Real Algebraic Closure of an Ordered Field, Implementation in *Axiom* *

Renaud Rioboo
LITP - Boite 168
Institut Blaise Pascal
Universtité Pierre et Marie Curie
4 Place Jussieu
F-75252 PARIS CEDEX 05
e-mail: rr@posso.ibp.fr

Abstract

Real algebraic numbers appear in many Computer Algebra problems. For instance the determination of a cylindrical algebraic decomposition for an euclidian space requires computing with real algebraic numbers. This paper describes an implementation for computations with the real roots of a polynomial. This process is designed to be recursively used, so the resulting domain of computation is the set of all *real algebraic numbers. An implementation for the real algebraic closure has been done in Axiom (previously called Scratchpad).*

Introduction

Real algebraic computations are somewhat different than general algebraic computations, since real numbers have a natural ordering that general algebraic numbers don't have. This induces that real number comparisons do not follow the general "equal" or "different" scheme, but a more complicated "smaller", "equal" or "greater" one. Moreover real algebraic computations introduce a distinction between the different roots of a polynomial (even of an irreducible one). For instance comparing $\sqrt{2}$ and $-\sqrt{2}$ yields to the result "greater" only because the symbol $\sqrt{2}$ is implicitly attached to the particular root of the polynomial $X^2 - 2$ that is positive.

In many applications the effective ordering should be computed. For instance in real geometry the description or the solution of a problem very often involves polynomial inequalities as well as equalities. As an example one can consider in the euclidian plane the intersection of a line $x = c$ with the cubic $y^2 = x^3 - x$, we clearly see that if $c^3 - c < 0$ the intersection is empty, and that in this answer we have introduced an inequality.

After recalling some common properties about fields, we carefully examine simple extensions. Though most algorithms described in this part are well known, we slightly change some of their requirements in order both to optimize the process, and generalize it to other fields than usual rational numbers. This is done for a better understanding of the main goal of this paper that is the introduction of *algebraic towers*, and *real closure*. A full implementation has been done in the *Axiom* computer algebra system that is up to our knowledge the only complete implementation of the *real closure*

Basic definitions about fields

Definition 1

(i) *An* extension field $\mathbf{K}_1$ *of a field* $\mathbf{K}$ *is any field containing* $\mathbf{K}$.

(ii) *Let* $\mathbf{K}_1$ *be an extension field of* $\mathbf{K}$, *an element* x *of* $\mathbf{K}_1$ *is said to be* algebraic over $\mathbf{K}$ *if there exists a polynomial* P *with coefficients over* $\mathbf{K}$, *such that* $P(x) = 0$.

(iii) *An extension field* $\mathbf{K}_1$ *of* $\mathbf{K}$ *is said to be* algebraic *if all its elements are algebraic over* $\mathbf{K}$.

*the final preparation of this paper was supported by the basic research project No 9846 POSSO.

(iv) *An extension field* $\mathbf{K}_1$ *of* $\mathbf{K}$ *is a* simple extension *of* $\mathbf{K}$ *if it is obtained by adjoining a single element* x_0 *to* $\mathbf{K}$*; this simple extension is denoted by* $\mathbf{K}(x_0)$.

Definition 2 *An simple extension* $\mathbf{K}(x_0)$ *of* $\mathbf{K}$ *is said to be* algebraic *if there exists a polynomial* P *such that* $P(x_0) = 0$. *The polynomial* P *can be chosen to be irreducible and monic; it is then called the* minimal *polynomial of* x_0, *and its degree is the* degree of the algebraic extension *denoted by* $[\mathbf{K}(x_0) : \mathbf{K}]$.

Every element of an algebraic extension is algebraic over $\mathbf{K}$. The field $\mathbf{K}(x_0)$ is then isomorphic to the modulus field $\mathbf{K}[X]/P(X)$ of the polynomials modulo P where P is the minimal polynomial of x_0. The above isomorphism is used in most implementations of algebraic extensions. Since the roots of a polynomial don't change by multipling by an element of $\mathbf{K}$, usual implementations use a monic polynomial as the defining polynomial for an algebraic extension.

When the defining polynomial is monic we can define algebraic extensions of a ring instead of a field since elements can still be reduced modulo the defining polynomial. This is the actual Axiom implementation for algebraic extensions and yields to the notion of *integral rings* that is more satisfactory under a computational point of view. In this paper we however will not take advantage of this fact and only consider field extensions since the process is to our opinion clearer and much easier to explain. Extensions to algebraic ordered rings are however currently being written as a generalization of the algorithms explained here.

Definition 3 *A field* C *is said to be* algebraically closed, *if it has no nontrivial algebraic extension. Let* $\mathbf{K}$ *be a field,* C *is an* algebraic closure *of* $\mathbf{K}$ *if*

(i): C *is algebraically closed, and (ii):* C *is an algebraic extension of* $\mathbf{K}$.

Every field has an algebraic closure; and if C_1 and C_2 are two algebraic closures of the same field they are isomorphic. This enables to speak of the algebraic closure of a field.

Ordered and real fields

Definition 4 *A field* $\mathbf{K}$ *is said to be* ordered *if it has a* total *ordering* $\leq$ *satisfying*

(i): $x \leq y \Longrightarrow x + z \leq y + z$, *and (ii):* $(0 \leq x, 0 \leq y) \Longrightarrow 0 \leq xy$

Proposition 1 *Let* $\mathbf{K}$ *be an arbitrary field, it is equivalent that:*

(i) $\mathbf{K}$ *can be ordered,*

(ii) -1 cannot be expressed as a sum of squares of elements of $\mathbf{K}$,

(iii) for all $x_1, \cdots, x_n$ *of* $\mathbf{K}$ *we have:* $\sum_{i=1}^{n} x_i^2 = 0 \Longrightarrow x_1 = \cdots = x_n = 0$

A field having these properties is called a *real field.* We see that a real field has no defined order relation, it is only possible to find one (and usually more) that will change the real field into an ordered field. We can view a real field as an ordered field where the the order relation is not explicitly known. A real field (and as a particular case an ordered field) is always of characteristic 0.

