Towards Certified Algorithms for Exact Real Arithmetic

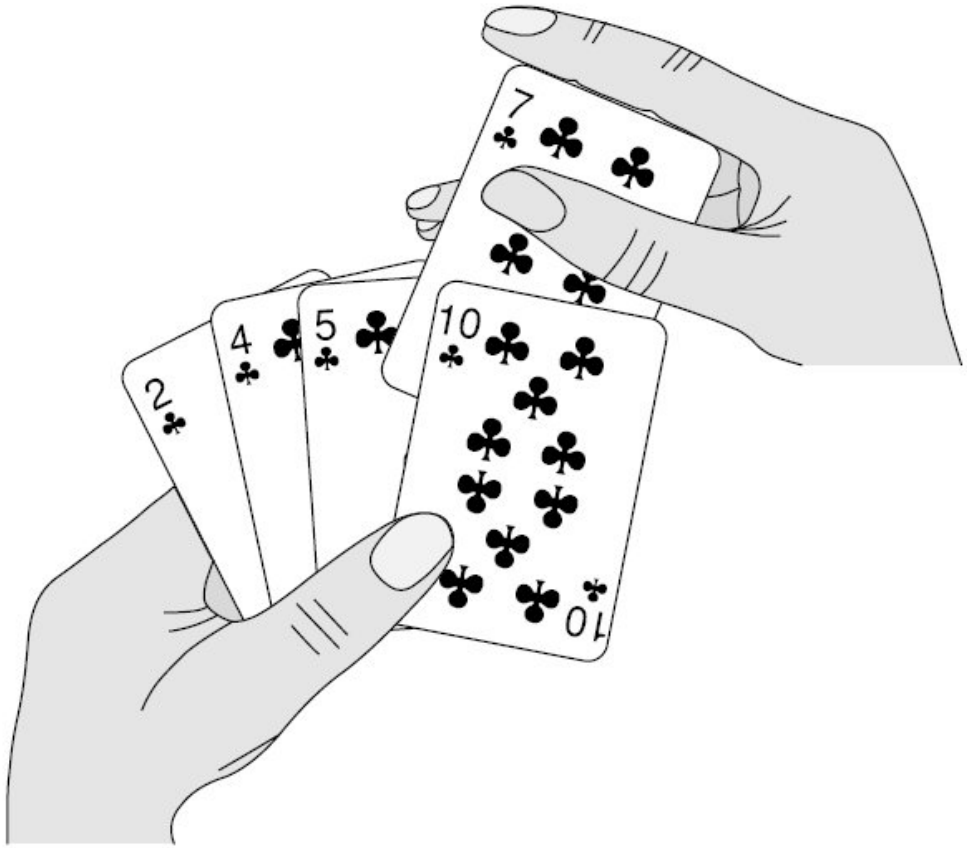
CCC 2017

(LORIA, Nancy, June 26-30 2017)

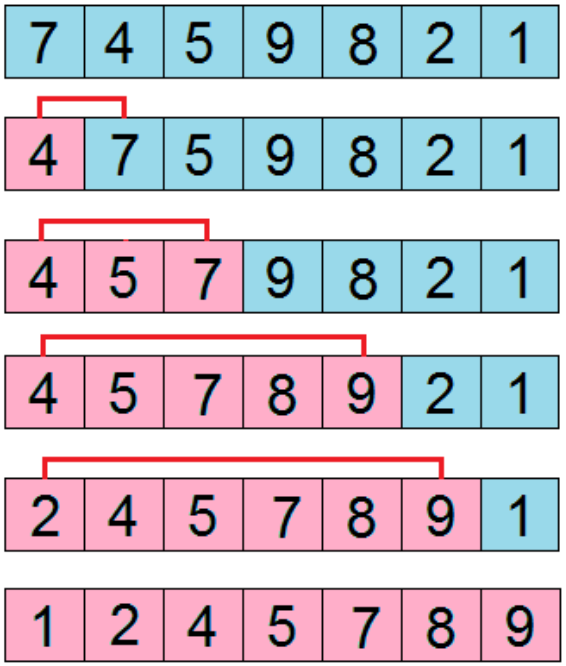
Sunyoung Kim, Gyesik Lee, Sewon Park, Martin Ziegler

An Example

Example: Insertion Sort



Example: Insertion Sort




```
void insertion_sort ( int *data, int n )
{
  int i, j, remember;
  for ( i = 1; i < n; i++ )
  {
    remember = data[(j=i)];
    while ( --j >= 0 && remember < data[j] )
        data[j+1] = data[j];
    data[j+1] = remember;
  }
}
```

