Effective Analysis: Foundations, Implementation and Certification
January 11 - 15, 2016

Andrej Bauer: Tutorial on Type theory.

Part I: Type theory and Equality reflection
We shall review the basics of dependent type theory and explain its computational content, especially in relation to computable mathematics. We shall then focus on the equality reflection rule, which gives type theory great expressive power at the price of ruining its good syntactic properties, such as strong normalization. Nevertheless equality reflection has a perfectly good validation in the context of computable mathematics and is therefore compatible with a computational understanding of type theory.

Part II: Type theory and Programming
We shall discuss how one might implement a mechanized proof checker, or even a proof assistant, for type theory with equality reflection. In the presence of equality reflection type checking is undecidable, which implies that a proof checker must necessarily receive advice in addition to the judgment it is supposed to check. We shall take the view that such advice amounts to a program and that the proof checker is simply an evaluator. We then face the design question: what sort of a programming language is suitable for describing type-theoretic derivations?

Part III: Type theory and Formalization
In the last part we will study how to use the expressive power of type theory with equality reflection and the associated programming language. Thanks to equality reflection we can describe not only the formation, introduction and elimination rules for almost any imaginable type constructor, but also its conversion rules. The associated programming language makes possible the implementation of commonly used formalization techniques (implicit arguments, type classes, canonical structures, etc) at the user level. This greatly reduces the complexity of the trusted kernel, and simultaneously gives the user flexibility that is not readily available in a proof assistant which relies on specific proof-checking techniques.

Ulrich Berger: Constructive logic for concurrent real number computation.

Tsuiki's infinite Gray-code [1] provides a digital representation of real numbers that is both constructive and non-redundant, that is, every real number has exactly one representation. This unlikely combination of properties is possible since an infinite Gray-code is a stream of digits that may be undefined at one position. Computation with infinite Gray-code can be modelled by a Turing Machine with two concurrently operating heads guaranteeing that at least one head will always read from a defined position. We present a logic that captures such concurrent computation. The logic is constructive in the sense that it allows for programs extraction from proofs. As an example we show how to extract a program that converts infinite Gray-code into signed digit representation.
[1] Hideki Tsuiki. Real Number Computation through Gray Code Embedding. Theoretical Computer Science, 284(2):467–485, 2002.

Sylvie Boldo: Formal verification of numerical analysis programs.

From a (partial) differential equation to an actual program is a long road. This talk will present the formal verification of all the steps of this journey. This includes the mathematical error due to the numerical scheme (method error), that is usually bounded by pen-and-paper proofs. This also includes round-off errors due to the floating- point computations.
The running example will be a C program that implements a numerical scheme for the resolution of the one-dimensional acoustic wave equation. This program is annotated to specify both method error and round-off error, and formally verified using interactive and automatic provers. Some work in progress about the finite element method will also be presented

Pieter Collins: Implementing Logic and Real Arithmetic.

In this talk I will discuss issues arising in implementing a general, configuarable, extensible, usable, and efficient library for real arithmetic in C++ and Haskell, based on work on Ariadne (C++) and AERN (Haskell). We aim to support Real, LowerReal and UpperReal types, and subtypes of Positive numbers, since all have uses in analysis. Further, the information provided by an object may be Abstract (a symbolic formula), Effective (an algorithm for computing arbitrarily accurate bounds), Validated (bounds to a finite precision) and Approximate (no guarantees on the error). Conversion from Abstract to Effective requires concrete Algorithms, whereas conversion from Effective to Validated requires an Accuracy or Precision, but ideally should be automated using defaults. Exact number types of Dyadic and Rational should be supported, and concrete numerical implementations may be given using double- or multiple-precison floating-point numbers, or fixed-point numbers. All this generality yields a jungle of classes; can we implement these in a clean and efficient way?
Joint work with Michal Konečný

Eva Darulova: Programming with Numerical Uncertainties.

Numerical software, common in scientific computing or embedded systems, inevitably uses an approximation of the real arithmetic in which most algorithms are designed. Finite-precision arithmetic, such as fixed-point or floating-point, is a common and efficient choice, but introduces an uncertainty on the computed result that is often very hard to quantify. We need adequate tools to estimate the errors introduced in order to choose suitable approximations which satisfy the accuracy requirements.