Definition 5 *A* real closed field *is a real field that has no nontrivial real algebraic extension. Let* $\mathbf{K}$ *be an ordered field,* R *an extension field of* $\mathbf{K}$*;* R *is a* real closure *of* $\mathbf{K}$ *if*

(i): R *is real closed, (ii):* R *is an algebraic extension of* $\mathbf{K}$, *and (iii): the only ordering on* R *extends the ordering on* $\mathbf{K}$.

Every ordered field has a real closure, and if R_1 and R_2 are two real closures of the same field they are isomorphic; this enables us to talk of the real closure of an ordered field. In particular we see that the real closure of an ordered field is itself an ordered field.

Real closed fields have the usual properties of real numbers, in particular if R is a real closed field then the algebraic extension $\mathsf{R}[X]/(X^2 + 1)$ is an algebraically closed field. When considering a topology on real closed fileds or on their algebraic closure, we will always consider them with their euclidian topology (induced by the order on R) for which polynomials are continuous, and where elementary analysis remains true. For more about real and ordered fields see [BCR-88].

Proposition 2 *Let* $\mathbf{K}$ *be an ordered field,* $\mathbf{K}(a)$ *a simple algebraic extension of* $\mathbf{K}$ *generated by* a, P_a *the minimal polynomial of* a *over* $\mathbf{K}$. *The number of orders over* $\mathbf{K}(a)$ *extending the order over* $\mathbf{K}$ *is the number of roots of* P_a *in the real algebraic closure of* $\mathbf{K}$.

In other words, let us suppose that we want to compute with quantities that involve some real root of an irreducible polynomial P_a. The algebraic way to deal with these numbers is to work in the simple algebraic extension $\mathbf{K}(a)$. But we loose the order relation though P_a has real roots and the field is real.

When we consider the simple algebraic extension $\mathbf{K}(a)$, the symbol a implicitly denotes any root of P_a. In real terms the symbol a denotes any real root of P_a. A particular ordering on $\mathbf{K}(a)$ specifies one real root of P_a

among all its possible roots. Each compatible ordering on $\mathbf{K}(a)$ places the symbol a "at its right place" on the real line.

Given an ordered field $\mathbf{K}$, an irreducible polynomial P_a with k (> 0) real roots, we have k orderings of the real field $\mathbf{K}(a)$, inducing k ordered fields $\mathbf{K}_{a_1} \cdots \mathbf{K}_{a_k}$; each a_i is now one of the k real roots of P_a. The symbol a_i in the notation $\mathbf{K}_{a_i}$ is no more an arbitrary root of P_a but a specific real root on the line (i.e. a root in the real algebraic closure). For the purpose of computing with real algebraic numbers, we will consider each of these ordered fields as an algebraic object.

Simple ordered extensions

We are now ready to define *ordered* extensions for an ordered field.

Definition 6 *Let* $\mathbf{K}$ *be an ordered field,* $\mathbf{K}_1$ *an algebraic extension of* $\mathbf{K}$ *that is real,* $\mathbf{K}_1$ *is a* ordered extension *of* $\mathbf{K}$ *if it is ordered with an ordering that is compatible with the ordering on* $\mathbf{K}$. $\mathbf{K}_1$ *is a* simple *ordered extension if it is a simple algebraic extension and an ordered extension.*

As we can see, the difference between simple ordered extensions and algebraic extensions for an ordered field, is that an ordered extension has an ordering that extends the basic ordering. This property will be useful when trying to construct higher levels of extensions.

Basic specifications

Our purpose is to make an *Axiom* constructor that behaves like the standard *SimpleAlgebraicExtension* that enables a user to work with algebraic numbers defined by a polynomial. To construct a simple algebraic extension we need to give a base field $\mathbf{K}$ to extend, and a univariate defining polynomial for the generator α of the extension.

However for the case of ordered fields these parameters aren't sufficient, since we need more parameters to define a real algebraic number. Let us consider the simple case of extending the field $\mathbf{Q}$ of the rational numbers by the polynomial $X^2 - 2$, and call α a root of $X^2 - 2$; we can't answer the simple question $\alpha > 0$ since we don't know whether α is $\sqrt{2}$ or $-\sqrt{2}$.

To distinguish among the different real roots of a polynomial P_α we will introduce one parameter more that is an isolation interval for the real root. To construct an ordered extension $\mathbf{K}_\alpha$ (α is the generator of the extension), we will need :

- A base ordered field $\mathbf{K}$ to extend,
- A univariate polynomial P_α with coefficients in $\mathbf{K}$ such that $P_\alpha(\alpha) = 0$
- An interval (a, b) of $\mathbf{K}$ that isolates one real root of P_α.

Just like in the standard *Axiom* constructor *SimpleAlgebraicExtension* the internal representation of the elements of a ordered extension will be polynomials in one variable over the base ordered field $\mathbf{K}$. These polynomials are reduced modulo the defining polynomial P_α in a standard way. We will name our constructor *SimpleOrderedAlgebraicExtension*.

Let us look closer to the requirements for these parameters.

> The base field can be any ordered field where the isolating interval can be refined to any precision. This is always the case for archimedian fields such as $\mathbf{Q}$ or any of its ordered extensions.
>
> The defining polynomial P_α verified by the generator α needs not to be irreducible over the base field $\mathbf{K}$ since a real algebraic number can be defined with any multiple of its minimal polynomial. Like in the *D5* ([Duv-87]) system we only need a square free polynomial. This results in considerable time saving since factoring of polynomials can be very long, particularly when the base field is not the usual rational numbers field, but for instance one of its algebraic extensions. In order to save time during computations this polynomial is regularly updated.
>
> In a similar way, the isolating interval (a, b) for a root of P_α is refined during computations. Just like for the polynomial, any application asking for the interval will see the updated value. Older values should be saved with explicit copies by the programmer if needed. For technical reasons that we will later see, we will also need b to be such that $P_\alpha(b) \neq 0$.

