DedekindCutArithmetic.jl: A Julia implementation of exact real arithmetic based on Dedekind cuts

Luca Ferranti, Aalto University

In CCA 2008, Andrej Bauer introduced Marshall (Bauer 2008), a programming language written in OCaml for exact real arithmetic based on Abstract Stone Duality (Bauer and Taylor 2009). At the end of the presentation, a challenge was thrown to the audience, *Can this be implemented as library in an existing programming language?*. Today, we show the answer is yes.

This talk introduces DedekindCutArithmetic.jl, a library that allows for computations with exact reals. Leveraging Julia's lisp-style syntactic macros, `DedekindCutArithmetic.jl` offers an embedded domain specific language (eDSL), which allows to write cuts and expressions with quantifiers in a notation resembling the traditional mathematical one.

Since it is an eDSL, it canc ompose with to other libraries in the Julia ecosystem. For example, similarly to Marshall, the library relies on interval Newton method to speed up refinement of cuts and quantifiers. However, the derivative is computed using `ForwardDiff.jl` (Revels, Lubin, and Papamarkou 2016), a library for high-performance forward-mode automatic differentiation. As an additional advantage, being the language embedded in Julia, `DedekindCutArithmetic.jl` automatically inherits all the features of the language, such as support for higher-order functions and recursion, without the need to be re-implemented.

During this talk, I will introduce the features of `DedekindCutArithmetic.jl`, focusing on its architecture, design choices and trade-offs. I will also demostrate in a few examples how the library can be composed with other libraries in the Julia ecosystem, to be applied to computational geometry and scientific computing domains. Finally, I will also give an overview of the roadmap for the library, describing current limitations, next steps and open-questions.

Since few lines of code generally say more than 1000 words, the following copy-pastable working example demonstrates the syntax and some core functionalities of the library.

```julia
using DedekindCutArithmetic

# Textbook example of dedekind cuts, define square-root
my_sqrt(a) = @cut x ∈ ℝ, (x < 0) ∨ (x * x < a), (x > 0) ∧ (x * x > a)

sqrt2 = my_sqrt(2);

# evaluate to 80 bits precision, this gives an interval with width <2^-80 containing √2
refine!(sqrt2; precision=80)
# [1.4142135623730949, 1.4142135623730951]

# Define maximum of a function f: [0, 1] → ℝ as a Dedekind cut
my_max(f::Function) = @cut a ∈ ℝ, ∃(x ∈ [0, 1] : f(x) > a), ∀(x ∈ [0, 1] : f(x) < a)

f = x -> x * (1 - x)

fmax = my_max(f);

refine!(fmax) # evaluate to 53 bits of precision by default
# [0.24999999999999992, 0.25000000000000006]
```

References

Bauer, Andrej. 2008. "Efficient Computation with Dedekind Reals." In *Fifth International Conference on Computability and Complexity in Analysis,(eds. V Brattka, r Dillhage, t Grubba, a Klutch), Hagen, Germany*. Citeseer.

Bauer, Andrej, and Paul Taylor. 2009. "The Dedekind Reals in Abstract Stone Duality." *Mathematical Structures in Computer Science* 19 (4): 757–838.

Revels, J., M. Lubin, and T. Papamarkou. 2016. "Forward-Mode Automatic Differentiation in Julia." *arXiv:1607.07892 [Cs.MS]*. https://arxiv.org/abs/1607.07892.