I will present a new programming model where the scientist writes his or her numerical program in a real-valued specification language with explicit error annotations. It is then the task of our verifying compiler to select a suitable floating-point or fixed-point data type which guarantees the needed accuracy. I will show how a combination of SMT theorem proving, interval and affine arithmetic and function derivatives yields an accurate, sound and automated error estimation which can handle nonlinearity, discontinuities and certain classes of loops.

Additionally, finite-precision arithmetic is not associative so that different, but mathematically equivalent, orders of computation often result in different magnitudes of errors. We have used this fact to not only verify but actively improve the accuracy by combining genetic programming with our error computation with encouraging results.

Boris Djalal: Newton sums for an effective formalization of algebraic numbers.

An algebraic number is defined by a polynomial with coefficients in the base field plus one piece of information to retain one root of the polynomial. We formalize a way to compute the composed sum and composed product of polynomials in Coq, which constitutes a component for addition and multiplication of algebraic numbers. In order to improve efficiency, our computation uses a representation of polynomials in term of Newton power series, which is the power series whose coefficients are the Newton sums; we formalize the algorithm to transform a polynomial into a Newton power series and vice versa. This requires the formalization of power series, which we achieve following the SSreflect style through truncated power series. Incidentally, we develop an abstract theory of poles of fractions.
Key words: formalization of mathematics, algebraic numbers, fractions, polynomials, Newton power series.

Fabian Immler: Verified Numerics for ODEs in Isabelle/HOL.

This talk is about verified numerical algorithms in Isabelle/HOL, with a focus on guaranteed enclosures for solutions of ODEs. The enclosures are represented by zonotopes, arising from the use of affine arithmetic. Enclosures for solutions of ODEs are computed by set-based variants of the well-known Runge-Kutta methods.

All of the algorithms are formally verified with respect to a formalization of ODEs in Isabelle/HOL: The correctness proofs are carried out for abstract algorithms, which are specified in terms of real numbers and sets. These abstract algorithms are automatically refined towards executable specifications based on lists, zonotopes, and software floating point numbers. Optimizations for low-dimensional, nonlinear dynamics allow for an application highlight: the computation of an accurate enclosure for the Lorenz attractor. This contributes to an important proof that originally relied on non-verified numerical computations.

Michal Konečný: Exact Real Number Computation in AERN.

Haskell provides a number of abstractions that can help make programs concise and clear, as well as fairly efficient. As these abstractions are solidly grounded in mathematics, the language is particularly attractive for implementing mathematics and, in particular, exact real number and function computation. I plan to give an overview of existing approaches to implementing exact real numbers and functions in Haskell, including various versions of the AERN library. The vision for the AERN library is to provide a convenient abstraction of the real numbers and functions with familiar composable mathematical semantics, while allowing the programmer to choose and optimise an evaluation strategy, including parallelisation and distributed deployment. I plan to demonstrate the current state of AERN and outline plans for its future development. I also plan to report on intermediate results of benchmarks that compare various representations of real numbers and continuous real functions.

Catherine Lelay: A new approach to formalize real numbers in the UniMath library.

There is various way to formalize real numbers in a proof assistant. The most common are axiomatization, Cauchy sequences, and Dedekind cuts. The choice of a particular formalization is strongly related to the chosen system, but also to the use of these real numbers. The UniMath library is a Coq library which aims to formalize mathematics using the univalent point of view. A consequences of univalence axioms is a better manipulation of sets. So, it is easy to chose Dedekind cuts to formalize real numbers in this library. Unfortunately, the definition may be tedious due to the numbers of hypothesis and to the number of cases to define some basic operations. I will present in this talk a new approach to formalize real numbers inspired by Dedekind cuts.

Rob Lewis: Algebra and Analysis in the Lean Theorem Prover.

Lean is a new proof assistant based on dependent type theory, being developed at Microsoft Research and CMU. A powerful and efficient type class mechanism allows us to construct and reason about the algebraic hierarchy and concrete number systems in a uniform, robust way. In this talk, I will explain the implementation of algebraic structures and the real numbers in Lean, and why this approach eases the development of further automation. I will also describe Polya, an automated tool for verifying real-valued inequalities which is being implemented in Lean.

Henri Lombardi: Towards a constructive theory of O-minimal structures.