Throughout this section we will consider $\mathbf{K}_\alpha$ the ordered extension of an ordered field $\mathbf{K}$ with generator α defined by a square free polynomial P_α, and interval (a, b).

In *Axiom*, when defining a domain, there is always two ways to view an object of the domain. We can view it as an element of the domain we are defining, or as an element of the domain we use to represent the objects of the new domain. Here when viewing an object as an algebraic number we will denote it with a greek letter (i.e. ω). We will use the notation Rep_ω (or $\mathrm{Rep}_\omega(X)$ if necessary) for the algebraic number ω viewed as its representation, namely a polynomial in one variable over the base field.

Arithmetic operators

We can now examine the implementation for the arithmetic operations in our field. Most algorithms here are derived from [Loo-82]. In *Axiom* many functions are inherited by the category of the object and default implementations are supplied for them. In an actual code very few functions need to be explicitly written by the programmer. Our basic field operations will be: addition, subtraction, multiplication, division, and the zero test that checks if an element ω of $\mathbf{K}_\alpha$ is zero.

Addition, subtraction, and multiplication

There is no real problem to add or subtract algebraic elements since $\mathbf{K}_\alpha$ is isomorphic to the modulus field $\mathbf{K}[X]/P(X)$, and since addition and subtraction don't increase the degree of the polynomials involved. To add or substract two elements of $\mathbf{K}_\alpha$ we simply add or substract their representations.

Multiplication increases the degree, so simple operations on polynomials don't suit our needs. We need to multiply the polynomials and reduce the result modulo the defining polynomial P_α. Note however that this may destroy integral nature of terms: let us suppose that P_α has integer coefficients but is not monic; even if Rep_{ω_1} and Rep_{ω_2} have integer coefficients, $\mathrm{Rep}_{\omega_1\omega_2}$ has rational coefficients.

Zero check and division

Before implementing division we have to see how to check equality to zero in order to avoid zero division. Let us recall here that the defining polynomial P_α is not supposed to be irreducible, the algebraic number α can be a root of a divisor of P_α.

The non-real solution given by the D5 system is that whenever we encounter a polynomial P that has a non trivial gcd with P_α, we have to introduce a *case distinction*. Computations are separated in two parts:

- one computation corresponding to the case where the generator α is a root of $\gcd(P, P_\alpha)$, and
- another corresponding to the case where α is a root of $P/\gcd(P, P_\alpha)$.

This technique enables to work with all the roots of P_α, but strong typing of *Axiom* objects is lost since the result of a zero check is no more a boolean value but a list of cases induced by subcomputations.

In fact, in the real case our generator α is a specific real number since it is characterized by the interval (a, b). We can answer to the zero check in a boolean manner.

Let us consider an element ω of $\mathbf{K}_\alpha$, checking ω to zero is in fact checking $\mathrm{Rep}_\omega(\alpha)$ in the real algebraic closure of $\mathbf{K}$ (recall that Rep_ω is a polynomial). Let us consider the polynomial D that is the gcd of Rep_ω and P_α, we only have to know if $D(\alpha)$ is equal to zero in the real closure of $\mathbf{K}$. Since P_α is square free, D is also square free and can only have one root in the interval (a, b). Since our isolating interval (a, b) was chosen such that $P_\alpha(b) \neq 0$ and that P_α has only one root in the interval (a, b) we have $P_\alpha(a)P_\alpha(b) \leq 0$ and the root is in fact in the interval $[a, b[$.

Moreover, whenever we find $D(\alpha) = 0$, the algebraic number α can be defined by either P_α or D, and we can use D instead of P_α in further computations. We can replace P_α by D without altering any property of the field we are defining. $\mathbf{K}_\alpha$ is characterized by the real algebraic number α, not by the parameters P_α and (a, b) used to define it; any set of parameters defining the same algebraic number α defines the same field $\mathbf{K}_\alpha$. Since we are defining the field, this modification is safe as long as the parameters are not used for other purposes.

In a similar way, when we find $D(\alpha) \neq 0$, we have $\frac{P_\alpha}{D}(\alpha) = 0$ and P_α can be replaced by P_α/D.

Division is classically implemented using the Bezout relation for univariate polynomials. In the actual code the zero test is merged inside inversion, so that gcd computation is only done once.

The order relation

We have now completed the field operations for $\mathbf{K}_\alpha$, and want to implement the ordering on $\mathbf{K}_\alpha$; up to a subtraction it is equivalent to compute the sign of any element ω of $\mathbf{K}_\alpha$. So our problem reduces to determining the sign of $\mathrm{Rep}_\omega(\alpha)$ in the real closure of $\mathbf{K}$. Since we already know how to check if $\mathrm{Rep}_\omega(\alpha) = 0$ we can assume that $\omega \neq 0$.

Restating our problem in the real closure R of $\mathbf{K}$ we want to know the sign of $\mathrm{Rep}_\omega(\alpha)$ given a, b, α, P_α such that: $a \leq \alpha < b$, $P_\alpha(\alpha) = 0$, $\mathrm{Rep}_\omega(\alpha) \neq 0$, and $P_\alpha(b) \neq 0$. If we can find a refinement (a_1, b_1) of (a, b) on which Rep_ω remains of constant sign, the sign of $\mathrm{Rep}_\omega(\alpha)$ will be that of $\mathrm{Rep}_\omega(a_1)$ or $\mathrm{Rep}_\omega(b_1)$. Our problem is now to find a_1 and b_1 in $\mathbf{K}$ such that $a \leq a_1 \leq \alpha < b_1 \leq b$, and that Rep_ω has no roots on $[a_1, b_1[$.