Example: Insertion Sort in Coq

```
Definition insert (n:nat) (l:list nat) : list nat.
Proof.
  (* Description of a recursive algorithm*)
Defined.

Definition sort (l:list nat) : list nat.
Proof.
  (* Description of a recursive algorithm*)
Defined.
```

Example: Insertion Sort in Coq

```
Definition sort_spec (l:list nat) :
  {l' | sorted l' /\ permutation l l'}.
Proof.
  (* Description of a recursive algorithm and
     proof of the required property *)
Defined.
```

Example: Insertion Sort in Coq

```
Definition sort_spec (l:list nat) :
  {l' | sorted l' /\ permutation l l'}.
Proof.
  (* Description a recursive algorithm and
     proof of the required property *)
Defined.

Extraction Language Ocaml.

Extraction "insert_sort.ml" sort_spec.
```

A Historic Case:

Why Certified Algorithms matter!

Hales' proof of the Kepler conjecture

No arrangement of equally sized spheres filling space has a greater average density than that of the cubic close packing and hexagonal close packing arrangements.

Hales' proof of the Kepler conjecture

- Hales' proof in August 1998 consisted of
  - 300 pages of texts and
  - 3 Gigabytes of computer programs and data.

Hales' proof of the Kepler conjecture

- Hales' proof in August 1998 consisted of
  - 300 pages of texts and
  - 3 Gigabytes of computer programs and data.

- Submitted to Ann. Math.
  - after 5 years of refereeing process
  - the panel of 12 referees was 99% certain of the correctness of the proof.
  - Ann. Math. published the text proofs (121 pages long) only.

Geuvers' comments

- Hales needed to prove that 1039 complicated inequalities hold.
- He used computer programs to verify the inequalities.
- The referees had problems with his approach:
  - verifying the inequalities themselves by hand would be impossible
  - one week per inequality is still 25 man years of work.
- They could not consider to verify the computer programs Hales used.

Computerization of mathematical proofs

- In 2004, Hales himself announced his intention to have *formal* version of his original proof.

- His intention was then realized through a project called *Flyspeck* on 10th August 2014, 10 years after his announcement.

- Two proof assistants, *HOL Light* and *Isabelle, are used*.

- Finally published in "Forum of Mathematics, Pi" on May 29, 2017.

Computerization of mathematical proofs

May 2017

Forum of Mathematics, Pi (2017), Vol. 5, e2, 29 pages
doi:10.1017/fmp.2017.1

A FORMAL PROOF OF THE KEPLER CONJECTURE

THOMAS HALES[1], MARK ADAMS[2,3], GERTRUD BAUER[4], TAT DAT DANG[5], JOHN HARRISON[6], LE TRUONG HOANG[7], CEZARY KALISZYK[8], VICTOR MAGRON[9], SEAN MCLAUGHLIN[10], TAT THANG NGUYEN[7], QUANG TRUONG NGUYEN[1], TOBIAS NIPKOW[11], STEVEN OBUA[12], JOSEPH PLESO[13], JASON RUTE[14], ALEXEY SOLOVYEV[15], THI HOAI AN TA[7], NAM TRUNG TRAN[7], THI DIEP TRIEU[16], JOSEF URBAN[17], KY VU[18] and ROLAND ZUMKELLER[19]

[1] University of Pittsburgh, USA;
email: hales@pitt.edu, nguyenquangtruong270983@gmail.com
[2] Proof Technologies Ltd, UK
[3] Radboud University, Nijmegen, The Netherlands;
email: mark@proof-technologies.com
[4] ESG – Elektroniksystem- und Logistik-GmbH, Germany;
email: Gertrud.Bauer@alumni.tum.de
[5] CanberraWeb, 5/47-49 Vicars St, Mitchell ACT 2911, Australia;
email: dangtatdatusb@gmail.com
[6] Intel Corporation, USA;
email: johnh@ecsmtp.pdx.intel.com
[7] Institute of Mathematics, Vietnam Academy of Science and Technology, Vietnam;
email: hltruong@math.ac.vn, ntthang.math@gmail.com, tthan@math.ac.vn, tntrung@math.ac.vn
[8] University of Innsbruck, Austria;
email: cezary.kaliszyk@uibk.ac.at
[9] CNRS VERIMAG, France;
email: magron@lix.polytechnique.fr
[10] Amazon, USA;
email: seanmcl@gmail.com
[11] Technische Universität München, Germany;
email: nipkow@in.tum.de
[12] University of Edinburgh, UK;
email: sobua@inf.ed.ac.uk
[13] Philips Electronics North America Corporation – Andover, MA, USA;
email: joe.pleso@gmail.com
[14] The Pennsylvania State University, USA;
email: jason.rute@gmail.com

