# Computation in Real Closed Infinitesimal and Transcendental Extensions of the Rationals.

**Citation for published version:**
de Moura, LM & Passmore, GO 2013, Computation in Real Closed Infinitesimal and Transcendental Extensions of the Rationals. in MP Bonacina (ed.), Automated Deduction – CADE-24: 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings. vol. 7898, Lecture Notes in Computer Science, Springer Japan, pp. 178-192. https://doi.org/10.1007/978-3-642-38574-2_12

**Digital Object Identifier (DOI):**
10.1007/978-3-642-38574-2_12

**Link:**
Link to publication record in Edinburgh Research Explorer

**Document Version:**
Peer reviewed version

**Published In:**
Automated Deduction – CADE-24

# Computation in Real Closed Infinitesimal and Transcendental Extensions of the Rationals

Leonardo de Moura[1] and Grant Olney Passmore[2]
leonardo@microsoft.com, grant.passmore@cl.cam.ac.uk

[1] Microsoft Research, Redmond
[2] LFCS, Edinburgh and Clare Hall, Cambridge [⋆]

**Abstract.** Recent applications of decision procedures for nonlinear real arithmetic (the theory of real closed fields, or RCF) have presented a need for reasoning not only with polynomials but also with transcendental constants and infinitesimals. In full generality, the algebraic setting for this reasoning consists of *real closed transcendental and infinitesimal extensions of the rational numbers*. We present a library for computing over these extensions. This library contains many contributions, including a novel combination of Thom's Lemma and interval arithmetic for representing roots, and provides all core machinery required for building RCF decision procedures. We describe the abstract algebraic setting for computing with such field extensions, present our concrete algorithms and optimizations, and illustrate the library on a collection of examples.

## 1 Overview and Related Work

Decision methods for nonlinear real arithmetic are essential to the formal verification of cyber-physical systems and formalized mathematics. Classically, these decision methods operate over the *theory of real closed fields* (RCF), the first-order theory of the reals with addition, multiplication and equality and inequality predicates. RCF is decidable, admits quantifier elimination, and a variety of mature (though necessarily worst-case hyperexponential) decision procedures exist for it. Much research has gone into making RCF decision procedures practical, especially for restricted classes of formulas commonly arising in applications.

In recent years, it has become apparent that the classical approach underlying most complete RCF methods, the 'real algebraic number approach,' is insufficient for many applications. This approach has its roots in the most influential strand of RCF decision procedure research, the theory of *cylindrical algebraic decomposition* (CAD) [2,3]. In CAD and related techniques, one makes crucial use of the following observation: Since RCF is a complete theory with $\langle \mathbb{R}, +, *, <, 0, 1 \rangle$ as a model, then, when implementing an RCF decision procedure, one is free to compute over *any* RCF while still being sure that the resulting computations

[⋆] Grant Passmore was supported by the EPSRC [EP/I011005/1 and EP/I010335/1] and by NSF EXPEDITION CNS-0926181.

are valid over $\mathbb{R}$. This is important from a computational point of view, as $\mathbb{R}$ is uncountable with uncomputable basic operations.

In the classical approach, instead of working over $\mathbb{R}$, one works over the *real algebraic numbers* $\mathbb{R}_{alg}$, the subfield of $\mathbb{R}$ consisting of real numbers that are roots of univariate polynomials with integer coefficients. This structure is a countable real closed field with computable basic operations, and thus provides a logically sufficient computational substructure for making RCF decisions.[3] Note, though, that this field contains no *transcendental* elements such as $\pi$ or $e$. Indeed, a real number is transcendental precisely when it is not algebraic.

On the one hand, this lack of transcendental elements seems logically inconsequential and even computationally desirable, as transcendentals are undefinable over RCF and almost all of them are uncomputable. However, various new applications have given rise to a need for computing in real closed fields containing transcendentals. This need is especially apparent when one considers cyber-physical systems [19,1,8]. In this setting, one needs to reason about ODEs which govern the continuous dynamics of a mixed discrete-continuous system. Solving for such trajectories gives rise to arithmetical constraints involving both the standard RCF operations and the constant $e$. If these ODEs occur in the context of aircraft maneuvers with angular positions, then one often needs to reason also with $\pi$. Similar combinations of RCF with transcendental constants arise in mainstream efforts in formalized mathematics, such as Thomas Hales's Flyspeck project, where many inequalities of this form await verification [15].

Finally, in addition to RCFs containing common transcendental constants, there is also a need for computing in RCFs containing *infinitesimals*. This stems from applications as well, albeit indirectly: as RCF is computationally infeasible, many researchers have focused on developing decision procedures for restricted but practically useful fragments of the theory. In two of the most useful fragments, the $\exists$ and $\exists\forall$ fragments, novel decision methods have been developed which rely on infinitesimals [5,14,7]. Some of these procedures, such as the singly exponential Grigor'ev-Vorobjnov $\exists$RCF method, are also of immense theoretical interest. However, many of them have never been implemented. The lack of a viable library for computing with real closed fields containing infinitesimals has been an impediment to this line of research. A robust library providing the computational substructure for reasoning in such real closed fields would remove a serious obstruction to work on nonlinear real arithmetic, allowing decision procedure researchers to focus on higher-level concerns, especially on novel decision methods which can then rely on this foundational library as a black-box.

In this paper, we present a library for computing in real closed fields containing computable transcendental, infinitesimal and algebraic elements. In particular, our library supports computing over real closed transcendental and infinitesimal extensions of the rational numbers. This is realized through the theory of *real closures*, a classical technique in real algebraic geometry which allows one to construct new real closed fields from arbitrary ordered fields.

[3] In fact, $\mathbb{R}_{alg}$ is the *prime model* of RCF, which means $\mathbb{R}_{alg}$ isomorphically embeds into every real closed field. In this sense, $\mathbb{R}_{alg}$ is the "smallest" real closed field.

Our main contribution is threefold:

1. We show how real closed fields containing computable transcendental, infinitesimal and algebraic elements can all be constructed and computed in using a single uniform method. This includes a novel approach to representing algebraic elements which combines a classical result in real algebraic geometry known as Thom's Lemma with modern interval arithmetic.
2. We develop the abstract algebra in a concrete algorithmic manner, and present several optimizations we have devised over the naive methods. These optimizations have been vital to making our library practical.
3. We describe how researchers can immediatly make use of our library, make it available for download (source included), and give examples designed to help the decision procedure researcher easily get started in this area.

*Related Work.* The combination of transcendental constants and infinitesimals with nonlinear real arithmetic has been explored in many ways.

In the MetiTarski prover for transcendental inequalities [1,8], of which the second author of this paper is a coauthor, transcendentals are approximated to a fixed accuracy using families of algebraically-expressible upper and lower bounds. This approach does not integrate the transcendentals into the RCF field arithmetic, but rather uses a combination of resolution theorem proving and the algebraic bounds to reduce the proof of an inequality involving transcendentals to a sequence of pure RCF decisions (taken over $\mathbb{R}_{alg}$). This is sufficient for many applications, especially for classes of engineering problems with non-tight inequalities, but fails when the bounds are insufficient, especially when equality reasoning is needed. Modifications to MetiTarski which make use of our library should allow for much more powerful reasoning with transcendental constants, especially with regards to proving tight inequalities and identities.