Changing variables

We will use Descartes' rule of sign, and a change of variable in order to give a sufficient condition for a polynomial P to have no roots over an interval $[c, d[$. Let us recall that by Descartes' rule of sign we know that if a polynomial has no sign variation in the list of it's non-null coefficients then it has no roots over $]0, +\infty[$.

We will change the definition of sign variations in order to fit our needs. Let P be any polynomial, instead of taking the non-null coefficients of P we will consider all its coefficients between 0 and degree(P), that is including the eventual zeros. We will define the sign variation of a list $n_0, n_1, \cdots, n_k$ so that it counts the sign variations of $n_0, n_{i_1}, \cdots, n_{i_l}$ where the n_{i_j} are the non-null elements of $n_1, \cdots, n_k$. In this definition zero counts as a sign, so the only difference between our sign variation and the usual one is that the constant coefficient of P is counted as a sign whether it is null or not.

With this new definition we can see ([Rio-91]) that if a polynomial has no sign variation then it has no roots in $[0, +\infty[$. But our problem was stated for an arbitrary interval $[c, d[$. To make a mapping between $[c, d[$ and $[0, +\infty[$ we will use the change of variables:

$$F_{c,d} \quad : \quad x \mapsto \frac{dx + c}{x + 1}$$

that maps $[0, +\infty[$ on $[c, d[$ and the complex right half plane $\mathrm{Re}(z) \geq 0$ on the disc of center $\frac{c+d}{2}$ and diameter $\frac{d-c}{2}$. For a polynomial $P(X)$ we consider the polynomial:

$$P^*_{c,d}(X) = (X + 1)^n P(\frac{dX + c}{X + 1})$$

where n = degree(P), and whose roots on $[0, +\infty[$ are transformed into the roots of P on $[c, d[$ by $F_{c,d}$. Now checking the sign variations of $P^*_{c,d}$ gives us a sufficient condition to check for the absence of roots on $[c, d[$.

Sign determination

Since polynomials are continuous functions, and since $\mathrm{Rep}_\omega(\alpha) \neq 0$, we know that there exists a disc in the algebraic closure of $\mathbf{K}$ on which $\mathrm{Rep}_\omega(\alpha)$ is never null.

To complete sign determination we only have to find such an interval. Since α is in $[a, b[$, we can refine it splitting it in $[a, \frac{a+b}{2}[$ and $[\frac{a+b}{2}, b[$[1] and selecting the one that contains α until $\mathrm{Rep}^*_{\omega a,b}$ has no sign variation. This is always possible in an archimedian field such as $\mathbf{Q}$ or any of it's ordered extensions.

[1]This is to avoid a 3 cases test that we introduced semi open intervals and asked that $P_\alpha(b) \neq 0$.

Real algebraic closure

We have completed our ordered field operations for $\mathbf{K}_\alpha$, and we can compute with one specific root of a square free polynomial. We can compute for instance in $\mathbf{Q}_{\sqrt{2}}$ or in $\mathbf{Q}_{-\sqrt{2}}$ but we cannot yet answer to more complicated questions such as: $(\sqrt{2} + (-\sqrt{2}) = 0)$ since they involve different extensions (here $\mathbf{Q}_{\sqrt{2}}$ and $\mathbf{Q}_{-\sqrt{2}}$).

Towers of extensions

Using primitive elements we could always compute with quantities that involve only one real algebraic number, but this method is very slow since the degrees of the defining polynomials are prohibitive.

Since the field $\mathbf{K}_\alpha$ we have constructed has the same properties than the base field $\mathbf{K}$, we can construct ordered extensions of $\mathbf{K}_\alpha$ with the same method. We can give us a square free polynomial P_β with coefficients on $\mathbf{K}_\alpha$, and an isolating interval $[a_\beta, b_\beta[$ for a real root β of P_β to construct an ordered field $\mathbf{K}_{\alpha,\beta}$. For instance we can construct $\mathbf{Q}_{\sqrt{2}}$ using $X^2 - 2$ and $[0, 2[$ and then construct $\mathbf{Q}_{\sqrt{2},-\sqrt{2}}$ using $Y^2 - 2$ and $[-2, 0[$, the question $\sqrt{2} + (-\sqrt{2}) = 0$ can be answered in $\mathbf{Q}_{\sqrt{2},-\sqrt{2}}$. That is to say in axiom-like syntax:

```
)ab domain SOAE SimpleOrderedAlgebraicExtension
RN := Fraction(Integer)
Ext1 := SOAE(RN,0..2,monomial(1,2)-2,'s2)
Ext2 := SOAE(Ext1,(-2)..0,monomial(1,2)-2,'_-s2 )
```

In a more general way we can iterate the previous process to construct *towers of extensions* where every new algebraic number is defined in terms of previously defined ones. Starting with an ordered field $\mathbf{K}$ we can construct $\mathbf{K}_{\alpha_1}$ using P_{α_1} and $[a_1, b_1[$ above $\mathbf{K}$, and $\mathbf{K}_{\alpha_1 \cdots \alpha_{n+1}}$ using $P_{\alpha_{n+1}}$ and $[a_{n+1}, b_{n+1}[$ above $\mathbf{K}_{\alpha_1 \cdots \alpha_n}$. This induces an efficient method to compute with a sequence of real algebraic numbers $\alpha_1 \cdots \alpha_n$.