We present a geometric theory for the algebra of real numbers (without sign test and not using dependent choice). This constructive approach to "real closed fields" is based on a good description/axiomatization of semialgebraic continuous functions from R^n to R. It can be seen as a constructive rewriting of the classical theory of real closed rings. By a convenient extension of this theory, our aim is to describe constructively the main properties of definable continuous functions for classical O-minimal structures, and to axiomatize these properties in a geometric theory.

Victor Magron: Certified Roundoff Error Bounds Using Semidefinite Programming and Formal Floating Point Arithmetic.

Roundoff errors cannot be avoided when implementing numerical programs with finite precision. The ability to reason about rounding is especially important if one wants to explore a range of potential representations, for instance in the world of FPGAs. This problem becomes challenging when the program does not employ solely linear operations as non-linearities are inherent to many interesting computational problems in real-world applications. Existing solutions to reasoning are limited in presence of nonlinear correlations between variables, leading to either imprecise bounds or high analysis time. Furthermore, while it is easy to implement a straightforward method such as interval arithmetic, sophisticated techniques are less straightforward to implement in a formal setting. Thus there is a need for methods which output certificates that can be formally validated inside a proof assistant. We present

a framework to provide upper bounds of absolute roundoff errors. This framework is based on optimization techniques employing semidefinite programming and sums of squares certificates, which can be formally checked inside the Coq theorem prover. A common issue is that semidefinite programming use finite precision floating point numbers, thus the sums of square certificate is only correct up to a certain numerical error. We address this issue by using, again, finite precision floating point numbers, this time inside Coq. More precisely, we consider polynomials in Coq whose coefficients are intervals of floating point numbers and we use the Coq library of floating point intervals constructed by Guillaume Melquiond. Our tool covers a wide range of nonlinear programs, including polynomials and transcendental operations as well as conditional statements. We illustrate the efficiency and precision of this tool on non-trivial programs coming from biology, optimization and space control.

Erik Martin-Dorel: CoqInterval: A Toolbox for Proving Non-linear Univariate Inequalities in Coq.

The verification of floating-point mathematical libraries requires computing numerical bounds on approximation errors. Due to the tightness of these bounds and the peculiar structure of approximation errors, such a verification is out of the reach of generic tools such as computer algebra systems. In fact, the inherent difficulty of c computing such bounds often mandates a formal proof of them. In this talk I will present the CoqInterval library [1], which offers tactics to automatically and formally prove bounds on univariate expressions involving elementary functions in the Coq proof environment. These tactics build upon a formalization of floating-point arithmetic, interval arithmetic, and Taylor models. All the computations are performed inside Coq's logic, relying on the reflection paradigm. I will also present some experimental results for comparing the performance of our tactic with various existing tools dedicated to the proof of inequalities over the reals.
[1] http://coq-interval.gforge.inria.fr/

Norbert Müller: Wrapping in Exact Real Arithmetic.

A serious problem common to all interval algorithms is that they suffer from wrapping effects, i.e. unnecessary growth of approximations during a computation. This is essentially connected to functional dependencies inside vectors of data computed from the same inputs. Reducing these effects is an important issue in interval arithmetic, where the most successful approach uses Taylor models.

In TTE Taylor models have not been considered explicitly, as they use would not change the induced computability, already established using ordinary interval computations. However for the viewpoint of efficiency, they lead to significant improvements.

In the talk we report on recent improvements on the iRRAM software for exact real arithmetic (ERA) based on Taylor models. The techniques discussed should also easily be applicable to other software for exact real computations as long as they also are based on interval arithmetic.

As instructive examples we consider the one-dimensional logistic map and a few further discrete dynamical systems of higher dimensions
Joint work with Franz Braußе, Trier, and Margarita Korovina, Novosibirsk. The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA

grant agreement n° PIRSES-GA-2011-294962-COMPUTAL and from the DFG/RFBR grant CAVER BE 1267/14-1 and 14-01-91334.

Iosif Petrakis: Bishop's Stone-Weierstrass theorem for compact metric spaces revisited.

In [1] and [2] Bishop formulated BSWcms, a theorem of Stone-Weierstrass type for compact metric spaces (i.e., complete and totally bounded metric spaces) using the notion of a (Bishop-)separating set of uniformly continuous real-valued functions. We present a direct constructive proof of a Stone-Weierstrass theorem for totally bounded metric spaces (SWtbms) which implies BSWcms. Our proof is elementary with a clear computational content, in contrast to Bishop's non-trivial proof of BSWcms and his hard to motivate concept of a Bishop-separating set of uniformly continuous functions. All applications of BSWcms found in [2] are proved directly by SWtbms. Key role to our proof plays the set of real-valued Lipschitz functions on a totally bounded metric space. We work within Bishop's informal system of constructive mathematics BISH.