[15] University of Utah, USA;
email: solovyev.alexey@gmail.com
[16] AXA China Region Insurance Company Limited, Hong Kong;
email: trieudiep87@gmail.com
[17] Czech Institute of Informatics, Robotics and Cybernetics (CIIRC), Czech Republic;
email: urban@cs.ru.nl
[18] Chinese University of Hong Kong, Hong Kong;
email: vukhacky@gmail.com
[19] email: Roland.Zumkeller@gmail.com

Received 21 November 2014; accepted 9 December 2016

Abstract

This article describes a formal proof of the Kepler conjecture on dense sphere packings in a combination of the HOL Light and Isabelle proof assistants. This paper constitutes the official published account of the now completed Flyspeck project.

2010 Mathematics Subject Classification: 52C17

1. Introduction

The booklet *Six-Cornered Snowflake*, which was written by Kepler in 1611, contains the statement of what is now known as the Kepler conjecture: no packing of congruent balls in Euclidean three-space has density greater than that of the face-centered cubic packing [27]. This conjecture is the oldest problem in discrete geometry. The Kepler conjecture forms part of Hilbert's 18th problem, which raises questions about space groups, anisohedral tilings, and packings in Euclidean space. Hilbert's questions about space groups and anisohedral tiles were answered by Bieberbach in 1912 and Reinhardt in 1928. Starting in the 1950s, Fejes Tóth gave a coherent proof strategy for the Kepler conjecture and eventually suggested that computers might be used to study the problem [6]. The truth of the Kepler conjecture was established by Ferguson and Hales in 1998, but their proof was not published in full until 2006 [18].

The delay in publication was caused by the difficulties that the referees had in verifying a complex computer proof. Lagarias has described the review process [30]. He writes, 'The nature of this proof . . . makes it hard for humans to check every step reliably. . . . [D]etailed checking of many specific assertions found them to be essentially correct in every case. The result of the reviewing process produced in these reviewers a strong degree of conviction of the essential correctness of this proof approach, and that the reduction method led to nonlinear programming problems of tractable size.' In the end, the proof was published without complete certification from the referees.

Computerization of mathematical proofs

Formal proofs? Coq, Isabelle? Proof assistants?

Practice in Numerical Engineering

(excerpted from a work by Müller and Ziegler, 2014)

Practice in Numerical Engineering

It generally neglects questions of correctness,

leading to a mix of criticism and fatalism.

Practice in Numerical Engineering

”How do you know that your answers are as accurate as you claim?”

Practice in Numerical Engineering

- Typical answers are
  - *"I tested the method with some simple examples and it worked"*,
  - *"I repeated the computation with several values of n and the results agreed to three decimal places"*,
  - *"the answers looked like what I expected"*,
  - …

Practice in Numerical Engineering

- Typical answers are
  - *"I tested the method with some simple examples and it worked"*,
  - *"I repeated the computation with several values of n and the results agreed to three decimal places"*,
  - *"the answers looked like what I expected"*,
  - …

There are many instances of programs that delivered incorrect results for a considerable period of time before the error was found.

Exact Real Arithmetic (ERA)

Exact Real Arithmetic (ERA)

Convenient and practically efficient framework

for rigorous numerical algorithms.

(as propagated by Müller and Ziegler, 2014)

Exact Real Arithmetic (ERA)

- ERA consists of, and combines, four aspects:
  1. Recursive Analysis — the Theory of Computing over real numbers, (smooth) functions, and (closed) Euclidean subsets.
  2. Real Complexity Theory as resource-oriented refinement of (1).
  3. An imperative programming language with rigorous semantics of computable operations on continuous objects appearing as entities (ERA).
  4. A library implementing, and efficiently realizing, (much of) the semantics according to (3) such as
     - C++ library for iRRAM

Exact Real Arithmetic (ERA)

- ERA consists of, and combines, five aspects:
  1. Recursive Analysis — the Theory of Computing over real numbers, (smooth) functions, and (closed) Euclidean subsets.
  2. Real Complexity Theory as resource-oriented refinement of (1).
  3. An imperative programming language with rigorous semantics of computable operations on continuous objects appearing as entities (ERA).
  4. A library implementing, and efficiently realizing, (much of) the semantics according to (3) such as
     - C++ library for iRRAM
  5. Formal verification of the tools or library developed based on (3) and (4).