If we look at the polynomial P_{α_i} defining $\mathbf{K}_{\alpha_1 \cdots \alpha_i}$, we see that it has coefficients on $\mathbf{K}_{\alpha_1 \cdots \alpha_{i-1}}$, and that a number in $\mathbf{K}_{\alpha_1 \cdots \alpha_i}$ is represented by a univariate polynomial with coefficients on $\mathbf{K}_{\alpha_1 \cdots \alpha_{i-1}}$. A polynomial with coefficients on $\mathbf{K}_{\alpha_1 \cdots \alpha_i}$ can be viewed as a two variables polynomial with coefficients on $\mathbf{K}_{\alpha_1 \cdots \alpha_{i-1}}$, that is to say $\mathbf{K}_{\alpha_1 \cdots \alpha_i}[X]$ can be viewed as $\mathbf{K}_{\alpha_1 \cdots \alpha_{i-1}}[X, \alpha_i]$. Iterating this method we see that an element of $\mathbf{K}_{\alpha_1 \cdots \alpha_i}$ can be viewed as a multivariate polynomial in $\alpha_1, \cdots, \alpha_i$ with coefficients on $\mathbf{K}$, and that a polynomial with coefficients on $\mathbf{K}_{\alpha_1 \cdots \alpha_i}$ is a multivariate polynomial with coefficients on $\mathbf{K}$ (i.e.$\mathbf{K}_{\alpha_1 \cdots \alpha_i}[X] = \mathbf{K}[X, \alpha_i, \cdots \alpha_1]$. We can look to the defining polynomials for an algebraic tower of extensions as a triangular system of multivariate polynomials with coefficients on $\mathbf{K}$. The system is triangular because each element is reduced modulo the defining polynomial of the extension.

Given a triangular set of multivariate polynomials and a corresponding list of intervals (that we can always take on the base field) we can construct the corresponding ordered tower. There is no real code in these towers of extensions since they only avoid the user to "handly" construct each level of extension and to change the types of the different objects involved

Implicit towers

Fixed length towers enable a user to work with a given set of real algebraic numbers; though this fits many applications sometimes new algebraic numbers appear while computations are done. The introduction of a new algebraic number requires a tedious rebuilding a new tower for further computations. Moreover, in a fixed length tower $\mathbf{K}_{\alpha_1 \cdots \alpha_n}$, every element is in fact a polynomial in *all* the variables $\alpha_1 \cdots \alpha_n$, even if it doesn't depend on all of the algebraic quantities $\alpha_1 \cdots \alpha_n$. This introduces more algebraic levels than necessary for a given element.

Representation

We will use a different coding in order to minimize the height of towers used in the process. We will include the algebraic tower used to construct an object *inside* its representation. To define a real algebraic quantity ω we will give:

- a polynomial P_α describing a real algebraic number α,
- an isolating interval $[a_\alpha, b_\alpha[$ for α,
- a value giving ω in function of α.

Let us look closer at these parameters.

> The polynomial P_α is such that $P_\alpha(\alpha) = 0$; its coefficients need not necessary be in the base field but are real algebraic numbers in the sense of the present representation.
>
> The isolating interval should be such that $a_\alpha \leq \alpha < b_\alpha$ and $P_\alpha(b_\alpha) \neq 0$. The algebraic numbers a_α and b_α isolate one root of P_α. In practice however they remain on the base field.
>
> The value giving ω in function of α is a univariate polynomial as it would be in a simple ordered extension. The only difference is that we don't need it to have coefficients on the base field, but only real algebraic coefficients.

We see that the only requirements we have for the parameters is that they should be real algebraic. Since we are defining real algebraic numbers, we only want them to be expressed in terms of previously defined real algebraic numbers.

Now the recursive nature of real algebraic numbers can appear in a simple recursive definition of an *Axiom* constructor : *RealClosure*. The argument of that constructor is the field of which we are computing the closure. The representation for the elements in *RealClosure* is recursive as in the standard *Axiom* constructor*MultivariatePolynomial*. Here is the representation part of the objects :

```
Structure := Record(name :   Symbol,
                    nullpol : SparseUnivariatePolynomial($),
                    approx : Segment($),
                    value :   SparseUnivariatePolynomial($))
Rep := Union(TheField , Structure)
```

Here *TheField* is the ordered field we are closing, thus basic numbers are considered algebraic. The field "name" in the record is used to print the objects and to give an ordering on the different algebraic quantities involved.

Let us look how would be represented the number $\sqrt{2} + 2$ when computing the real closure of the rational numbers:

- the "name" field contains a name to print the algebraic number $\sqrt{2}$, let's call it α.
- The "nullpol" field contains a polynomial verified by the algebraic number $\sqrt{2}$, for instance $X^2 - 2$, here its coefficients are rational numbers.
- The "approx" field contains an isolating interval for $\sqrt{2}$, let's say $[0, 2[$.
- the "value" field contains the value of $\sqrt{2} + 2$, in function of $\sqrt{2}$, that is to say $X + 2$, this polynomial has rational coefficients.

This number or any other defined with $\sqrt{2}$ can now appear in definitions for further real algebraic objects.

Operations

In order to implement the operations on our domain, let us recall that an element of *RealClosure* only has in its representation the necessary algebraic quantities used to define it. So the algebraic towers corresponding to different elements are different, and we must update their internal towers in order to make an operation in a proper simple ordered extension.

Our first utility operations will be the ones needed to manipulate the tower of an element. We will have three operations: *isAlgebraic?*, *moreAlgebraic*, and *lessAlgebraic*. The operation *isAlgebraic?* takes an element ω of the closure and an expression α; it checks if α appears in the defining tower of ω. In the same spirit *moreAlgebraic* forces ω to depend on α, and *lessAlgebraic* removes if it can the dependency of ω in α (i.e. if ω doesn't depend on the algebraic quantity α).

Now, as we can update the towers of different elements, we can implement the ordered field operations themselves. Let us suppose that we want to implement an operation *op* of two arguments of the real closure that returns an element of the real closure. That is to say in axiom-like syntax: *op*: (\$,\$) $\rightarrow$ \$, meaning that *op* is a two arguments operation of the real closure.

