Mechanising mathematics

Between them, **Professors William Farmer** and **Jacques Carette** have years of experience in building theorem proving and symbolic computation systems. Now, they are focusing their skills on developing a revolutionary mechanised mathematics system

To begin, could you outline the central objectives of the MathScheme project? What do you hope to achieve?

WF: Our principal objective is to revolutionise how people learn and practice mathematics by developing an integrated system of computer-based tools for a wide range of mathematical activities, that are accessible to a wide range of practitioners. Unlike many similar projects, our principal non-objective is to change how people learn and practice mathematics. In other words, we want to provide computerised support for all aspects of the process of doing mathematics, but without imposing a preferred workflow onto practitioners. We want to greatly increase the efficiency of users of mathematics.

There is a powerful synergy between computation and deduction. How have computer algebra systems and theorem proving systems helped to break this synergy?

JC: Most computations (like solving the roots of a polynomial or computing an integral) rely on established algorithms – but those algorithms have preconditions on their applicability. Thus, to use these algorithms, one must first prove that the situation at hand satisfies those preconditions. Similarly, certain (difficult) proofs could be made significantly simpler if certain computations (such as factoring polynomials over exotic number fields) could just 'be done'. And of course, in larger applications of mathematics, there tends to be a continual back-and-forth between computation and deduction. Remarkably, today, there is no single system that allows this. Current systems, which are extremely good at what they do, focus quite singularly on one half of this (important!) combination of computation and deduction.

How will algorithmic mathematics and axiomatic mathematics be integrated into a single system? Could you outline a typical biform theory?

WF: Biform theories are actually quite simple: they just happen to contain the declarative aspects of a mathematical theory (its main axioms, definitions, concepts, etc.), as well as the algorithms that make such a theory effective. Instead of keeping the definition of 'integration' and the main properties of integrals completely separate from the computation of integrals (numerically or in closed-form), we put this information together. The reason this has not been done before is that these two aspects turn out to live in very different conceptual frameworks, which are frequently presented in incompatible ways – something that undergraduate mathematics professors tend to scrupulously hide from their students. Quite a lot of work is needed to have theories of mathematics where proofs and computations co-exist peacefully.

What kinds of software systems are going to be created to support the application of mathematics? What methodologies will be used in this development?

JC: We are experimenting with a range of options. The scale of a single, new and all-encompassing system is too large for our team to take on. Thus, we are building a small, core system with a novel method of communication with other systems ('trustable communication') to enable our system to understand and trust results computed by other software.

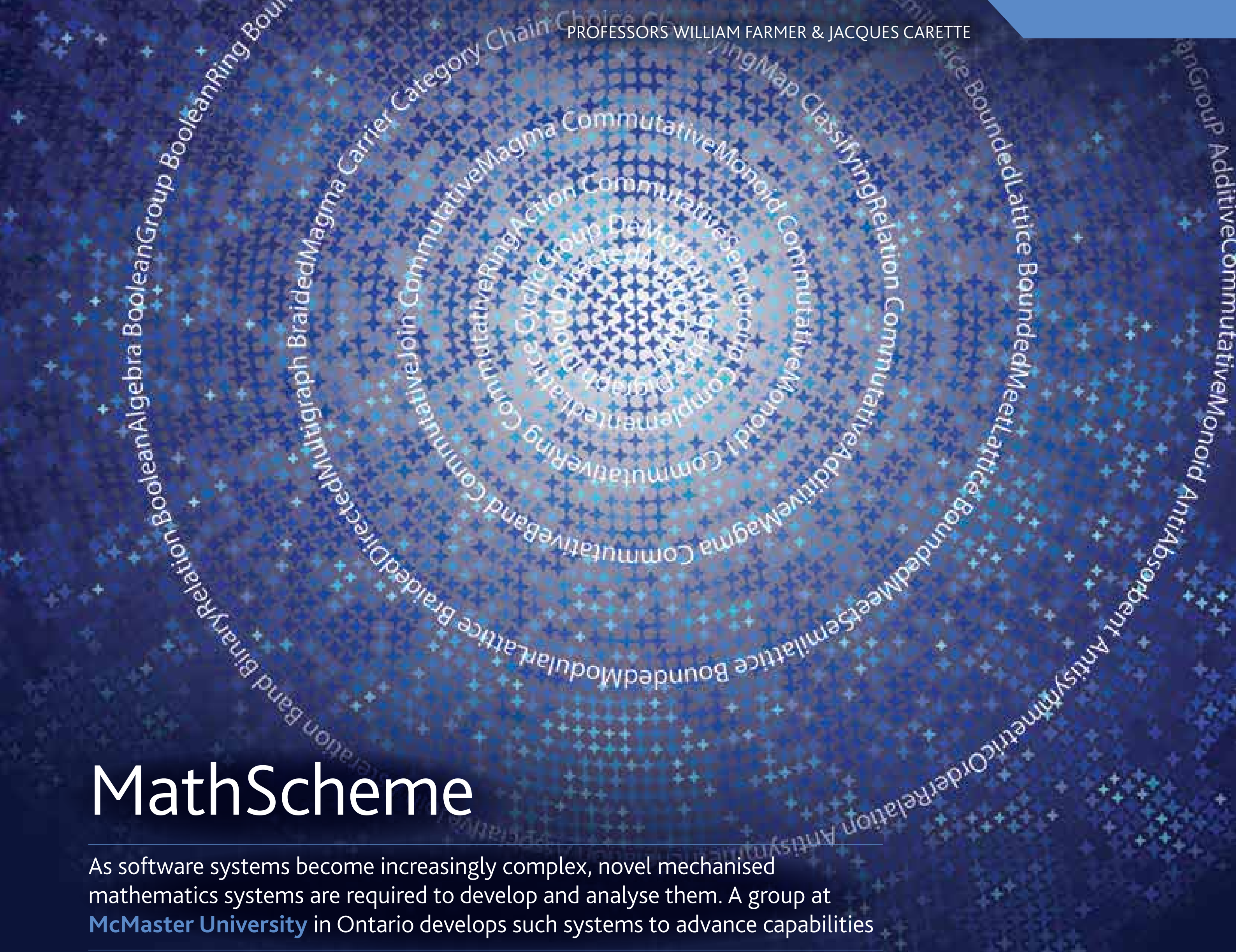
We have developed a series of conceptual and software products which represent our approach. Without going into technical details, we have developed Chiron, syntax frameworks, biform theories, the MathScheme language and its library, realms and trustable communication. We have also used, as well as invented, a lot of techniques in metaprogramming (programmes which manipulate programmes) throughout our framework.