A very different incorporation of transcendental constants has been taken by the interval constraint propagation (ICP) community, as exemplified by the tools RealPaver [13], RSolver [20], iSat [10] and dReal [12]. Their methods are incomplete even for $\exists$RCF, but are extremely effective in some classes of applications. Unlike complete RCF decision methods, their foundations are not based upon real closed field arithmetic, but rather on computing with interval approximations to field values. This arithmetic can be very efficient and is always sound, but it comes at the cost of the so-called *interval dependency problem.* This commonly gives rise to the over-approximation of intervals and is a source of incompleteness. Our work combines interval approximations with exact techniques stemming from real algebraic geometry, yielding a library for exact computations suitable for building complete RCF decision methods.

Major research on reasoning with infinitesimals has been done both in the ACL2 [11] and Isabelle/HOL [9] proof assistants. These efforts have focused on a particular class of real closed fields, the Hyperreals, which provide the basis for the nonstandard analysis (NSA) approach to differential calculus. In NSA, the Hyperreals are used mainly to justify the consistency of the NSA axioms, especially the crucial Transfer Axiom, and one does not compute with their elements in the same way that one does when working over a countable, computable real closed field like $\mathbb{R}_{alg}$ during RCF decision procedures. In particular, the Hyperreals are a wildly uncomputable structure which depend on the choice of a non-principal ultrafilter over $\mathbb{N}$.[4] Thus, though both works involve real closed fields containing infinitesimals, our goals and approaches are very different.

The closest work to ours is that of Rioboo [21]. In this work, a library for computation in the real closure of a single infinitesimal extension of an ordered field was built within the computer algebra system Axiom. Though we share many goals and some high-level aspects of the approach (in particular, the explicit use of field towers as in the generic real closure method of Ligatsika, Rioboo and Roy [17]), our work is very different. First, the Rioboo infinitesimal methods revolve around a representation of algebraic elements using Puiseux series, a generalization of power series allowing fractional exponents. We take a completely different approach, combining Thom's Lemma with interval arithmetic. Second, Rioboo's library does not support extensions involving transcendental constants such as $\pi$ and $e$. Making use of intervals in the root representation is crucial to the way we treat transcendental constants. Finally, his implementation only supports the use of a single infinitesimal, though he discusses supporting multiple infinitesimals as future work. Our library supports multiple transcendentals, infinitesimals and algebraic elements simultaneously.

Finally, let us mention the groundbreaking work of Coste-Roy and its later refinements which showed how Thom's Lemma could be used algorithmically to represent algebraic elements over arbitrary real closed fields, even those containing infinitesimals [6,18,3]. We build our root representation upon this work, combining their derivative sign condition chains with interval arithmetic techniques, resulting in novel root isolation and sign determination methods.

## 2 Theoretical Background

An ordered field is a field equipped with a total order $\leq$ upon its elements s.t.

$$\forall xyz\,[(x \leq y \;\Rightarrow\; x+z \leq y+z) \quad\wedge\quad (0 \leq x \;\wedge\; 0 \leq y \;\Rightarrow\; 0 \leq x * y)].$$

Both $\mathbb{Q}$ and $\mathbb{R}$ with their respective orderings are ordered fields. The complexes cannot be made into an ordered field as neither $\sqrt{-1} \leq 0$ nor $0 \leq \sqrt{-1}$ is consistent with the ordered field axioms. A field is *real closed* iff it is an ordered field with two additional properties: First, positive elements are squares

$$\forall x\left(0 \leq x \Rightarrow \exists y(x = y^2)\right),$$

and second, all polynomials of odd degree have a root. This latter property is expressed using an axiom scheme, with one axiom for each $n \in \mathbb{N}$:

$$\forall a_0 a_1 \ldots a_{2n} \exists z\left(z^{2n+1} + a_{2n}z^{2n} + \ldots + a_1 z + a_0 = 0\right).$$

[4] Note that it is consistent with ZF that *no* non-principal ultrafilters on $\mathbb{N}$ exist. Thus, to build a Hyperreal field, an uncomputable choice principle is needed. The wild uncomputability is contributed both from $\mathbb{R}$ and the ultrapower construction.

Observe that $\mathbb{R}$ and $\mathbb{R}_{alg}$ are real closed but $\mathbb{Q}$ is not.

Let $\mathbb{K}_1, \mathbb{K}_2$ be fields s.t.

$$\mathbb{K}_1 = \langle K_1, +_{\mathbb{K}_1}, *_{\mathbb{K}_1}, -_{\mathbb{K}_1}, 0_{\mathbb{K}_1}, 1_{\mathbb{K}_1} \rangle \quad \text{and} \quad \mathbb{K}_2 = \langle K_2, +_{\mathbb{K}_2}, *_{\mathbb{K}_2}, -_{\mathbb{K}_2}, 0_{\mathbb{K}_2}, 1_{\mathbb{K}_2} \rangle$$

and $K_1 \subset K_2$. If the function symbols of $\mathbb{K}_1$ and $\mathbb{K}_2$ agree over all elements of $K_1$ (i.e., $0_{\mathbb{K}_1} = 0_{\mathbb{K}_2}$, $1_{\mathbb{K}_1} = 1_{\mathbb{K}_2}$, $\forall x, y \in K_1(x +_{\mathbb{K}_1} y = x +_{\mathbb{K}_2} y)$, and so on), then we say that $\mathbb{K}_1$ is a *subfield* of $\mathbb{K}_2$, that $\mathbb{K}_2$ is an *extension field* of $\mathbb{K}_1$, and that $\mathbb{K}_2/\mathbb{K}_1$ (pronounced "$\mathbb{K}_2$ over $\mathbb{K}_1$") is a *field extension.*[5] When no confusion should arise, we use $\mathbb{K}_1 \subset \mathbb{K}_2$ to indicate that $\mathbb{K}_1$ is a subfield of $\mathbb{K}_2$.