We are defining *RealClosure(TheField)* with representations as above, *op(number1,number2)* is given by:

- if *number1* and *number2* are both in *TheField* return *op(number1,number2)* (since *op* exists in *TheField*).
- Up to a reorganization of terms, we can assume that the greatest algebraic quantity appearing in *number1* and *number2* is *number1.name*.
- if *number2* is in *TheField* or if *number2.name* < *number1.name* then:
  - call *op* recursively with arguments *number1* and *number2* forced to be algebraic in *number1.name*.
  - eventualy remove the algebraic dependency of the result in *number1.name*, and return it.
- Now *number1* and *number2* both depend on the same algebraic quantity *number1.name*. Let *EffectiveField* be the simple extension of *RealClosure(TheField)* by the algebraic quantity defining both *number1* and *number2*. This field is licit because *RealClosure(TheField)* has the correct properties to be an argument of *SimpleOrderedAlgebraicExtension*.
- Transform the univariate polynomials *number1.value* and *number2.value* to elements of *EffectiveField*, and call *alg1* and *alg2* the results.
- Now *op* exists in *EffectiveField*. Call *op* with arguments *alg1* and *alg2* (in *EffectiveField*), transform the result back to univariate polynomials and call it *aValue*.
- Let *res* be the algebraic number defined by *number1.name*, *number1.approx* *number1.nullpol* and *aValue*. Eventualy remove the dependecy of *res* in *number1.name*, and return it.

As we can see there are three operations called *op* involved:

- the base field *op* used in the trivial case,
- the effective field *op* used when algebraic towers match at their first level,
- the real closure *op* recursively called when algebraic towers don't match at theit first level.

As we saw in the previous section on simple extensions, implementation of an operation *op* relies on operations on the base field. Here the operation *op* in *EffectiveField* will itself call other operations in *RealClosure(TheField)* (since it is the base field of the simple extension). The latter code terminates since when called from *EffectiveField*, an operation of *RealClosure* has arguments with one less level of algebraicity.

For clarity let us see what happens when we add $\sqrt{2}+1$ and $\sqrt{3}$ in the real closure of $\mathbf{Q}$:

- $\sqrt{2}+1$ and $\sqrt{3}$ are both represented as towers of height 1.
- $+$ is called with arguments $\sqrt{2}+1$ and $\sqrt{3}$ in the real closure, let us suppose that the main algebraic quantity is $\sqrt{2}$. Since algebraic quantities differ, $\sqrt{3}$ is transformed in a tower of height 2 with $\sqrt{2}$ as its first level algebraic quantity. The operation $+$ is then recursively called with those two arguments.
- Now the first levels of both argument match, and the effective field is the simple extension of the real closure defined by the polynomial X^2-2, and a suitable isolating interval.
- The values expressing both arguments as polynomials in one variable over the real closure are:
  - $X+1$ for $\sqrt{2}+1$
  - the constant polynomial $\sqrt{3}$ (since it doesn't depend on $\sqrt{2}$).
- Both values are then transformed into elements of the effective field, since they are reduced modulo X^2-2, they remain the same.
- $+$ is called in the effective field with those two arguments. Addition in a simple extension is just polynomial addition, thus $+$ of univariate polynomials over the real closure is called.
- Since polynomial addition relies on coefficient addition, to complete the polynomial addition of $X+1$ and the constant polynomial $\sqrt{3}$ there is a call to the real closure addition $1+\sqrt{3}$. Let us examine this call:

  - Arguments of this call are 1 and the tower of height 1 $\sqrt{3}$.
  - The number 1 is transformed in a tower of level 1 with main algebraic quantity $\sqrt{3}$. The real closure addition is then recursively called with these arguments.
  - As above the effective operation is called and univariate polynomial addition is called with arguments X and the constant polynomial 1.
  - This addition returns $X+1$ since there is no more call to real closure addition.
  - since $X+1$ is simplified modulo the defining polynomial of $\sqrt{3}$, the effective field addition is terminated with result $\sqrt{3}+1$.
  - This result is transformed back to univariate polynomial over the real closure becoming $X+1$, the algebraic quantities are here X^2-3 that defines $\sqrt{3}$ and a suitable isolating interval.
  - Since $X+1$ is not constant, and since X^2-3 is not linear, algebraicity cannot be removed, and the call terminates with $\sqrt{3}+1$ as result.

- This completes polynomial addition with result $X+(\sqrt{3}+1)$, and the effective field operation terminates with result $\sqrt{2}+(\sqrt{3}+1)$.
- As in the previous call, algebraicity cannot be removed so the result is finally given. It is a tower of height 2 with first level defining polynomial: X^2-2 and first level value $X+a$, where a is a tower of height 1 whose defining polynomial is X^2-3 and value is $X+1$.

Operations with only one argument are simply done in the effective field, removing eventual superfluous algebraic levels as above. As we saw the key to all the operations is the correct construction of an effective field and a call to its operations. In fact this process is a recursive version of manual tower manipulations.

Other functions used to produce new real algebraic numbers, and some access functions to the different defining parameters of a real algebraic number were also implemented in order to provide the user with a reasonable interface.

Examples

We will give here some example of use in axiom of our constructor. Computations were done using *axiom* version 1.0 on an IBM RS6000 model 320H[2].