Are there any further applications of the software system that will be developed? Where will it be used in the academic and commercial environments?

WF: We are working closely with many of our McMaster colleagues on software certification, a more flexible notion than pure software verification. We see fruitful applications of MathScheme to the production of correct-by-construction software. Of course, we are also very interested in applications to mathematics education. Eventually, we see ourselves broadening to more areas of application in mathematics.

Finally, what has been your greatest achievement on the project thus far? Have you made any breakthroughs?

JC: Probably our greatest achievement is our continued agreement: we come from rather different backgrounds and yet have worked together for 10 years already towards a common goal, never wavering. We do not think in terms of breakthroughs; our aim is to revolutionise a domain by giving it entirely new tools (think of how the advent of power tools in replacement of hand tools changed the construction industry). We are progressing, doggedly and purposefully, towards our objective. We re-use all the good ideas that we can, and we invent 'in anger' when we must.



MathScheme

As software systems become increasingly complex, novel mechanised mathematics systems are required to develop and analyse them. A group at **McMaster University** in Ontario develops such systems to advance capabilities

SOUND MATHEMATICS IS essential in all modern technological developments. As increasingly complex systems are built, mathematical reasoning becomes more difficult and prone to errors – especially within software systems. Mechanised mathematics systems (MMSs) are software systems that support the mathematics process by providing tools for performing computation, proof and other kinds of mathematical reasoning. They have the potential to revolutionise the design, implementation and analysis of complex, sophisticated software systems.

Presently two major types of MMSs exist: computer algebra systems and computer theorem proving systems. Computer algebra systems provide algorithms for symbolic computation and, whilst being relatively fast and easy to use, they are not rigorous, trustworthy nor broad in scope. Computer theorem proving systems provide tools for creating formal proofs. Although they are based on well-defined logical foundations and can support a wide range of mathematics, they are difficult to use and often lack the specific knowledge needed to perform many routine computations. This artificial division between the two types of systems has broken the synergy that working mathematicians leverage between deduction and computation. There needs to be a renewed effort to reunite these interactions within a single framework if significant advances to mechanised mathematics are to be made.

INTEGRATING DEDUCTION AND COMPUTATION

MathScheme is a long-term project taking place at McMaster University in Ontario with the aim of producing such a framework. Led by Professors Jacques Carette and William Farmer, the MathScheme team is developing tools, techniques and eventually a new system, in which formal deduction and symbolic computation are tightly integrated. Whilst previous attempts to conjoin such systems have produced disappointing results, Farmer explains that their approach is different: "We are convinced that it can only be done in a new system which is aimed at supporting the full range of mathematical activities, while leveraging all that has been learned until now regarding each of the separate tasks". Part of MathScheme's approach is to create a core system with 'trustable communication' capabilities, allowing it to understand and trust results computed by other software, thereby exploiting existing libraries of mechanised mathematics.