If $\mathbb{K}_1 \subset \mathbb{K}_2$ and $S \subset K_2$, then $\mathbb{K}_1(S)$ denotes the smallest subfield of $\mathbb{K}_2$ extending $\mathbb{K}_1$ and containing $S$. If $\varsigma \in K_2$ then $\mathbb{K}_1(\varsigma)$ denotes $\mathbb{K}_1(\{\varsigma\})$. We say that $\mathbb{K}_1(S)$ is the result of *adjoining* the elements of $S$ *to* $\mathbb{K}_1$. An extension of the form $\mathbb{K}_1(\varsigma)/\mathbb{K}_1$ is called *simple.* If $\varsigma$ is the root of a polynomial in $\mathbb{K}_1[x]$, then the extension $\mathbb{K}_1(\varsigma)/\mathbb{K}_1$ is *algebraic*, and $\varsigma$ is *algebraic over* $\mathbb{K}_1$. Otherwise, the extension is *transcendental*, and $\varsigma$ is *transcendental over* $\mathbb{K}$. We can iterate the process of taking simple extensions so as to obtain non-simple ones, i.e., $\mathbb{K} \subset \mathbb{K}(\varsigma_1) \subset (\mathbb{K}(\varsigma_1))(\varsigma_2)$. We write $\mathbb{K}(\varsigma_1, \varsigma_2)$ for $(\mathbb{K}(\varsigma_1))(\varsigma_2)$. In this case, $\mathbb{K}(\varsigma_1, \varsigma_2)/\mathbb{K}(\varsigma_1)$ is simple, but $\mathbb{K}(\varsigma_1, \varsigma_2)/\mathbb{K}$ is not. A (finite or infinite) sequence of extensions $\mathbb{K}_1 \subset \mathbb{K}_2 \subset \ldots \subset \mathbb{K}_n \subset \ldots$ is called a *field tower.*

*Example 1.* $\mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$ is a field tower, and from it we can deduce that $\mathbb{Q}$ and $\mathbb{R}$ are both subfields of $\mathbb{C}$. However, the extension $\mathbb{C}/\mathbb{Q}$ is very different than the extension $\mathbb{C}/\mathbb{R}$. In particular, to obtain $\mathbb{C}$ from $\mathbb{Q}$ one must adjoin uncountably many elements, while to obtain $\mathbb{C}$ from $\mathbb{R}$ one need only adjoin $\sqrt{-1}$.

Let us now examine the process of field adjunction. Consider the field $\mathbb{Q}(\sqrt{2})$ resulting from adjoining $\sqrt{2}$ to $\mathbb{Q}$. Since $\sqrt{2}$ is a root of the polynomial $x^2 - 2 \in \mathbb{Q}[x]$, the extension $\mathbb{Q}(\sqrt{2})/\mathbb{Q}$ is algebraic. How can we build this field extension? Since a field is closed under its arithmetic operations, we know that as $\sqrt{2} \in \mathbb{Q}(\sqrt{2})$, then, for instance, $1/\sqrt{2}$, $23/\sqrt{2} + 1/2$, and $11/(3 * \sqrt{2})$ all must be in $\mathbb{Q}(\sqrt{2})$ as well. This suggests that we consider elements of $\mathbb{Q}(\sqrt{2})$ to be formal ratios of elements of the polynomial ring $\mathbb{Q}[\sqrt{2}]$, where $\sqrt{2}$ is taken to be a symbolic indeterminate subject to the constraint that $\sqrt{2} * \sqrt{2} = 2$. In fact, the situation is more subtle than this, and we describe it algorithmically in Sec. 3.3.

For transcendental extensions, we also make use of ratios of polynomials. If $\tau$ is transcendental over $\mathbb{K}$, then $\mathbb{K}(\tau)$ is isomorphic to $\mathbb{K}(x)$, where $\mathbb{K}(x)$ is the field of fractions of the polynomial ring $\mathbb{K}[x]$, i.e., the field of formal rational functions drawn from $\{p(x)/q(x) \mid p(x), q(x) \neq 0 \in \mathbb{K}[x],\ gcd(p(x), q(x)) = 1\}$. This isomorphism holds because as $\tau$ is transcendental, it has no nontrivial algebraic relationships with the elements of $\mathbb{K}$.

Let us return now to ordered fields. If $\mathbb{K}$ is an ordered field, then the subfields and extension fields of $\mathbb{K}$ we are interested in are those which respect $\mathbb{K}$'s ordering relation, i.e., $\mathbb{K}_1 \subset \mathbb{K} \subset \mathbb{K}_2$ implies $\leq_{\mathbb{K}_1} \subset \leq_{\mathbb{K}} \subset \leq_{\mathbb{K}_2}$. Any extension or subfield of an ordered field that we consider in this paper shall be of this form.

[5] Note that this field extension notation using "/" is purely formal, and does not imply a quotient structure or anything along those lines. See Ex. 1 for an example use.

If $\mathbb{K}$ is an ordered field, then it is possible to adjoin an *infinitesimal* element $\epsilon$ to $\mathbb{K}$ by treating $\mathbb{K}(\epsilon)/\mathbb{K}$ as a transcendental extension whose ordering extends the ordering of $\mathbb{K}$ subject to the following constraint:

$$\epsilon > 0 \ \wedge \ \forall k \in \mathbb{K} \ (k > 0 \ \Rightarrow \ \epsilon < k)\,.$$

We describe algorithmically the orderings in field extensions in Sec. 3.

Finally, let us turn to *real closures*. It is a fundamental result in real algebraic geometry that *every* ordered field $\mathbb{K}$ possesses a unique minimal algebraic real closed extension. This field is called the *real closure of* $\mathbb{K}$ and is written $\widetilde{\mathbb{K}}$. For example, $\widetilde{\mathbb{Q}} = \mathbb{R}_{alg}$ and $\widetilde{\mathbb{R}} = \mathbb{R}$. In this work, we are concerned with computing in real closed fields $\widetilde{\mathbb{K}}$, where $\mathbb{K}$ is obtained from $\mathbb{Q}$ by finitely many transcendental and infinitesimal extensions. In passing from $\mathbb{K}$ to $\widetilde{\mathbb{K}}$, a countably infinite collection of algebraic elements will be adjoined. However, to compute over $\widetilde{\mathbb{K}}$, we only need, at any given time, an extension of $\mathbb{K}$ by finitely many algebraic elements. $\widetilde{\mathbb{K}}$ is obtained "in the limit."[6]

To provide general support for nonlinear real arithmetic decision methods, including those methods requiring transcendentals and infinitesimals, we must provide all of the basic ordered field operations (arithmetic and ordering) over arbitrary subfields of $\widetilde{\mathbb{K}}$. The most challenging aspect occurs over proper algebraic extensions of $\mathbb{K}$, as this requires reasoning about the roots $\alpha \in \widetilde{\mathbb{K}}$ of polynomials $p \in \mathbb{K}[x]$. If $\mathbb{K}$ contains no infinitesimal elements, then these roots can always be isolated using intervals with rational endpoints. However, if $\mathbb{K}$ contains infinitesimal elements, then the situation is considerably more complicated.

## 3 Implementing Field Extensions

Our package implements towers of extensions beginning with $\mathbb{Q}$. Rationals are implemented as a pair of multi-precision integers. We support three kinds of extensions: transcendental, infinitesimal and algebraic. In our library, there is always a linear order $\prec$ between fields in a given tower. Consider a tower $\mathbb{Q} \subset \mathbb{Q}(\varsigma_1) \subset \ldots \subset \mathbb{Q}(\varsigma_1, \varsigma_2, \ldots, \varsigma_k)$. Then, it will hold that $\mathbb{Q}(\varsigma_1, \ldots, \varsigma_i) \prec \mathbb{Q}(\varsigma_1, \ldots, \varsigma_{i+1})$. Moreover, our field extensions must be constructed in a sequence s.t. transcendental extensions $\prec$ infinitesimal extensions $\prec$ algebraic extensions.