Basic use :

[2]Computing times are indicative, many parts of the code still need to be optimized

```
axiom(1)->Alg := RealClosure(Fraction(Integer))

   (1)  RealClosure Fraction Integer
                                                 Type: Domain
                 Time: 0.04 (IN) + 0.02 (OT) = 0.06 sec
axiom(2)->n1:=algebraicOf(0..2,monomial(1,2)-2,'s2)$Alg
-- creates an algebraic number

   (2)  s2
                               Type: RealClosure Fraction Integer
      Time: 3.90 (IN) + 0.12 (EV) + 1.14 (OT) = 5.16 sec
axiom(3)->(n1+2)**2

   (3)  4s2 + 6
                               Type: RealClosure Fraction Integer
      Time: 0.39 (IN) + 0.11 (EV) + 0.24 (OT) = 0.74 sec
axiom(4)->n2:=algebraicOf(0..2,monomial(1,2)-3,'s3)$Alg

   (4)  s3
                               Type: RealClosure Fraction Integer
      Time: 0.11 (IN) + 0.03 (EV) + 0.06 (OT) = 0.20 sec
axiom(5)->1/(n1-n2)

   (5)  - s3 - s2
                               Type: RealClosure Fraction Integer
      Time: 0.05 (IN) + 1.56 (EV) + 0.10 (OT) = 1.71 sec
axiom(6)->% < 0

   (6)  true
                                                Type: Boolean
      Time: 0.24 (IN) + 0.13 (EV) + 0.08 (OT) = 0.45 sec
axiom(7)->mainSegOf(n1)    -- gives updated interval


   (7)  [1,2[
    Type: RightOpenInterval RealClosure Fraction Integer
                  Time: 0.02 (IN) + 0.10 (OT) = 0.12 sec
```

Computations in the real closure are done the same way than in the field, let us see some harder examples. The roots of the polynomial X^4-10X^2+1 are $-\sqrt{2}-\sqrt{3}$, $\sqrt{2}-\sqrt{3}$, $-\sqrt{2}+\sqrt{3}$ and $\sqrt{2}+\sqrt{3}$. We will introduce two algebraic quantities *alg1* and *alg2* to be $-\sqrt{2}+\sqrt{3}$ and $\sqrt{2}+\sqrt{3}$. We will then check $(\frac{alg1+alg2}{2})^2$ to 3.

```
axiom(1)->Alg := RealClosure(Fraction(Integer))

   (1)  RealClosure Fraction Integer
                                                  Type: Domain
                  Time: 0.01 (IN) + 0.01 (OT) = 0.02 sec
axiom(2)->alg1 := algebraicOf(0..2,_
                    monomial(1,4)-monomial(10,2)+1,_
                    '_-s2_+s3 )$Alg

   (2)  -s2+s3
                                Type: RealClosure Fraction Integer
       Time: 2.17 (IN) + 0.01 (EV) + 1.02 (OT) = 3.20 sec
axiom(3)->alg2 := algebraicOf(2..4,_
                    monomial(1,4)-monomial(10,2)+1,_
                    's2_+s3 )$Alg

   (3)  s2+s3
                                Type: RealClosure Fraction Integer
       Time: 0.13 (IN) + 0.02 (EV) + 0.07 (OT) = 0.22 sec
axiom(4)->(((alg1+alg2)/2)**2 = 3)@Boolean

   (4)  true
```

```
                                         Type: Boolean
     Time: 2.14 (IN) + 1.57 (EV) + 0.30 (OT) = 4.01 sec
axiom(5)->mainAlgOf alg1 --gives the defining polynomial

           4      2
   (5)  ?  - 10?  + 1
       Type: SparseUnivariatePolynomial RealClosure ...
                   Time: 0.01 (IN) + 0.13 (OT) = 0.14 sec
axiom(6)->mainAlgOf alg2

                    3
   (6)  ? + -s2+s3  - 10-s2+s3
       Type: SparseUnivariatePolynomial RealClosure ...
                   Time: 0.01 (IN) + 0.08 (OT) = 0.09 sec
axiom(7)->alg2

                  3
   (7)  - -s2+s3  + 10-s2+s3
                            Type: RealClosure Fraction Integer
                                        Time: 0.06 (OT) = 0.06 sec

axiom(8)->mainVarOf alg2

   (8)  s2+s3
                                    Type: Union(Symbol,...)
                   Time: 0.02 (IN) + 0.14 (OT) = 0.16 sec
```

As we can see here the defining polynomial for *alg2* has been updated since it was a mutiple of $X - alg1$. Simplifications occured during various zero checks during computation. However since *alg2* was constructed using a polynomial, algebraic dependency is not removed on *alg2* but it will be in any computation with it:

```
axiom(9)->alg2**2

                 2
   (9)  - -s2+s3  + 10
                            Type: RealClosure Fraction Integer
     Time: 0.02 (IN) + 0.10 (EV) + 0.02 (OT) = 0.14 sec
axiom(10)->mainVarOf %

   (10)  -s2+s3
                                    Type: Union(Symbol,...)
                   Time: 0.01 (IN) + 0.05 (OT) = 0.06 sec
```

Let us see a non obvious square roots comparison.

```
axiom(1)->Alg := RealClosure(Fraction(Integer))

   (1)  RealClosure Fraction Integer
                                              Type: Domain
                   Time: 0.01 (IN) + 0.01 (OT) = 0.02 sec
axiom(2)->mysqrt(n:PositiveInteger):Alg ==_
          algebraicOf(0..n,_
          monomial(1,2)-n,_
          concat("s",n::String)::Symbol)$Alg
-- this avoids typing
   Function declaration mysqrt : PositiveInteger ->
   RealClosure Fraction Integer has been added to
   workspace.
                                                Type: Void
                                 Time: 0.04 (IN) = 0.04 sec
axiom(3)->s28:=mysqrt(28);s82:=mysqrt(82);_
          s33:=mysqrt(33);s74:=mysqrt(74);
-- we are not interested to see the results
   Compiling function mysqrt with type PositiveInteger
   -> RealClosure Fraction Integer
                            Type: RealClosure Fraction Integer
     Time: 1.36 (IN) + 0.11 (EV) + 0.10 (OT) = 1.57 sec
axiom(4)->s28+s82<s33+s74

   (4)  true
                                             Type: Boolean
   Time: 0.02 (IN) + 64.20 (EV) + 0.02 (OT) = 64.24 sec
```