[1] E. Bishop: Foundations of Constructive Analysis, McGraw-Hill, 1967.
[2] E. Bishop and D. Bridges: Constructive Analysis, Grundlehren der Math. Wissenschaften 279, Springer, Heidelberg-Berlin-New York, 1985.

Sebastian Posur: Category theory as a foundation for algorithms and programming in computer algebra.

Lots of problems in computer algebra can be stated in the language of category theory, e.g., the intersection of ideals as a pullback of two submodules, or the ideal membership problem as the existence of a lift along a given monomorphism. In my talk I will present CAP (short for categories, algorithms, and programming) which is a software project implemented in the computer algebra system GAP using the language of category theory as its foundation. In CAP it is possible to implement sophisticated algorithms and data structures only using basic categorical operations as primitives. This abstract approach makes a proper implementation of spectral sequence computations or Serre quotients not only feasible but also applicable in various different contexts.

Egbert Rijke: Localizations at omega-compact types as sequential colimits.

Homotopy type theory is a branch of mathematics in which ideas from many different fields of modern mathematics and computer science combine. Among its main distinctive features are Voevodsky's univalence axiom, which allows us to use a type theoretic universe as an object classifier, and higher inductive types, which allows presentations of types via generators and relations by means of their universal property. Among the higher inductive types are homotopy coequalizers, homotopy pushouts, the n-spheres for any n, cylinders, truncations, localizations, the Cauchy real numbers, and many more.

The 'homotopical' data contained in types can sometimes complicate working with them, so it is natural to consider situation in which the higher complexity of a type is limited. The simplest type is the contractible type, which has a term with the property that any other term may be identified with it. The next level consists of types all of whose identity

types are contractible, the so-called mere propositions. The types whose identity types are mere propositions are called sets, or 0-types, and now we may continue to define the (n+1)-truncated types (or simply (n+1)-types) to be those whose identity types are n-truncated. Intuitively, the n-types are those whose k-th identity types for k at least n+1 contain no information, which is why they are called truncated. The subuniverse of n-truncated types has a universal property which makes them pleasant to work with, namely that its inclusion functor in the actual universe has a left adjoint, and its left adjoint is in fact a higher modality.

Modalities are a more general phenomenon, including localization. For any type A, a type X is said to be A-local if any map $f : A \to X$ has a homotopy unique extension along the unique map $A \to 1$. Shulman has proposed a higher inductive type which constructs a left adjoint to the inclusion of the A-local types into the (univalent) universe of all types, called localization at A. The n-truncated types are precisely the S^{n+1}-local types, where S^{n+1} is the (n+1)-sphere. The left adjoint of the inclusion of the n-truncated types into the universe has a special name: the n-truncation. The n-truncation is given as a recursive higher inductive type, so this has raised the question of formulating a theory of higher inductive types which is general enough to fit these in a rigid scheme.

However, the spheres have a nice feature which grants us some more possibilities: they are omega-compact. We show that localizations at omega-compact types can be obtained as a sequential colimit of homotopy coequalizers, which are the most basic higher inductive type without self-referential in the constructors. In particular, we obtain that the n-truncations, which have previously only been understood as a recursive higher inductive type, can be described as a certain homotopy coequalizer.
This is joint work with Floris van Doorn (CMU), and work in progress.

Monika Seisenberger: Verification of Discrete and Real-timed Railway Control Systems.

The objective of this talk is to give an overview of logical methods used in the verification of traditional solid state interlockings and the European Rail Traffic Management System (ERTMS).

The first part is concerned with traditional Railway interlockings, often specified using a graphical language, called Ladder Logic. We give a semantics for this language and show how to get from such a specification to a SAT solving problem. This process has been automated, and realistic Interlocking examples, provided by our industrial partner Siemens Rail Automation, have been verified using various automated theorem proving tools [1]. We further applied our own SAT solver, which we extracted from a formal constructive proof of the completeness of the DPLL proof system. The extracted SAT solver is a verified algorithm, which either yields a model or a DPLL refutation of a given clause set [2].