Abstractly, we can view a field extension as a mapping that, given an implementation for the operations of an ordered field $\mathbb{K}$, "lifts" these operations to the field extension $\mathbb{K}(\varsigma)$. In this Section, we describe how we implement these operations for each kind of extension. Let us fix some preliminaries.

Let $B$ be the set of *binary rationals* (also known as *dyadic rationals*). $B$ consists of rationals of the form $a/2^k$. $B$ does not form a field, but it forms a ring and is closed under division by 2. The implementation of addition and multiplication for binary rationals is more efficient than that for rational numbers. Moreover,

[6] The situation is similar with packages for rational arithmetic. These packages allow one to compute over $\mathbb{Q}$ by constructing rationals "on demand" as they are needed. Of course, at any given time, only finitely many rationals have been constructed.

binary rationals can be normalized using just bit-shifting operations, instead of expensive integer gcd and division. Finally, division can be approximated to any precision using the same approach used in floating point arithmetic.[7] We say an interval of the form $(l, u)$ is a $B_\infty$-interval if $l, u \in (B \cup \{-\infty, \infty\})$ and $0 \notin (l, u)$. We produce interval approximations for all elements of our extension fields. For every non-zero element $a$, $\mathsf{interval}(a)$ is a $B_\infty$-interval $(l, u)$ containing $a$. Moreover, if $a$ is not constructed using infinitesimals, then $l, u \in B$, and we then provide a procedure to refine the size of $(l, u)$ to any desired precision. As the associated interval for a non-zero element never contains zero, the sign of any element can be read off from its associated interval. This allows us to decide comparisons between field elements by reducing the comparison of $a$ and $b$ to the sign of $a - b$.

### 3.1 Transcendental Extensions

At the bottom of our field towers, we support computable transcendental reals such as $\pi$ and $e$. For adjoining a transcendental element $\tau$, we require the user to provide a procedure $\mathsf{approximate}(\tau)$ s.t. given any $i \in \mathbb{N}$, $\mathsf{approximate}(\tau)(i)$ returns an open interval $(l, u)$ s.t. $\tau \in (l, u)$ and $l, u \in B$. Moreover, the approximation must converge in the following sense: Let $\mathsf{width}(l, u) = u - l$. Then, for any $k$ there must exist an $i$ s.t. $\mathsf{width}(\mathsf{approximate}(\tau)(i)) \leq 1/2^k$. In our prototype, we provide implementations of $\mathsf{approximate}(\pi)$ and $\mathsf{approximate}(e)$.

When extending a field $\mathbb{K}$ with an irrational number $\tau$, it is the user's responsibility to guarantee that $\tau$ is indeed transcendental with respect to $\mathbb{K}$. If this is not the case, then our implementation may not terminate when executing the sign determination algorithm in $\mathbb{K}(\tau)$. Note that transcendence is always relative to the field being extended, e.g., $\pi$ and $\sqrt{\pi}$ are both transcendental over $\mathbb{Q}$, but $\sqrt{\pi}$ is not transcendental over $\mathbb{Q}(\pi)$ as it is a root of $x^2 - \pi \in (\mathbb{Q}(\pi))[x]$.

As discussed in Sec. 2, we represent the elements of $\mathbb{K}(\tau)$ as formal rational functions $p(\tau)/q(\tau)$, with $\tau$ treated as an indeterminate. Since $\tau$ is transcendental over $\mathbb{K}$, it is easy to check that $q(\tau)$ is not the zero polynomial by simply verifying that $q(\tau)$ is not identically zero using standard polynomial arithmetic over $\mathbb{K}[\tau]$.

The field operations for $\mathbb{K}(\tau)$ are based on polynomial arithmetic build upon the arithmetic operations of the field arithmetic for $\mathbb{K}$. We use the standard normal form of rational functions where the polynomial gcd of the numerator and denominator is one, and the denominator is a monic polynomial. In this representation, two values are equal iff they have the same normal form. The polynomial gcd is implemented using the standard Euclidean algorithm based on the polynomial remainder algorithm because it can be easily implemented for polynomials in $\mathbb{K}[x]$ when $\mathbb{K}$ is a computable field.

*Example 2.* Given $\frac{1}{2}\pi$, $\frac{1}{\pi+1} \in \mathbb{Q}(\pi)$, their sum is equal to $\frac{\frac{1}{2}\pi^2+\frac{1}{2}\pi+1}{\pi+1}$.

[7] Note that we can view binary rationals as arbitrary precision floating point numbers with negative exponents.

The approximating interval of size $1/2^k$ for a non-zero element $a = p(\tau)/q(\tau)$ is computed using interval arithmetic. Our procedure keeps refining the interval approximations for $\tau$ and the coefficients of $p(\tau)$ and $q(\tau)$ until the resulting interval for $a$ does not contain zero and has width $\leq 1/2^k$. It is easy to see that this procedure always terminates when $\tau$ is transcendental over $\mathbb{K}$.

Since the approximating interval does not contain zero, we can use it to infer the sign of any element of $\mathbb{K}(\tau)$. Moreover, we can decide whether $a < b$ by computing the sign of $a - b$. For efficiency in the actual implementation, we first try to compare $a$ and $b$ using their approximating intervals. If the intervals do not overlap, then we can answer the query by simply comparing the lower and upper bounds of the intervals. Otherwise, we refine the approximating intervals until they do not overlap or their size is smaller than a user-provided threshold. If the threshold is reached, then we compute the sign of $a - b$.

### 3.2 Infinitesimal Extensions

An infinitesimal extension $\mathbb{K}(\epsilon)/\mathbb{K}$ adjoins a new infinitesimal to $\mathbb{K}$. Our implementation supports an arbitrary number of infinitesimals. Each new infinitesimal is infinitely smaller than any previously added infinitesimal. Note that every infinitesimal is also transcendental with respect to $\mathbb{K}$. Because of this, we also use formal rational functions to represent the elements of $\mathbb{K}(\epsilon)$. It then suffices to present the interval machinery we use to compute the ordering relation.

Note that $1/\epsilon$ is larger than any element of $\mathbb{K}$. We say $1/\epsilon$ is an *infinite value.* The initial interval approximation for $\epsilon$ is the interval $(0, 1/2^{k_\epsilon})$, where $k_\epsilon \in \mathbb{N}$ is a user-specified parameter. Thus, the initial interval approximation for $1/\epsilon$ is $(2^{k_\epsilon}, \infty)$. We say intervals of the form $(-\infty, u)$ and $(l, \infty)$ are *non-refinable.* Only elements constructed using infinitesimals may have non-refinable intervals.

Given a non-zero polynomial $p(\epsilon)$ of the form $a_n\epsilon^n + \ldots + a_1\epsilon + a_0$ with $a_0 \neq 0$, the approximating interval of width $1/2^k$ for $p(\epsilon)$ is the approximating interval of width $1/2^k$ for $a_0$. If $a_0 = 0$, we say $p(\epsilon)$ is *infinitesimal.* Consider $p(\epsilon)$ infinitesimal and let $a_i$ be the first non-zero coefficient. If $a_i$ is negative, then the approximating interval for $p(\epsilon)$ is $(-1/2^{k_\epsilon}, 0)$, otherwise it is $(0, 1/2^{k_\epsilon})$.

