Domain-Specific Languages of Mathematics

Date: 2014-04-29

Applicants: *Cezar Ionescu* and *Patrik Jansson.*

Supported by: *Miroslaw Staron* (VPref.Gru@CSE), *Roger Johansson* (PA D), *Johan Jonasson* (PA TM), *Ana Bove* (PA DV@GU).

We propose to create a new bachelor-level course with the immediate aim of improving the mathematical education of computer scientists and the computer science education of mathematicians. We believe the course can be the starting point for far-reaching changes, leading to a restructuring of the mathematical training especially for engineers, but perhaps also for mathematicians themselves.

Computer science, viewed as a mathematical discipline, has certain features that set it apart from mainstream mathematics. It places much more emphasis on syntax, tends to prefer formal proofs to informal ones, and views logic as a tool rather than (just) as an object of study. It has long been advocated, both by mathematicians [Wells, 1995, Kraft, 2004] and computer scientists [Gries and Schneider, 1995, Boute, 2009], that the computer science perspective could be valuable in general mathematical education. Until today, this has been convincingly demonstrated (at least since the classical textbook of Gries and Schneider [1993]) only in the field of discrete mathematics. In fact, this demonstration has been so successful, that we increasingly see the discrete mathematics courses being taken over by computer science departments. This is a quite unsatisfactory state of affairs, for at least two reasons.

First, any benefits of the computer science perspective remain within the computer science department and the synergy with the wider mathematical landscape is lost. The mathematics department also misses the opportunity to see more in computer science than just a provider of tools for numerical computations. Considering the increasing dependence of mathematics on software, this can be a considerable loss.

Second, computer science (and other) students are exposed to two quite different approaches to teaching mathematics. For many of them, the formal, tool-oriented style of the discrete mathematics course is easier to follow than the traditional mathematical style. Since, moreover, discrete mathematics tends to be immediately useful to them, this makes the added difficulty of continuous mathematics even less palatable. As a result, their mathematical competence tends to suffer in areas such as real and complex analysis, or linear algebra.

This is a serious problem, because this lack of competence tends to infect the design of the entire curriculum. For example, a course in "Modeling of sustainable energy systems" for Chalmers' CSE[1] students has to be tailored around this limitation, meaning that the models, methods, and tools that can be presented need to be drastically simplified, to the point where contact with mainstream research becomes impossible.

We propose that a focus on *domain-specific languages* (DSLs) can be used to repair this unsatisfactory state of affairs. In computer science, a DSL "is a computer language specialized to a particular application domain" (Wikipedia), and building DSLs is increasingly becoming a standard industry practice. Empirical studies show that DSLs lead to fundamental increases in productivity, above alternative modelling approaches such as UML [Tolvanen, 2011]. Moreover, building DSLs also offers the opportunity for interdisciplinary activity and can assist in reaching a shared understanding of intuitive or vague notions (see, for example, the work done at Chalmers in cooperation with the Potsdam Institute for Climate Impact Research in the context of Global Systems Science).

Thus, a course on designing and implementing DSLs can be an important addition to an engineering curriculum. Our key idea is to combine this with a rich source of domains and applications: mathematics. Indeed, mathematics offers countless examples of DSLs: the language of group theory, say, or the language of probability theory, embedded in that of measure theory. The idea that the various branches of mathematics are in fact DSLs embedded in the "general purpose language" of set theory was (even if not expressed in these words) the driving idea of the Bourbaki project, which exerted an enormous influence on present day mathematics.

A course on *DSLs of Mathematics (DSLM)* allows us to present classical mathematical topics in a way which builds on the experience of discrete mathematics: giving specifications of the concepts introduced, paying attention to syntax and types, and so on. For the mathematics students, used to a more informal style, the increased formality is justified by the need to implement (fragments of) the language. We will provide a wide range of applications of the DSLs introduced, so that the new concepts can be seen "in action" as soon as possible.

The course will have two major learning outcomes. First, the students should be able to design and implement a DSL in a new domain. Second, they should be able to handle new mathematical areas using the computer science perspective.