How to Approach to Formal Verification

1. Extending Hoare Logic

Extending Hoare Logic

- *Hoare logic* is a formal system with a set of logical rules for reasoning rigorously about the correctness of computer programs.
- A specification of a program C is written by a Hoare triple:

$$\{P\}\ C\ \{Q\}$$

- P and Q are predicates describing possible states of mutable variables.

Extending Hoare Logic

- *Hoare logic* is originally introduced using a very simple imperative language and subsequently refined by many researchers.

- Separation logic is an extension than can deal with pointers and local reasoning.

Extending Hoare Logic

Algorithm 5 Gaussian Elimination using multivalued test semantics

```
 1: procedure GAUSS(n, r : INTEGER, A[n,n] : REAL, var x[n] : REAL)        // Require: rank(A)= r
 2:     var i, j, k, pi, pj, π[n] : ℕ;  var t, B[n,n] : ℝ                  // Return some x ≠ 0 such that A · x = 0
 3:     for i := 1 to n do                                                 // Initialization B := A, π := id
 4:         π[i] := i ; for j := 1 to n do B[i,j] := A[i,j] end for
 5:     end for
 6:     for k := 1 to r do                                                 // Convert B into reduced row echelon form:
 7:         CHOOSEPIVOT(n, k, B, pi, pj)                                   // Find pi, pj such that B[pi, pj] ≠ 0.
 8:         for j := 1 to n do swap(B[k,j], B[pi,j]) end for               // Exchange rows #k and #pi.
 9:         for i := 1 to n do swap(B[i,k], B[i,pj]) end for               // Exchange columns #k and #pj.
10:         swap(π[k], π[pj])
11:         for j := k to n do                                             // Scale row #k by 1/B[k,k]
12:             B[k,j] := B[k,j]/B[k,k]                                    // and subtract B[i,k]-fold from rows #i = k+1 ... n:
13:             for i := k+1 to n do B[i,j] := B[i,j] − B[i,k] ∗ B[k,j] end for
14:         end for
15:     end for
16:     x[π[n]] := 1                                                       // Back-substition for x, taking into account permutation π:
17:     for i := n − 1 downto 1 do
18:         t := 0 ; for j := n downto i+1 do t := t + B[i,j] ∗ x[π[j]] end for
19:         x[π[i]] := −t
20:     end for
21: end procedure
```

(Gaussian Elimination with comments by Müller et al. 2016)

Extending Hoare Logic

- *Sewon Park* has presented a simple extension of Hoare logic supporting part of ERA in iRRAM.
- We hope to extend it and formally verify its soundness.
- The proof assistant Coq is our tool for verification.