Let $k \in \mathbb{K}(\epsilon)$ s.t. $k = (a_n\epsilon^n + \ldots + a_1\epsilon + a_0)/(\epsilon^m + \ldots + b_1\epsilon + b_0) \neq 0$. If $a_0 \neq 0$ and $b_0 \neq 0$, then an approximating interval of size $1/2^k$ is computed by refining the intervals for $a_0$ and $b_0$ until the desired precision is reached. If either $a_0$ or $b_0$ is non-refineable, then $k$ is also non-refinable. Note that we never have $a_0 = 0$ *and* $b_0 = 0$, since in this case the numerator and denominator can be simplified by dividing them by $\epsilon$. Thus, if $a_0 = 0$, we must have $b_0 \neq 0$, and $k$ is an infinitesimal value. If $b_0 = 0$, then $a_0 \neq 0$, and $k$ is an infinite value. Note that even when we cannot refine an approximating interval for $k$, we can still compute a $B_\infty$-interval containing $k$.

### 3.3 Algebraic Extensions

At the top of our towers sit algebraic extensions. Recall that an algebraic extension $\mathbb{K}(\alpha)/\mathbb{K}$ is obtained by adjoining to $\mathbb{K}$ a root $\alpha$ of a polynomial $p \in \mathbb{K}[x]$.

Given such an extension, we call $p$ the *defining polynomial of* $\alpha$. Note that the algebraic extensions of $\mathbb{K}$ that we support are always subfields of $\widetilde{\mathbb{K}}$. To represent elements of $\mathbb{K}(\alpha)$, we need to be able to compute with roots of $p$ which reside in the real closure $\widetilde{\mathbb{K}}$. *Thom's Lemma* is a classical result in real algebraic geometry which guarantees that we can always distinguish the roots of a polynomial over a real closed field (even those containing infinitesimals) using only the signs of its derivatives [6]. We base our representation upon this fact, and introduce a number of enhancements. Due to space limitations, we are forced to only present the most salient aspects of how we compute in algebraic extensions.[8]

*Sign assignments* Given a set of polynomials $Q$, a *sign assignment* $S$ is a mapping from $Q$ to $\{-1, 0, 1\}$. To improve readability, we shall represent $S$ using sets of atoms. For example, $\{q_1 \mapsto -1, q_2 \mapsto 0\}$ is represented as $\{q_1 < 0, q_2 = 0\}$. We identify a root $\alpha$ of a polynomial $p$ using a pair consisting of an open $B_\infty$-interval and a sign assignment for a subset of the derivatives of $p$. The sign assignment stores the sign of these derivatives at $\alpha$. If $\mathbb{K}$ does not contain infinitesimal extensions, then the sign assignments are not necessary for distinguishing roots. For example, $\sqrt{2}$ can be encoded as $(x^2 - 2, (1, 2), \{\})$, i.e., as the only root of $x^2 - 2$ within $(1, 2)$ satisfying the empty sign assignment.

*Example 3.* Let $\mathbb{Q}(\epsilon)/\mathbb{Q}$ be an infinitesimal extension. The three roots of the polynomial $\epsilon^2x^5 - \epsilon x^3 - \epsilon x^2 + 1 \in (\mathbb{Q}(\epsilon))[x]$ can be encoded as

$$\begin{array}{ll}
(\epsilon^2x^5 - \epsilon x^3 - \epsilon x^2 + 1, (-\infty, 0), & \{\}) \\
(\epsilon^2x^5 - \epsilon x^3 - \epsilon x^2 + 1, (0, \infty), & \{60\epsilon^2x^2 - 6\epsilon > 0\}) \\
(\epsilon^2x^5 - \epsilon x^3 - \epsilon x^2 + 1, (0, \infty), & \{60\epsilon^2x^2 - 6\epsilon < 0\})
\end{array}$$

We need a sign assignment to distinguish the two positive roots because they are bigger than any real number, and cannot be isolated using an interval. In the example above, the sign of the third derivative was used to distinguish between these two roots. Recall that *Thom's Lemma* guarantees that we can always distinguish the roots of a polynomial over an RCF using only the signs of its derivatives. Since the interval contains only two roots that need to be discriminated, we only need to find *one* derivative that has a different sign for each root. We use the third derivative, because it is the lowest degree derivative which discriminates the two roots. For example, the fourth derivative $120\epsilon^2x$ is not used because it is positive for both roots. Later, we show how the *Sign Determination Algorithm* is used to compute the signs of these derivatives. This example also demonstrates that we often need only a proper subset of the derivatives.

*Square-free polynomials* We say a polynomial $p \in \mathbb{K}[x]$ is *minimal* if $p$ does not contain a non-trivial factor. A polynomial is *square-free* if it is does not have roots with multiplicity greater than 1. Note that given a square-free polynomial $p$ and an interval $(a, b)$ that contains only one root of $p$ s.t. $a$ and $b$ are not roots of $p$, it follows that $\mathsf{sign}(p(a)) = -\mathsf{sign}(p(b))$, where $\mathsf{sign}$ is a function that

[8] Further details shall be in an expanded version. See Sec. 4 for a source code URL.

maps a value into the set $\{-1, 0, 1\}$. Given a polynomial $p$, we define the function $\mathsf{sqf}(p) = p/\mathsf{gcd}(p, p')$, where $p'$ is the first derivative of $p$. It is well-known that $\mathsf{sqf}(p)$ is square-free with the same roots as $p$. WLOG, let us now consider polynomials of the form $a_n x^n + \ldots + a_0$ s.t. $a_0 \neq 0$. In our representation, we do not require defining polynomials to be minimal because polynomial factorization over extension fields is an expensive operation. Instead, we use square-free polynomials because they are faster to be computed, and more importantly, as the Intermediate Value Theorem holds over every real closed field, we can refine the (refinable) interval $(a, b)$ containing a single root $\alpha$ by just computing the sign of $p$ at the midpoint $m$: If $\mathsf{sign}(p(a)) = \mathsf{sign}(p(m))$, then the new interval is $(m, b)$. If $\mathsf{sign}(p(b)) = \mathsf{sign}(p(m))$, then the new interval is $(a, m)$. In the very unlikely case when $\mathsf{sign}(p(m)) = 0$, then $\alpha$ is actually the binary rational $m$.