Computing time seems a little long for this example of a sign computation in a tower of heigth 4, but it cannot be solved using primitive elements techniques. As far as we know, another implementation using tower concepts takes significantly more time. Computation is long because many interval refinements take place; let us look how much an interval has been refined:

```
axiom(5)->(hi(mainSegOf(s28))-lo(mainSegOf(s28)))_
          :: SmallFloat

   (5)  8.149072527885437E-10
                                          Type: SmallFloat
                   Time: 0.95 (IN) + 0.66 (OT) = 1.61 sec
```

Since the process is independent of the base field we can compute in a "floating real closure" saving some time:

```
axiom(1)->Alg := RealClosure(SmallFloat)

   (1)  RealClosure SmallFloat
                                              Type: Domain
                                 Time: 0.01 (OT) = 0.01 sec
axiom(2)->mysqrt(n:PositiveInteger):Alg ==_
          algebraicOf(0..n,_
          monomial(1,2)-n,_
          concat("s",n::String)::Symbol)$Alg
   Function declaration mysqrt : PositiveInteger ->
   RealClosure SmallFloat has been added to workspace.
                                                Type: Void
                                 Time: 0.04 (IN) = 0.04 sec
axiom(3)->s28:=mysqrt(28);s82:=mysqrt(82);_
          s33:=mysqrt(33);s74:=mysqrt(74);
   Compiling function mysqrt with type PositiveInteger
   -> RealClosure SmallFloat

                                Type: RealClosure SmallFloat
     Time: 1.51 (IN) + 0.18 (EV) + 0.05 (OT) = 1.74 sec
axiom(4)->s28+s82<s33+s74

   (4)  true
                                             Type: Boolean
   Time: 0.04 (IN) + 41.57 (EV) + 0.01 (OT) = 41.62 sec
axiom(5)->hi(mainSegOf(s28))-lo(mainSegOf(s28))

   (5)  8.149072527885437E-10
                                Type: RealClosure SmallFloat
                  Time: 0.03 (IN) + 0.18 (OT) = 0.21 sec
axiom(6)->(hi(mainSegOf(s28))+lo(mainSegOf(s28)))/2
-- an approximate value

   (6)  5.2915026221307926
                                Type: RealClosure SmallFloat
                  Time: 0.07 (IN) + 0.24 (OT) = 0.31 sec
```

The advantage over complete floating point computations is that we don't loose precision for algebraic computations:

```
axiom(7)->s28 ** 2
```

```
   (7)  28.0
                               Type: RealClosure SmallFloat
     Time: 0.02 (IN) + 0.01 (EV) + 0.02 (OT) = 0.05 sec
axiom(8)->sqrt(28)$SmallFloat ** 2

   (8)  28.000000000000004
                                         Type: SmallFloat
               Time: 0.03 (IN) + 0.04 (OT) = 0.07 sec
axiom(9)->sqrt(28)$SmallFloat > s28

   (9)  true
                                            Type: Boolean
     Time: 0.06 (IN) + 0.03 (EV) + 0.08 (OT) = 0.17 sec
```

Conclusions

Given a suitable ordered field, we have given a constructor that takes this field as its sole argument and enables to work in the real closure of the field. Manipulation tools to work with real algebraic numbers are provided so that computing in the real closure is as easy as computing in the base field.

In particular, any algorithm working on the base field can work in the real closure without any change. Tools to transparently introduce new algebraic numbers were implemented so that new algebraic quantities can be introduced at any moment during computations.

The main difference between this implementation and [Lan-90] is that:

- defining polynomials are not supposed irreducible, and
- no "tower manipulation" is required by the programmer.

To our knowledge this is the only implementation satisfying these two requirements.

References

[BCR-88] J. Bochnak, M. Coste, M.F. Roy: *Géométrie algébrique réelle* Springer-Verlag, New-York 1988

[C.L-82] G. E. Collins, R. Loos: *Real zeros of polynomials* In *Computer Algebra* Springer-Verlag, New-York 1982 pp. 83-94

[Duv-87] D. Duval: *Diverses questions relatives au calcul formel avec des nombres algébriques* Institut Fourrier, Grenoble 1987

[DST-87] J. Davenport, Y. Siret, E. Tournier: *Calcul Formel : systèmes et algorithmes de manipulations algébriques* Masson, Paris 1988

[Joh-91] J. R. Johnson: *Algorithms for Polynomial Real Root Isolation* Department of Computer and Information Science, The Ohio State University, 1991

[JSW-88] R. D. Jenks, R. S. Sutor, S. M. Watt: *Scratchpad II: An abstract Datatype system for mathematical computation* In *Lecture Notes in Computer Science n. 296* p. 12 Springer-Verlag, New-York 1988

[Lan-90] L. Langmuyr: *Algorithms for a multiple algebraic extension* In *Effective methods in algebraic geometry* Progress in mathematics n. 94 pp. 235-248, Birkhauser Boston 1991

[Loo-82] R. Loos: *Computing in agebraic extensions* In *Computer Algebra* Springer-Verlag, New-York 1982 pp. 173-187

[Rio-91] R. Rioboo: *Quelques aspects du calcul exact avec les nombres réels* Institut Blaise Pascal, Paris 1991

[Rub-73] C. M. Rubald: *Algorithms for Polynomials over a Real Algebraic Number Field* University of Wisconsin ,1973

[Rum-76] S. M. Rump: *On the Sign of a Real Algebraic Number* In *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation* ACM, New York, 1976, pp. 238–241

[Zas-70] H. Zassenhaus, *A Real Root Calculus,* In *Computational Problems in Abstract Algebra* Ed. John Leach, Pergamon Press, Oxford, 1970, pp. 383–392