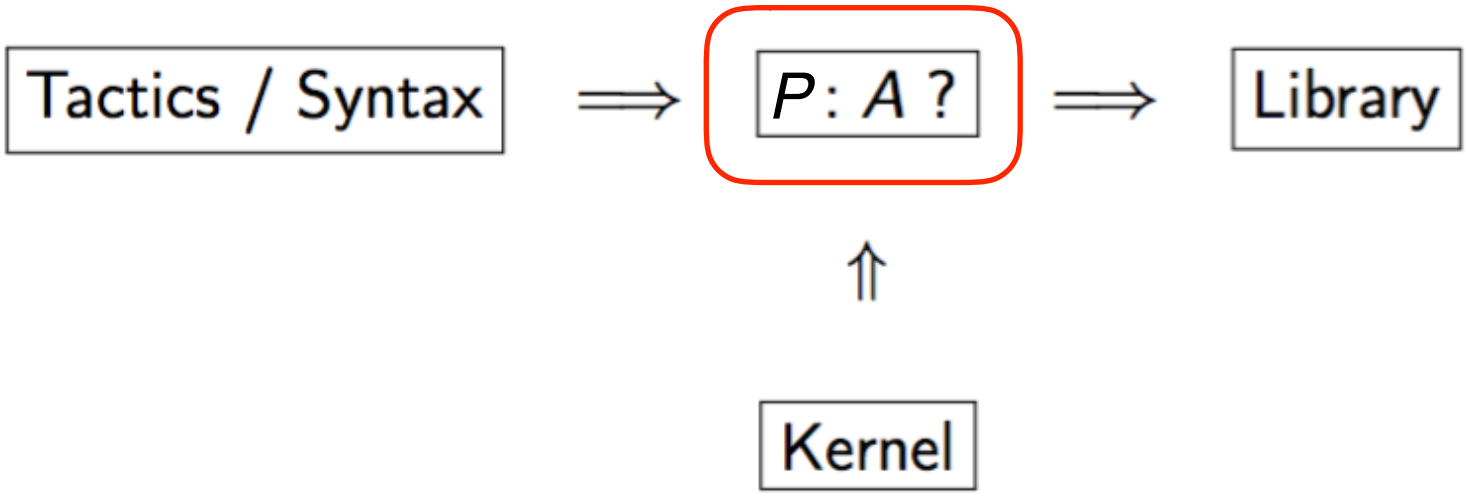
Understanding proof assistants

- A proof assistant
  - is a computer software to assist with the development of proofs by human-machine interaction

Understanding proof assistants

- A proof assistant
  - is a computer software to assist with the development of proofs by human-machine interaction
  - and contains some sort of interactive proof editor with which a human can guide the search for proofs.

Type checking in Coq



Some proof assistants

- Agda
  - Unified Theory of Dependent Types (UTT)
- Coq
  - Calculus of Inductive Constructions (CIC)
- HOL family (HOL4, HOL Light, ProofPower)
  - A classical higher-order logic
- Isabelle
  - Zermelo-Fraenkel set theory (ZFC), higher-order logic
- Minlog
  - First order natural deduction calculus
- Mizar
  - Tarski–Grothendieck set theory with classical logic
- PVS
  - A classical, typed higher-order logic

2. Using Tools for Source Code Analysis

Using tools for source code analysis

- Combination of
  - Why3
  - Frama-C
  - Coq
  - Libraries for Reals such as C-CORN, Mathcomp, …

Using tools for source code analysis

- Combination of
  - Why3
  - Frama-C
  - Coq
  - Libraries for Reals such as C-CORN, Mathcomp, …

- N. Müller has already achieved some progress.

Using tools for source code analysis

- Combination of
  - Why3
    - platform for deductive program verification
    - with a language for specification and programming
    - relying on external theorem provers
    - with a standard library of logical theories such as integer, reals, Boolean, sets, maps, …
  - Frama-C
  - Coq
  - Libraries for Reals such as C-CORN, Mathcomp, …

Using tools for source code analysis

- Combination of
  - Why3
  - Frama-C
    - a suite of tools dedicated to the analysis of the source code of software written in C
    - gathers several static analysis techniques in a single collaborative framework
  - Coq
  - Libraries for Reals such as C-CORN, Mathcomp, …

Using tools for source code analysis

- Combination of
  - Why3
  - Frama-C
  - Coq
  - Libraries for Reals such as C-CORN, Mathcomp, …
    - C-Corn: a huge library for constructive mathematics developed in Nijmegen
    - Mathcomp: implementation of Algebraic Real Numbers by Cyril Cohen
    - finding a suitable implementation of reals in Coq would be not so simple

3. In the Style of CompCert

CompCert

- CompCert is a formally verified optimizing compiler for a large subset of the C99 programming language which currently targets 32-bit PowerPC, ARM, x86 and x86-64 architectures.

CompCert

- CompCert is a formally verified optimizing compiler for a large subset of the C99 programming language which currently targets 32-bit PowerPC, ARM, x86 and x86-64 architectures.
- The compiler is specified, programmed and proved in Coq.
- The performance of its generated code is often close to that of GCC.

CompCert

- Some experts in CompCert think it would be possible to formalize everything about the tools like iRRAM in the style of CompCert.

CompCert

- Some experts in CompCert think it would be possible to formalize everything about the tools like iRRAM in the style of CompCert.
- We are going to check it, at least partly:
  - necessary types
  - suitable semantics
  - implementation of reals (when necessary)
  - soundness check
  - …