*Polynomial remainder sequences and Sturm-Tarski* Let $\mathsf{quo}(q, p)$ and $\mathsf{rem}(q, p)$ denote the polynomial quotient and remainder (resp.) of $q, p \in \mathbb{K}[x]$, i.e., $q = \mathsf{quo}(q, p) \cdot p + \mathsf{rem}(q, p)$ s.t. $\mathsf{deg}(\mathsf{rem}(q, p)) < \mathsf{deg}(p)$, where $\mathsf{deg}(p)$ is the degree of $p$. If $p$ is the defining polynomial for $\alpha$, then $p(\alpha) = 0$ and consequently $q(\alpha) = \mathsf{rem}(q, p)(\alpha)$. This allows us to use polynomial remainders to simplify any $q(\alpha)$. The *signed polynomial remainder* $\mathsf{srem}(q, p)$ is defined as $-\mathsf{rem}(q, p)$. A Sturm polynomial sequence $[s_1; s_2; \ldots; s_k]$ for polynomials $p$ and $q$ is defined inductively as $s_1 = p$, $s_2 = q$, $s_i = \mathsf{srem}(s_{i-2}, s_{i-1})$, where $\mathsf{srem}(s_{k-1}, s_k) = 0$. We use $\mathsf{sturm}(p, q)$ to denote the Sturm polynomial sequence for $p$ and $q$. Given a sequence $S$ of polynomials in $\mathbb{K}[x]$, we use $\mathsf{sv}(S, a)$ for the number of sign changes (ignoring zeroes) in the sequence when each polynomial is evaluated at $a$. For example, $\mathsf{sv}([2 + x^2 + x^3;\ 2x + 3x^2;\ -18 + 2x;\ -1], 0) = 1$, since there is only one sign variation in the sequence evaluated at 0. We use $\mathsf{pos}(q, p, a, b)$, $\mathsf{neg}(q, p, a, b)$ and $\mathsf{zero}(q, p, a, b)$ to denote the number of roots $\beta$ of $p$ s.t. $\beta \in (a, b)$ and $q(\beta)$ is positive, negative and zero respectively. The *Sturm-Tarski Theorem* states that given a polynomial sequence $S = \mathsf{sturm}(p, q \cdot p')$, it holds that[9]

$$\mathsf{sv}(S, a) - \mathsf{sv}(S, b) = \mathsf{pos}(q, p, a, b) - \mathsf{neg}(q, p, a, b).$$

Following Basu-Pollack-Roy [3], we define a *Tarski Query* $\mathsf{TaQ}(q, p; a, b)$ as

$$\mathsf{TaQ}(q, p; a, b) = \mathsf{sv}(\mathsf{sturm}(p, q \cdot p'), a) - \mathsf{sv}(\mathsf{sturm}(p, q \cdot p'), b)$$

and remark that

$$\begin{aligned}
\mathsf{TaQ}(1, p; a, b) &= \mathsf{zero}(q, p, a, b) + \mathsf{pos}(q, p, a, b) + \mathsf{neg}(q, p, a, b),\\
\mathsf{TaQ}(q, p; a, b) &= \mathsf{pos}(q, p, a, b) - \mathsf{neg}(q, p, a, b),\\
\mathsf{TaQ}(q^2, p; a, b) &= \mathsf{pos}(q, p, a, b) + \mathsf{neg}(q, p, a, b).
\end{aligned}$$

Moreover, $\mathsf{TaQ}(1, p; a, b)$ is the number of roots of $p$ in the interval $(a, b)$. If $\alpha$ is the only root of $p$ in the interval $(a, b)$, then the sign of $q(\alpha)$ can be determined using $\mathsf{TaQ}(q, p; a, b)$.

[9] The Sturm-Tarski Theorem is actually for half-open intervals of the form $(a, b]$. WLOG, we assume that for any root $\alpha$ encoded as $(p, (a, b), S)$, $b$ is not a root of $p$. If it is, we encode $\alpha$ as $(p/(x - b), (a, b), S)$ instead.

*Sign determination* Tarski Queries are also used to implement the *Sign Determination Algorithm* [3]. Given a set of polynomials $Q = \{q_1, \ldots, q_k\}$, $\mathsf{signdet}$(Q, p, a, b) returns the feasible sign assignments of $Q$ at the roots of $p$ in the interval $(a, b)$. Actually, it computes more than that: for each sign assignments $S$, it returns the number of roots of $p$ in $(a, b)$ that satisfy $S$. For example, for $Q = \{q\}$, $\mathsf{signdet}$ can compute the feasible sign assignments by computing $\mathsf{TaQ}(1, p; a, b)$, $\mathsf{TaQ}(q, p; a, b)$ and $\mathsf{TaQ}(q^2, p; a, b)$ and solving the system of equations above. For $Q = \{q_1, q_2\}$, in the worst case, we have to compute $\mathsf{TaQ}(h, p; a, b)$ for each $h$ in the set $\{1, q_2, q_2^2, q_1, q_1q_2, q_1q_2^2, q_1^2, q_1^2q_2, q_1^2q_2^2\}$. In general for a set $Q = \{q_1, \ldots, q_k\}$, $\mathsf{signdet}$ will, in the worst case, have to compute $3^k$ Tarski Queries for polynomials of the form $\prod_{q\in Q, i\in\{0,1,2\}} q^i$, and solve a system of $3^k$ equations. We implement a more efficient $\mathsf{signdet}$ of Ben-Or *et al.* [4].

Now, assume that $\alpha$ is encoded as $(p, (a, b), S)$, and we want to determine the sign of $q(\alpha)$. We can decide that by computing $R = \mathsf{signdet}(\mathsf{poly}(S) \cup \{q\}, p, a, b)$, where $\mathsf{poly}(S)$ is the set of polynomials occurring in $S$. Then,

$$\begin{array}{l} \text{if } S \cup \{q = 0\} \in R \text{ then } q(\alpha) = 0, \\ \text{if } S \cup \{q > 0\} \in R \text{ then } q(\alpha) > 0, \\ \text{if } S \cup \{q < 0\} \in R \text{ then } q(\alpha) < 0. \end{array}$$

We know that one and only one of the cases above can be true because $p$ has only one root in the interval $(a, b)$ satisfying the sign conditions $S$.

To compute an upper-bound for the positive roots of $(\sum_{i=0}^{n} a_i x^i) \in \mathbb{K}[x]$, we use Knuth's bound $2(\mathsf{max}\{ \sqrt[k]{(-a_{n-k}/a_n)} \mid 1 \leq k \leq n, a_{n-k} < 0\})$. As the $a_i$ may be neither integer nor rational values, we estimate $(-a_{n-k}/a_n)$ using the approximating intervals for $a_{n-k}$ and $a_n$. Let $s$ be the upper bound of the resulting interval. If the upper-bound for $s$ is $\infty$, then so is the sought upper-bound. Otherwise, we compute the least integer $j$ s.t. $s \leq 2^j$. The value $j$ can be easily computed based on the bit-wise $\mathsf{log}_2$ operation for integers. Finally, we approximate the $k$th root as $2^{\frac{j}{k}+1}$. Note that if $\mathbb{K}$ does not contain infinitesimals, then the computed upper-bound is a binary rational of the form $2^m$. The lower-bound for a positive root is computed by computing the upper-bound for $x^n p(1/x)$. For negative roots, we compute the bounds for the positive roots of $p(-x)$.

