Journées FastRelax, 25-27 mai 2016, Toulouse

Programme

- 25 mai
  - Accueil 10h30
  - 11h Fredrik Johansson, Hypergeometric functions in Arb
  - 12h Déjeuner
  - 14h-15h Bruno Salvy, Symbolic-Numeric Tools for Multivariate Asymptotics
  - 15h Pause
  - 15h30-16h30 Laurent Thery, Intro à Coq et conjecture de Goldbach
  - 16h30 Discussions
- 26 mai
  - 10h Thomas Sibut-Pinote, Formally Verified Approximations of Definite Integrals
  - 11h Tillmann Weisser, NLVerify Non Linear Verification - a coq-Tactic
  - 12h Déjeuner
  - 14h Jean-Michel Muller, Fast2Sum et 2Sum sont plus robustes qu'on le croit
  - 15h Siegfried Rump, Recent results on floating-point arithmetic
  - 16h Pause
  - 16h30 Valentina Popescu, Data-parallel algorithms for arithmetic operations using floating-point expansions
  - 17h30 Bruno Salvy, Bilan du projet
  - 20h Dîner
- 27 mai
  - 10h Mioara Joldes & Paulo Antares Gilz & Frédéric Camps sur les travaux reliés au RDV impulsionnel validé.
  - 12h Déjeuner
  - 14h Fin

- *Fredrik Johansson, Hypergeometric functions in Arb*

The Arb library now has essentially complete support for rigorously evaluating the classical (second-order ODE) hypergeometric functions, that is, the Gauss hypergeometric function 2F1, Kummer's 1F1 and U functions, and their special cases. I discuss algorithms, implementation issues, and optimizations.

- *Valentina Popescu, Data-parallel algorithms for arithmetic operations using floating-point expansions*

Many scientific computing applications demand massive numerical computations on parallel architectures such as Graphics Processing Units (GPUs). Often these problems require a higher precision than the standard double floating-point (FP) available. We develop CAMPARY: a multiple-precision arithmetic library, using the CUDA programming language for the NVidia GPU platform. In our approach, real numbers are represented as the unevaluated sum of several standard machine precision FP numbers. This representation is called a FP expansion and it offers the simplicity of directly using the underlying FP hardware level.
We present new data-parallel algorithms for adding and multiplying FP expansions specially designed for extended precision computations on GPUs. Error bounds on the results are given along with performance measurements for random generated computations in two settings: (i) optimistic, the program doesn't use any intermediate memory; (ii) realistic, shared memory is used for thread communication. A CPU version of the library is also available and we present some preliminary results in the context of semidefinite programming, on a benchmark of difficult ill-posed problems, where multiple-precision FP arithmetic is needed.

- *Siegfried Rump, Recent results on floating-point arithmetic*

  We discuss some recent results on floating-point arithmetic, in particular improved error bounds for sums, dot products and general arithmetic expressions.

- *Bruno Salvy, Symbolic-Numeric Tools for Multivariate Asymptotics*

In the study of enumeration, a standard method of deriving asymptotic information about a sequence of interest is to study properties of its generating function (an analytic function which encodes the sequence as its Taylor series). In many interesting applications, however, one does not have access to the univariate generating function directly but knows it as a sub-series of the multivariate Taylor expansion of a rational function in several variables. Trying to determine asymptotic information from this less direct representation is the domain of Analytic Combinatorics in Several Variables (ACSV), and raises many interesting questions from a computational viewpoint. In particular, the methods of ACSV typically require one to isolate specific points on algebraic varieties and decide semi-algebraic queries about them. This talk will describe algorithms for solving some of the problems in this area through a combination of symbolic and numeric techniques, and discuss applications in Combinatorics, Probability, and Number Theory.

- *Thomas Sibut-Pinote, Formally Verified Approximations of Definite Integrals*

Finding an elementary form for an antiderivative is often a difficult task, so numerical integration has become a common tool when it comes to making sense of a definite integral. Some of the numerical integration methods can even be made rigorous: not only do they compute an approximation of the integral value but they also bound its inaccuracy. Yet numerical integration is still missing from the toolbox when performing formal proofs in analysis.

We present an efficient method for automatically computing and proving bounds on some definite integrals inside the Coq formal system. Our approach is not based on traditional quadrature methods such as Newton-Cotes formulas. Instead, it relies on computing and evaluating antiderivatives of rigorous polynomial approximations, combined with an adaptive domain splitting. This work has been integrated to the CoqInterval library.

- *Tillmann Weisser, NLVerify Non Linear Verification - a coq-Tactic*

Within the last decade the computer science community's attention has been drawn to automatically proving non linear inequalities within a proof assistant. One driving project for this development has been the formal proof of the Kepler Conjecture by Thomas Hales, which was completed in August 2014. Nowadays, automated verification of inequalities finds it's applications in software verification and system control.

Proving inequalities is equivalent to showing non negativity of an objective function. With NLVerify we propose a coq-tactic to attack such goals by using a Sums of Squares (SOS) method.

However, in contrast to other SOS approaches NLVerify does not use the SOS-certificate as a representation of the objective function but to build a tight interval enclosure.

An SOS certificate can be found by using semi-definite programming (SDP) solvers. From the formal point of view, this certificate can be built by any unsafe tools. The interval enclosure however, needs to be build inside COQ. We use the interval library and verified floating points (Flocq) to do this.

For the moment, NLVerify only solves polynomial goals. Work on semi algebraic and univariate transcendental functions (tan, cos, ...) is in progress and expected to be implemented soon.