, defining ordered n-tuples in terms of the ordered pair. (Norbert Wiener is apparently the first mathematician to publicly identify, in the 10910s, n-ary relations with subsets of n-tuples.) Since the 1970s, the information system community is indebted to Cood for his pioneering work on the foundations of the relational data model theory [4].

Codd discovered and publicized procedures for constructing a set of simple n-ary relations which can support a set of given data and constructed an extension of the calculus of binary relations capable of handling most typical data retrieval problems. Since then, relational database theory has been thorougly studied, and several textbooks are availables on the topic, namely [8], [12] and [5].

This thesis intends to show that the use of the pointfree binary relation calculus is beneficial in several respects. First, the fact that pointfree notation abstracts from ”points” or variables makes the reasoning more compact and effective. Second, proofs are performed by easy-to-follow calculation. Third, one is able to generalize the original theory.

3 Background

Work on pointfree-transforming and calculating relational database theory has already started. Some experiments have also been made in the area of operation and data refinement.

For instance in [10], abstract data-modeling is based on a calculus of data combinators which exhibit universal properties. These properties help in reasoning, transforming and calculating about them. Data-model calculation is shown to encompass not only data refinement but also data transformation, data-mining and data-migration.

Based on set-theory (and a modest use of category theory), reference [11] presents a constructive approach to relational database normalization theory. A set of laws which prevent from the violation of normal forms caused by partial dependencies, transitive dependencies and multi-valued dependencies are presented.

In [9] the theory of functional dependencies, wich is central to relational databases design techniques, is addressed in a pointfree style instead of reasoning in the standard set-teoretic model ”á la Codd”.

4 Activities

- ***First Year:*** Literature and information search. Development of models in the domain of relational database theory as it is formalized in [8] and [4].
- ***Second Year:*** Pointfree program calculation. Algebraic and Co-algebraic reification of developped applications.
- ***Third Year:*** Generalization of developped applications. Summary of theoric results obtained. Writing up.

5 Courses to be taken:

This doctoral program includes to take courses in the areas of: pointfree functional programming, algebra of programming, generic programming, category theory, calculus of binary relations, relational calculus.

References

1. R.C. Backhouse. Mathematics of program construction, 2004. Univ. of Nottingham, 2004. Draft of book in preparation. 608 pages.
2. R. Bird and O. de Moor. *Algebra of Programming*. Series in Computer Science. Prentice-Hall International, 1997. C. A. R. Hoare, series editor.
3. Paul Hoogendijk Ed Voerman Chritiene Aarts, Roland Backhouse and Jaap van der Woude. A relational theory of datatypes, 1992. Available from www.cs.nott.ac.uk/ rbc/papers.
4. E. F. Codd. A relational model of data for large shared data banks. *CACM*, 13(6):377–387, June 1970.
5. Hector García-Molina, Jeffrey D. Ullman, and Jennifer D. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2002. ISBN 0-13-031995-3.
6. Jeremy Gunawardena. Towards an applied mathematics for computer science, 1997. In M.S.Alber, B.Hu, and J.J. Rosenthal, editors, Current and Future Directions in Applied Mathematics. Birkhäuser, Boston.
7. Akihiro Kanamori. The empty set, the singleton, and the ordered pair. *The Bulletin of Symbolic Logic*, 9(3):273–298, 2003.
8. D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983. ISBN 0-914894-42-0.
9. J. N. Oliveira. First steps in pointfree functional dependency theory. Technical report, University of Minho, 2005.
10. J.N. Oliveira. Data processing by calculation, 2001. Lectures at the *6th Estonian Winter School in Computer Science* March 2001, Palmse, Estonia.
11. C. J. Rodrigues and J. N. Oliveira. *Normalization is Data Reification.* Technical Report UMDITR9702, University of Minho, Dec. 1997.
12. J. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.
13. V.Pratt. Origins of the calculus of binary relations. In *In Proc. Of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 248–254. IEEE Computer Society, 1992.