*Clean representations* We represent elements of $\mathbb{K}(\alpha)$ as polynomials $q(\alpha)$. We define inductively the predicate $\mathsf{clean}(a)$. If $a \in \mathbb{Q}$, then $\mathsf{clean}(a)$ holds if $a$ is an integer. If $a$ is an element of a transcendental or infinitesimal extension $\mathbb{K}(\varsigma)$, then $\mathsf{clean}(a)$ holds if $a$ is of the form $p(\varsigma)/1$ and for all coefficients $c$ of $p(\varsigma)$, $\mathsf{clean}(c)$ holds. Similarly, if $a$ is an element of an algebraic extension, then $\mathsf{clean}(a)$ holds if $a$ is represented by a polynomial with clean coefficients. When $\mathsf{clean}(a)$ holds, we say $a$ is *clean*. In our experiments, we observed that minimizing the use of $\mathsf{gcd}$ (especially with non-clean elements) is by far the most important optimization. Many operations with clean elements produce clean elements, and consequently do not require expensive normalization operations based on $\mathsf{gcd}$.

Let $a$ and $b$ be clean elements of transcendental or infinitesimal extensions. Then, $a + b$, $-a$ and $a \cdot b$ are also clean. We recall that for algebraic extensions

$\mathbb{K}(\alpha)$, an element $a$ is represented as a polynomial $q(\alpha)$, and this polynomial can be (optionally) simplified to $\mathsf{rem}(q, p)$, where $p$ is the defining polynomial for $\alpha$. If $p$ is monic with clean coefficients and $a$ is clean before applying the simplification, it will remain clean after applying it. Unfortunately, this is not the case for non-monic polynomials. For example, $\mathsf{rem}(\alpha^3+1, 3\alpha^2-1) = 1+(2/3)\alpha$.

To minimize the generation of non-clean elements, we generate Sturm sequences using polynomial pseudo-remainders. We use $\mathsf{pquo}(q, p)$ and $\mathsf{prem}(q, p)$ to denote the polynomial pseudo-quotient and pseudo-remainder of $q, p \in \mathbb{K}[x]$, and remark that $l^d q = \mathsf{pquo}(q, p) \cdot p + \mathsf{prem}(q, p)$, where $l$ is the leading coefficient of $p$, and $d$ is the number of iterations used to compute $\mathsf{pquo}(q, p)$ and $\mathsf{prem}(q, p)$. The signed pseudo remainder is defined as

$$\mathsf{sprem}(q, p) = \begin{cases} \mathsf{prem}(q, p), & \text{if } l < 0 \land d \text{ is odd} \\ -\mathsf{prem}(q, p), & \text{otherwise} \end{cases}$$

The main motivation for this is that for any element $a$, $\mathsf{sign}(\mathsf{sprem}(q, p)(a)) = \mathsf{sign}(\mathsf{srem}(q, p)(a))$. Because for Tarski Queries only the number of sign alternations matter, we can use $\mathsf{sprem}$ instead of $\mathsf{srem}$ when generating Sturm sequences.

Given a polynomial $p$ with clean coefficients, if we disable the algebraic normalizations using non-monic defining polynomials, and compute Sturm sequences using $\mathsf{sprem}$, then all elements in the generated sequence are clean. With this approach, we observed a dramatic performance improvement (cf. Sec. 4).

Now, let us show how we represent elements of $\mathbb{K}(\alpha)$ as polynomials even when the defining polynomial $p$ for $\alpha$ is not minimal. When $p$ is minimal, given a non-zero element $a$ represented using a polynomial $q(\alpha)$, we can represent $1/a$ using a polynomial $h(\alpha)$. Let $r$ be $\mathsf{rem}(q, p)$. As $p$ is minimal, $\gcd(p, r) = 1$. Then, using the extended gcd algorithm we can compute polynomials $h$ and $g$ such that $g \cdot p + h \cdot r = 1$ Since, $p(\alpha) = 0$ and $r(\alpha) = q(\alpha)$, we have $h(\alpha) \cdot q(\alpha) = 1$. If $p$ is not minimal, the $\gcd(p, r)$ may be different from 1, with $h$ the zero polynomial. To cope with this problem, we simply replace the defining polynomial $p$ for $\alpha$ with $(p/\gcd(p, r))$ whenever $\gcd(p, r) \neq 1$.

*Root isolation* Finally, we summarize all the steps used in our root isolation procedure for polynomials $p \in \mathbb{K}[x]$. First, we make sure that 0 is not a root of $p$, $p$ is square-free and has clean coefficients. Then, we estimate the lower and upper bounds for positive and negative roots. WLOG, we focus on the positive case. If the upper-bound is $\infty$, we use the sign determination procedure for distinguishing the positive roots using the signs of the derivatives of $p$. If the upper bound is not $\infty$, we try to isolate the roots using interval bisection and binary search. If $\mathbb{K}$ does not depend on infinitesimals, the procedure always terminates. If $\mathbb{K}$ depends on infinitesimals, to guarantee termination, we interrupt the binary search if the size of the interval in a branch is smaller than a user provided parameter, and switch to the approach based on sign determination. Note that if $p$ does not depend on infinitesimal values, then more efficient root isolation methods can be used [23].

## 4 Examples

In this section, we present a small set of examples using our package. Our library was implemented as a module in the Z3 theorem prover[10], and we provide a C API and Python bindings.

*Introductory examples.* We demonstrate the basic capabilities of our package using the Python bindings. A polynomial is described as a list of coefficients. `MkRoots` returns the roots of a polynomial as a list. In the following command we consider the roots of $x^2 - 2$. In Python, `**` is the power operator.

```
msqrt2, sqrt2 = MkRoots([-2, 0, 1])
print(sqrt2)
>> root(x^2 + -2, (0, +oo), {})
print(1/sqrt2)
>> 1/2*root(x^2 + -2, (0, +oo), {})
print(sqrt2**2 == 2)
>> True
print(sqrt2.decimal(10))
>> 1.4142135623?
print(sqrt2**3 + 1)
>> 2*root(x^2 + -2, (0, +oo), {}) + 1
```

The procedure `MkInfinitesimal` creates a new infinitesimal extension, while `Pi` and `E` return $\pi$ and $e$ respectively. In the following example, we extract the first (and only) root of the polynomial $x^3 + \epsilon\, x^2 + (\sqrt{2} + \pi)\, x - \pi$.

```
eps = MkInfinitesimal("eps")
pi  = Pi()
r   = MkRoots([-pi, sqrt2 + pi, eps, 1])[0]
print(r)
>> root(x^3 + eps*x^2 + (root(x^2 + -2, (0, +oo), {}) + pi)*x +
        -1*pi, (0, +oo), {})
print(r.decimal(10))
>> 0.6337173142?
```

Now, we show basic computations with infinitesimals and transcendentals. First, we compare $2 + 2\pi + \pi^2 - 2\epsilon - 2\pi\epsilon + \epsilon^2 < 2 + 2\pi + \pi^2$, and then we compare $\epsilon$ and $\sqrt[3]{\epsilon}$.

```
print(2 + 2*pi + pi**2 - 2*eps - 2*pi*eps + eps**2 < 2 + 2*pi + pi**2)
>> True
eps3 = MkRoots([-eps, 0, 0, 1])[0]
print(eps3 > eps)
```

[10] The code is available at `http://z3.codeplex.com/wikipage?title=CADE24`. Experiments were done on an Intel Core i7-2620 2.7Ghz CPU with 8Gb RAM, and the package was compiled using the GMP multi-precision library.

```
>> True
print(1/eps > 10000000000000000000000000000)
>> True
```

The examples above are stored in the file basic.py.