To achieve these objective, the course will consists of a sequence of case studies in which a mathematical area is first presented (for example, a fragment of linear algebra, probability theory, interval analysis, or differential equations), followed by a careful analysis that will reveal the domain elements needed to build a language for that domain. The DSL will first be used informally, in order to ensure that it is sufficient to account for intended applications (for example, solving equations, or specifying a certain kind of mathematical object). It is in this step that the computer science perspective will prove valuable for improving the students' understanding of the mathematical area. The DSL will then be implemented in Haskell. The resulting implementation

[1]CSE = Computer Science & Engineering = Datateknik = D

will be compared with existing ones, such as Matlab in the case of linear algebra, or R in the case of statistical computations. Finally, limitations of the DSL will be assessed and the possibility for further improvements will be discussed.

In a first instance, the course will be an elective course for the second year within a program such as SE, CSE, or Math. The potential students will have all taken first-year mathematics course, and the only prerequisite which some of them will not satisfy will be familiarity with functional programming. However, as the current data structures course (common to the Math and CSE programs) shows, math students are usually able to catch up fairly quickly, and in any case we aim to keep to a restricted subset of Haskell (no "advanced" features will be required).

To assess the impact in terms of increased quality of education, we propose to measure how well the students do in ulterior courses that require mathematical competence (in the case of engineering students) or software compentence (in the case of math students). For example, for CS and CSE students we will measure the percentage of students who, having taken DSLM, pass the third-year courses *Transforms, signals and systems* and *Control Theory (Reglerteknik)*, which are current major stumbling blocks. For math students, we will measure their performance in ulterior scientific computing courses.

Since the course will, at least initially, be an elective one, we will also have the possibility of comparing the results with those of a control group (students who have not taken the course).

A tentative work plan is as follows:

- summer and autumn 2014: in interaction with our colleagues from the various study programs, we will perform an assessment of the current status of potential students for the course in terms of their training (what prerequisites we can reasonably assume) and future path (what mathematical fields they are likely to encounter in later studies), and we will start work on a course plan (which we plan to submit in February 2015, so that the first instance of the course can start in autumn 2015). At the same time, we will make a systematic survey of similar courses being offered at other universities and contact their teachers for an exchange of views.

- spring 2015: we aim to organize several seminars around some of the mathematical areas to be presented within DSLM, with the participation of volunteer students. We will use the feedback from these seminars to develop course materials for use within the first instance.

- autumn 2015: we will run the first instance of DSLM (partly paid by the regular course budget, partly by this project)

- spring 2016 and 2017: we will use the feedback from students following the standard Chalmers evaluation in order to improve and further develop the course material. We will also involve other faculty from CSE and mathematics in the development of other mathematics courses (prel. Linear Algebra, Analysis) with the aim to incorporate these ideas also there.

- autumn 2016 and 2017: run next instance of DSLM course (now completely self-funded). Assist in updated versions of (prel.) Linear Algebra and Analysis.

Throughout all the periods, a major concern will be to work together with our colleagues in the mathematics department in order to distill the essential principles that can be "back-ported" to the other mathematics courses, such as Mathematical Analysis or Linear Algebra. Ideally, the mathematical areas used in DSLM will become increasingly challenging, the more the effective aspects of the computer science perspective are adopted in the first-year mathematics courses.

The budget required for 2014 is 200 kSEK or approximately two person-months (shared between the applicants). For 2015 until 2017 we ask for a total of 800 kSEK where the majority of the time will be spent on extending the work to improve the quality also of the current mathematics courses. This will ensure a broad impact on improved education quality at Chalmers, not only for an initial 20–50 students/year on the DSLM course, but also for up to ten times as many in the other courses.

References

R. Boute. The decibel done right: a matter of engineering the math. *Antennas and Propagation Magazine, IEEE*, 51(6):177–184, 2009.

D. Gries and F. B. Schneider. *A logical approach to discrete math.* Springer, 1993.

D. Gries and F. B. Schneider. Teaching math more effectively, through calculational proofs. *American Mathematical Monthly*, pages 691–697, 1995.

R. Kraft. Functions and parameterizations as objects to think with. In *Maple Summer Workshop, July 2004, Wilfrid Laurier University, Waterloo, Ontario, Canada*, 2004.

J. Tolvanen. Industrial experiences on using DSLs in embedded software development. In *Proceedings of Embedded Software Engineering Kongress (Tagungsband), December 2011*, 2011.

C. Wells. Communicating mathematics: Useful ideas from computer science. *American Mathematical Monthly*, pages 397–408, 1995.