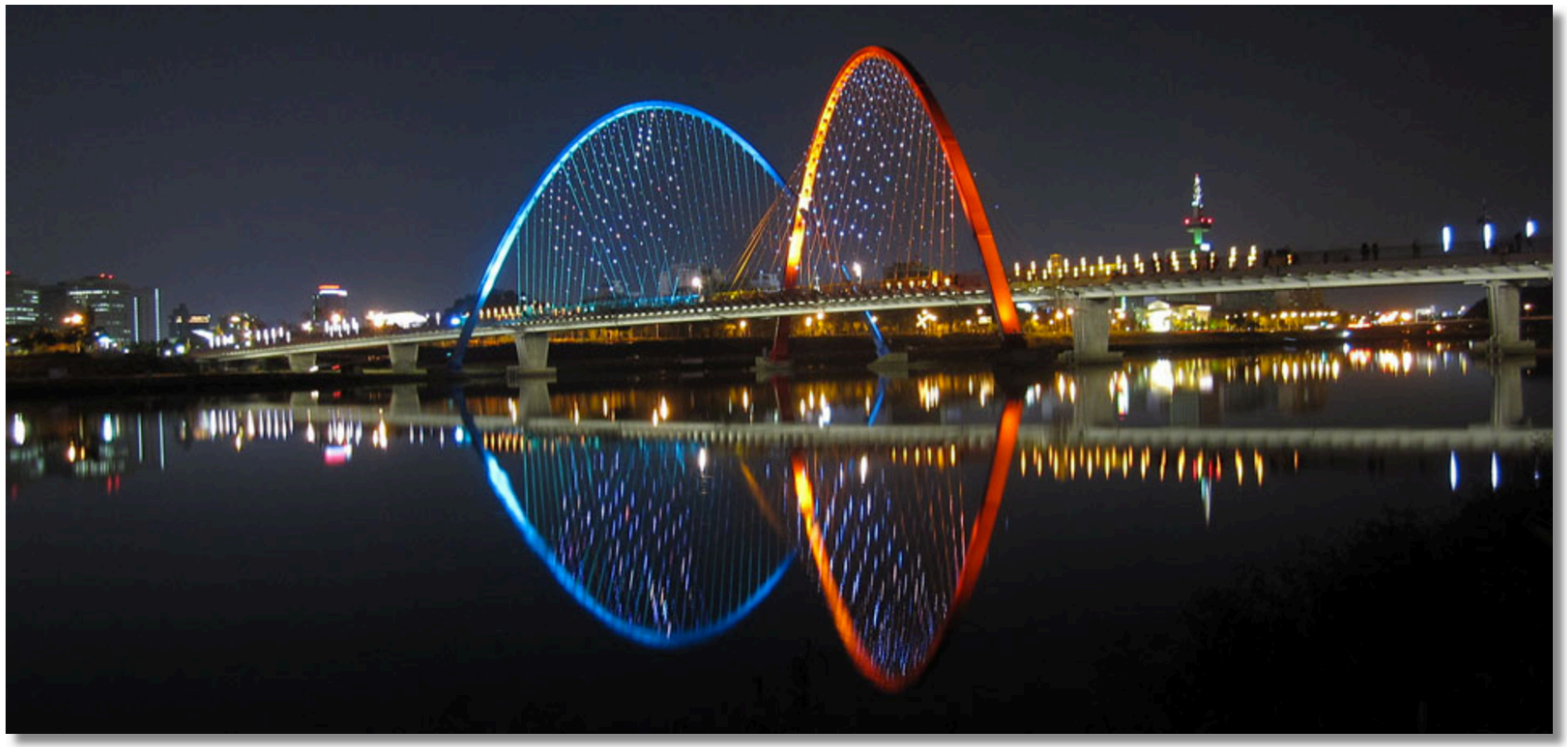
CompCert

- Some experts in CompCert think it would be possible to formalize everything about the tools like iRRAM in the style of CompCert.
- We are going to check it, at least partly:
  - necessary types
  - suitable semantics
  - implementation of reals (when necessary)
  - soundness check
  - …
- Probably, the work on *Ariadne* and *AERN* should be studied
  - to understand how they are built

CCA 2017

Fourteenth International Conference on Computability and Complexity in Analysis

July 24-27, 2017, Daejeon, Republic of Korea



Workshop on Real Verification

Workshop on Real Verification

July 28 (2017), organized by KAIST's School of Computing and co-located with the 14th International Conference on Computability and Complexity in Analysis

Invited Speakers

- Cyril Cohen, Inria Sophia-Antipolis, France
- Jeehoon Kang, Seoul National University, Republic of Korea
- Johannes Kanig, AdaCore
- Sunyoung Kim, Yonsei University, Republic of Korea
- Michal Konečný, Aston University, United Kingdom-
- Norbert Müller, Universität Trier, Germany
- Sukyoung Ryu, KAIST, Republic of Korea

Organizers

- Gyesik Lee, Hankyong National University, Republic of Korea
- Martin Ziegler, KAIST School of Computing, Republic of Korea