*MetiTarski.* MetiTarski uses the nlsat [16] nonlinear solver in Z3. Real algebraic number computations are often a bottleneck for nlsat, and were implemented using the textbook approach of polynomials with integer coefficients and an interval with binary rational endpoints. The following example was extracted from a MetiTarski/Z3 execution trace where nlsat times out after 30 min. In this example, nlsat assigns the first root of the following polynomial to variable $x$.

$$216\,x^{15} + 4536\,x^{14} + 31752\,x^{13} - 520884\,x^{12} - 42336\,x^{11} - 259308\,x^{10} + 3046158\,x^{9} + 140742\,x^{8} + 756756\,x^{7} - 5792221\,x^{6} - 193914\,x^{5} - 931392\,x^{4} + 3266731\,x^{3} + 90972\,x^{2} + 402192\,x + 592704$$

Then, it replaces $x$ with the assigned value in the polynomial $y^3 + x^3 + 1$, and timeouts trying to isolate the roots of the result. In our package presented in this paper, these two operations are performed in 0.05 secs (nlsat.py).

*Tower of extensions.* In this example, we create a tower of extensions containing two transcendental ($\pi$ and $e$), one infinitesimal ($\epsilon$), and 5 algebraic extensions. The algebraic extensions are the first roots of the following 5 polynomials:

$$\begin{aligned}
r_0 &:= x^4 + -2\,\epsilon\,x^3 + (\epsilon^2 - 4)\,x^2 + 4\epsilon\,x + 4 - 2\epsilon^2 + 4\\
r_1 &:= (-\epsilon^6 + 8\epsilon^4 - 20\epsilon^2 + 16)r_0 - 8\epsilon^5 + 32\epsilon^3 - 32\epsilon + (2\epsilon^4 - 8\epsilon^2 + 8)x^2\\
r_2 &:= x^5 + 3\,x^3 + r_1\,x^2 - 1\\
r_3 &:= x^5 + r_1\,x^3 + \pi\,r_2\,x^2 - 3\\
r_4 &:= 8\,x^5 + r_3\,x^4 + e\,x^2 + x - 7
\end{aligned}$$

All roots are isolated in 0.28 secs (tower8.py). However, if we do not use clean representations (cf. Sec. 3), it takes 31 secs for the 4th polynomial, and times out after 30 min in the last one (tower8_eager_norm.py).

*Rioboo examples [21].* All solved in a negligible amount of time (rioboo*.py).

*Strzebonski examples [22].* All solved in a negligible amount of time (strz.py).

## 5 Conclusion

We have presented a library for computing in real closed transcendental and infinitesimal extensions of the rationals. This provides a computational substructure sufficient for implementing many advanced (and hitherto practically unexplored) decision methods for nonlinear real arithmetic. We hope that both the library and the ideas underlying it will prove useful to the community, and that it may become the foundation of new practically useful decision methods.

## References

1. B. Akbarpour and L. C. Paulson. Applications of MetiTarski in the Verification of Control and Hybrid Systems. In *HSCC*, volume 5469 of *LNCS*. Springer, 2009.
2. Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical Algebraic Decomposition. I. The Basic Algorithm. *SIAM J. Comp.*, 13(4):865–877, 1984.
3. S. Basu, R. Pollack, and M. Roy. *Algorithms in Real Algebraic Geometry.* Springer, Secaucus, NJ, USA, 2006.
4. M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. In *STOC*. ACM, 1984.
5. J. Canny. Some algebraic and geometric computations in PSPACE. In *Twentieth ACM Symposium on Theory of Computing*, STOC. ACM, 1988.
6. M. Coste and M. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *JSC*, 5(1-2), 1988.
7. L. de Moura and G. O. Passmore. Exact nonlinear optimization on demand. *In preparation*, 2013.
8. W. Denman, B. Akbarpour, S. Tahar, M.H. Zaki, and L.C. Paulson. Formal verification of analog designs using MetiTarski. In *FMCAD*, pages 93 –100, 2009.
9. J. D. Fleuriot and L. C. Paulson. A Combination of Nonstandard Analysis and Geometry Theorem Proving, with Application to Newton's Principia. In *CADE-15*, LNCS, pages 3–16. Springer, 1998.
10. M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure. *JSAT*, 1:209–236, 2007.
11. R. Gamboa and M. Kaufmann. Nonstandard Analysis in ACL2. *JAR*, 27(4), 2001.
12. S. Gao, J. Avigad, and E. M. Clarke. $\delta$-complete decision procedures for satisfiability over the reals. In *IJCAR*, LNCS, pages 286–300. Springer, 2012.
13. L. Granvilliers and F. Benhamou. RealPaver: An Interval Solver using Constraint Satisfaction Techniques. *ACM Trans. on Maths. Software*, 32:138–156, 2006.
14. D. Y. Grigor'ev and N.N. Vorobjov Jr. Solving systems of polynomial inequalities in subexponential time. *JSC*, 5(1-2):37–64, 1988.
15. T. C. Hales, J. Harrison, S. McLaughlin, T. Nipkow, S. Obua, and R. Zumkeller. A revision of the proof of the Kepler conjecture. *Discrete & Computational Geometry*, 44(1):1–34, 2010.
16. D. Jovanović and L. de Moura. Solving non-linear arithmetic. In *IJCAR*, IJCAR, Berlin, Heidelberg, 2012. Springer.
17. Z. Ligatsikas, R. Rioboo, and M. Roy. Generic computation of the real closure of an ordered field. *Maths. and Comp. in Sim.*, 42(4-6):541 – 549, 1996.
18. B. Mishra and P. Pedersen. Arithmetic with real algebraic numbers is in NC. In *ISSAC '90*, ISSAC '90, pages 120–126, New York, NY, USA, 1990. ACM.
19. A. Platzer and E. M. Clarke. Formal verification of curved flight collision avoidance maneuvers: A case study. In *Formal Methods*. Springer, 2009.
20. S. Ratschan. Efficient Solving of Quantified Inequality Constraints over the Real Numbers. *ACM Trans. on Comp. Logic*, 7(4):723–748, 2006.
21. R. Rioboo. Infinitesimals and real closure. Technical report, Laboratoire D'Informatique de Paris 6, 2001.
22. A. Strzebonski. Computing in the field of complex algebraic numbers. *JSC*, 24(6), 1997.
23. Adam Strzeboński and Elias P. Tsigaridas. Univariate real root isolation in multiple extension fields. In *ISSAC'12*, pages 343–350, New York, NY, USA, 2012. ACM.