In the second part, we present our modelling of ERTMS, a next generation train control system, which aims at improving the performance/capacity of rail traffic systems, without compromising their safety. It generalizes from traditional interlockings to a system that includes on-board equipment and communication between trains and interlockings via radio block processors. Whilst the correctness of discrete interlocking systems is well-researched, it is challenging to verify ERTMS based systems for safety properties such as collision freedom due to the involvement of continuous data [3]. The modelling and verification is done in Real-Time Maude, a tool that allows for both simulation and verification of real-time and hybrid systems.

References

[1] P. James, A. Lawrence, F. Moller, M. Roggenbach, M. Seisenberger, A. Setzer, S. Chadwick, and K. Kanso, *Verification of Solid State Interlocking Programs*, In SEFM'13, LNCS **8368** (2014), 253–268.

[2] U. Berger, A. Lawrence, F. Nordvall Forsberg, M. Seisenberger, *Extracting Verified Decision Procedures: DPLL and Resolution.* Logical Methods in Computer Science 11(1:6), 2015.

[3] P. James, A. Lawrence, M. Roggenbach, M. Seisenberger. *Towards Safety Analysis of ERTMS/ETCS Level 2 in Real-Time Maude*, FTSCS 2015. To appear, Springer, 2016.

Peter Selinger: Combining numerical and number-theoretic methods to solve unitary approximation problems in quantum computing.

An important problem in quantum computing is the approximation of arbitrary unitary operators by quantum circuits that are build from some given finite set of gates. Preferably, the circuits should be short, and should be computed by an efficient algorithm. For nearly two decades, the standard solution to this problem was the Solovay- Kitaev algorithm, which is based on geometric ideas. This algorithm produces circuits of size $O(\log^c(1/\epsilon))$, where c is approximately 3.97. It was a long- standing open problem whether the exponent c could be reduced to 1.

In this talk, I will answer this question positively by reporting on a new class of randomized number-theoretic algorithms that achieve circuit size $O(\log(1/\epsilon))$ in the case of the commonly used Clifford+T gate set. In case the operator to be approximated is diagonal, the algorithm satisfies an even stronger property: it computes the optimal solution to the given approximation problem. In order to achieve optimal circuit sizes, the algorithm requires an oracle for integer factoring (such as a quantum computer); in the absence of a factoring oracle, the algorithm is still nearly optimal. Furthermore, termination of the algorithm is predicated on an unproven, but heuristically true, hypothesis about the distribution of prime numbers. A crucial element of the algorithm is the combination of computations in algebraic number fields (which are exact) with real-number computations (which can be done with variable precision).
This is joint work with Neil J. Ross.

Bas Spitters: Cubical sets as a classifying topos.

Homotopy Type Theory refers to a new interpretation of Martin-Löfs system of intensional, constructive type theory into abstract homotopy theory. Propositional equality is interpreted as homotopy and type isomorphism as homotopy equivalence. Logical constructions in type theory then correspond to homotopy-invariant constructions on spaces, while theorems and even proofs in the logical system inherit a homotopical meaning. As the natural logic of homotopy, constructive type theory is also related to higher category theory as it is used e.g. in the notion of a higher topos.

The univalent foundations aims to provide a foundation for mathematics based on the homotopical interpretation of type theory. Voevodsky's beautiful univalence axiom relates propositional equality on the universe with homotopy equivalence of small types. It provides the universe with the universal properties of an object classifier from higher toposes. Coquand's cubical set model and the related cubical type checker provide a computational interpretation for the univalent foundations.

We will provide a description of this model based on the constructive mathematics of the topos of cubical sets. To be precise, we will show what this topos classifies and how this

helps us to simplify the presentation of the model. Specifically, this will allow us to describe the geometrical realization as a geometric morphism from Johnstone's topological topos.

http://homotopytypetheory.org/

Laurent Théry: Proof and Computation.

In this talk, we are going to show on some elementary examples how computation can easily be incorporated inside proof in a proof system like Coq.

Geum Young Hee: Basins of attraction for optimal third-order multiple-root finders.

To ensure the convergence of an iterative method, it is significant to take a good initial guess close to the desired zero of the given nonlinear equation. We discuss the complex dynamical behavior of optimal third-order multiple-root finder by using their basins of attraction and provide the statistical date for the average number of iterations per point and the number of divergent points including CPU time.