However, the MathScheme approach does more than just integrate axiomatic and algorithmic mathematics. "We also leverage the structure inherent in mathematical knowledge; that mathematics itself has a rich mathematical structure is well-known, but this has not been seriously used as part of the architecture of any mechanised mathematics system," Carette enthuses. Furthermore, the team understands that the users of their new system will range from advanced developers to pure end-users. System developers normally make compromises, hoping for a one-size-fits-all solution, while the MathScheme developers have planned from the start to have a multi-faceted system that is geared towards various usage scenarios.

Therefore, MathScheme's first goal is to develop a formal framework that allows mathematical knowledge to be represented both declaratively using axioms and procedurally using algorithms, and it will provide a style of mathematical reasoning in which computation and deduction

MATHSCHEME: MECHANISING THE MATHEMATICS PROCESS

OBJECTIVES

- To develop software systems that support the process people use to create, explore, connect, and apply mathematics
- To significantly advance mechanised mathematics
- To produce a single framework in which formal deduction and symbolic computation are tightly integrated and develop tools and techniques to support this approach

KEY COLLABORATORS

Michael Kohlhase, Jacobs University

Florian Rabe, Jacobs University

FUNDING

Natural Sciences and Engineering Research Council of Canada

CONTACT

Professor William M Farmer

Department of Computing and Software
McMaster University
1280 Main Street West
Hamilton, Ontario L8S 4K1
Canada

T +1 905 525 9140 x 27039
E wmfarmer@mcmaster.ca

WILLIAM M FARMER is Professor and Chair of the Department of Computing and Software at McMaster University in Hamilton, Ontario. He holds a PhD in mathematics from the University of Wisconsin-Madison. Before joining McMaster in 1999, he conducted research in computer science for 12 years at The MITRE Corporation in Bedford, Massachusetts and taught computer science for two years at St Cloud State University in St Cloud, Minnesota. His primary research interests are applied logic, computer support for mathematical reasoning, and rigorous methods in software development.

JACQUES CARETTE is Associate Professor in the Department of Computing and Software at McMaster University. He holds a PhD in mathematics from the Université de Paris-Sud. Before joining McMaster in 2002, he worked for 11 years at Maplesoft Inc in Waterloo, Ontario on all aspects of the Maple computer algebra system. His primary research interests are the building of mechanised mathematics systems, metaprogramming, programming languages and game design.

are intertwined. The second project goal is to design and implement an MMS based on this framework that will provide services for building formal languages, theories, computations, deductions and mappings between theories – all the services that are necessary for mechanising the mathematics process.

MATHSCHEME LANGUAGE AND LIBRARY

The team has already developed several techniques; some that lay the theoretical foundations of their framework and others that are implementation techniques. Specifically, their techniques rely on biform theories and Chiron – an expressive, general-purpose logic for mechanising mathematics. Two of the implementations the team is working on are the MathScheme language and MathScheme library.

The MathScheme language is a cross between a programming language and a language for mathematical knowledge capture. It has a user-orientated, high-level syntax (unlike Chiron) influenced by the team's previous work on high-level theories. "It has been designed to feel quite familiar, even though it has some rather novel semantics," Carette adds.

The MathScheme library, on the other hand, is the group's current repository of mathematical knowledge, which they are expanding to incorporate computer science, and is written in the MathScheme language. It is an experimental formalisation of the theories of abstract algebra, basic data-structures and structured type constructors. The library is organised by the tiny theories method in which knowledge is distributed over a network of theories that are built up one concept at a time. Much of the structure of the library resides in theory morphisms instead of in the theories themselves. "We have an expander (to see what our theories correspond to in traditional notation), a type-checker, pretty-printing facilities similar to Chiron's, as well as an experimental translator to Chiron," Carette explains. "The library plays a key role, as it embodies the most concrete form of our ideas." Current work is first focusing on leveraging the structure already present in the library to automatically generate as much information as possible using meta-programming techniques, rather than implementing all of this by hand, as is traditionally done. Eventually, the library will form a network of biform theories interconnected by theory morphisms.

IMPROVING EFFICIENCY IN MATHEMATICS

Carette and Farmer have many years of experience in building theorem-proving and symbolic computation systems. As Farmer reflects: "We were not interested in merely building a 'better' system; that would be too easy. We wanted to build something which would clearly be recognised as a 'next generation' system". They have thus spent a great deal of time on methodological aspects; analysing what they see as the strengths and weaknesses of current systems, especially when it comes to fundamental design decisions as well as methodology. "In particular, a lot of current systems make false choices between certain things, such as trading off correctness for efficiency, or vice versa," Carette asserts.

The application areas for MathScheme's products are wide-ranging but the team expects that the first uses will be in software engineering (for building dependable software) and in mathematics education. Ultimately, Carette and Farmer see the tools and systems they are developing as a means to an end; they are passionate that their research is aimed at improving people's efficiency, rather than being centred on any given technology.

The application areas for MathScheme's products are wide-ranging but the team expects that the first uses will be in software engineering (for building dependable software) and in mathematics education