# Generalizing a Mathematical Analysis library in Isabelle/HOL

Jesús Aransay and Jose Divasón[⋆]

Departamento de Matemáticas y Computación,
Universidad de La Rioja
jesus-maria.aransay@unirioja.es, jose.divasonm@unirioja.es

**Abstract.** The HOL Multivariate Analysis Library (*HMA*) of Isabelle/HOL is focused on concrete types such as $\mathbb{R}$, $\mathbb{C}$ and $\mathbb{R}^n$ and on algebraic structures such as real vector spaces and Euclidean spaces, represented by means of type classes. The generalization of *HMA* to more abstract algebraic structures is something desirable but it has not been tackled yet. Using that library, we were able to prove the Gauss-Jordan algorithm over real matrices, but our interest lied on generating verified code for matrices over arbitrary fields, greatly increasing the range of applications of such an algorithm. This short paper presents the steps that we did and the methodology that we devised to generalize such a library, which were successful to generalize the Gauss-Jordan algorithm to matrices over arbitrary fields.

**Keywords:** Theorem proving, Isabelle/HOL, type classes, Linear Algebra

## 1 Introduction

The importance and use of theorem provers grow day to day, not only involving strictly the formalization of mathematical results but also in the verification of software and hardware. Isabelle is one of the most used and well-known theorem provers, on top of which different logics are implemented; the most explored of these varieties of logics is higher-order logic (or HOL), and it is also the one where the greatest number of tools (code generation, automatic proof procedures) are available. It has been successfully used, for instance, in the Flyspeck project (the largest formal proof completed to date) and in the formal verification of seL4, an operating-system kernel.

The HOL Multivariate Analysis Library (or, *HMA* for short) is a set of Isabelle/HOL theories that has been sucessfully used in concrete developments in Analysis, Topology and Linear Algebra. It contains about 2500 lemmas and 150 definitions and is based on the impressive work of J. Harrison in HOL Light [1]. Formalization of algorithms in Linear Algebra and code generation from datatypes and such algorithms had not been explored in *HMA*. To fulfill this goal, in [5] we presented a formalization of the Gauss-Jordan algorithm

[⋆] This author is sponsored by a research grant FPI-UR-12 of Universidad de La Rioja.

based on *HMA*. In that development, we set up Isabelle to generate code from the matrix representation presented in *HMA*. A refinement to immutable arrays was carried out to improve performance. We also formalized some of its well-known applications: computation of ranks, inverses, determinants, dimensions and bases of the four fundamental subspaces of a matrix and solutions of systems of linear equations. Verified code of these computations is generated to both SML and Haskell.

However, while formalizing the previous results we found a limitation in *HMA*: some important results that we needed were only proven for real matrices or for real vector spaces. Due to this fact, we were only able to generate verified code of the Gauss-Jordan algorithm for real matrices. But we were especially interested in matrices whose coeffients belong to some other fields. For instance, the rank over $\mathbb{Z}_2$ matrices permits the computation of the number of connected components of a digital image. In Neurobiology, this technique can be used to compute the number of synapses in a neuron (see [2] for details). This limitation arises since *HMA* derives from earlier formalizations limited to concrete types, such as $\mathbb{R}$, $\mathbb{C}$ and $\mathbb{R}^n$. Many results presented in *HMA* are ported from J. Harrison's work in HOL Light [1], where most theorems are proven only for $\mathbb{R}^n$. Another interesting application is the computation of determinants of $\mathbb{Q}$ matrices: commercial software performs such computations wrong (see [12]) in cases that could be critical in cryptology.

J. Hölzl et al. [4] improved significantly the *HMA*. They presented a new hierarchy of spaces based on type classes to represent the common structures of Multivariate Analysis, such as topological spaces, metric spaces and Euclidean spaces. This improvement showed the power of Isabelle's type system. Some limitations still remain; for instance, most properties about vector spaces are only demonstrated in *HMA* over real vector spaces, impeding us from working with matrices whose elements belong to other fields. Generalizing the results in *HMA* is a known problem but has not been tackled. J. Harrison already pointed it out in his work [1]: "many proofs are *morally the same* and it would be appealing to be able to use similar technology to generalize the proofs". J. Avigad also found this limitation when working with *HMA* in his formalization of the Central Limit Theorem [3]; he said that some concepts "admit a common generalization, which would unify the library substantially".

This short paper presents a work in progress which aims at being the foundation stone to get such a generalization. The final aim would be to generalize the library as far as possible. As work done, we present the generalizations and the methodology that permitted us to prove the Gauss-Jordan algorithm over matrices whose elements belong to an arbitrary field.

## 2 Generalization of HMA

Mathematical structures presented in *HMA* are defined by means of *type classes*; type classes are provided by Isabelle and have great advantages: they allow to organize polymorphic specifications, to create a hierarchy among different classes, to provide instances, to produce a simple sintax and to simplify proofs thanks to the Isabelle type inference mechanism. A type class $C$ specifies assumptions $P_1, \ldots, P_k$ for constants $c_1, \ldots, c_m$ (that are to be overloaded) and may be based on other type classes $B_1, \ldots, B_n$. Only one type variable $\alpha$ is allowed to occur in the type class specification. Hence, if we want to prove properties of arbitrary vector spaces (where two type variables appear), we have to use locales instead.

*Locales* are an Isabelle approach for dealing with parametric theories and they are specially suitable for Abstract Algebra as they allow to talk about carriers, sub-structures and existence of structures. On the other hand, code generation within locales with assumptions essentially does not work. Locales enable to prove theorems abstractly, relative to sets of assumptions. These theorems can then be used in other contexts where the assumptions themselves, or instances of the assumptions, are theorems. This form of theorem reuse is called *interpretation*. Locales generalize interpretation from theorems to conclusions, enabling the reuse of definitions and other constructs that are not part of the specifications of the locales.

We are on the borderline: our work requires to use abstract structures such as vector spaces or modules (we have to use locales) but we aim to preserve the executability (code generation). Our proposal is to work with a mix between locales and type classes: every possible lemma is generalized to newly introduced locales, but lemmas required in type classes are kept (because they belong there, or because they are obtained thanks to interpretation of the corresponding abstract locale).

## 2.1 An example of generalization

Let us illustrate the previous methodology with an example. A key lemma in *HMA* is the one which states the link between matrices and linear maps:

```
theorem matrix_works:
assumes "linear f"
shows "matrix f *v x = f (x::real^'n)"
```

It is stated for linear maps between real vector spaces. The *linear* predicate in the premise is introduced by the following locale definition:

```
locale linear = additive f for f :: "'a::real_vector ⇒ 'b::real_vector"
+  assumes scaleR: "f (scaleR r x) = scaleR r (f x)"
```

One parameter is only required: a map $f$. In the heading, the type of $f$ is fixed as a map between two real vector spaces (*real_vector* class). In order to generalize it to *arbitrary* vector spaces over the same field, we propose the following definition:

```
locale linear = B: vector_space scaleB + C: vector_space scaleC
  for scaleB :: "('a::field ⇒ 'b::ab_group_add ⇒ 'b)" (infixr "*b" 75)
  and scaleC :: "('a ⇒ 'c::ab_group_add ⇒ 'c)" (infixr "*c" 75) +
  fixes f :: "('b ⇒ 'c)"
  assumes cmult: "f (r *b x) = r *c (f x)"
  and add: "f (a + b) = f a + f b"
```

This new locale has three parameters, instead of one: the scalar multiplications *scaleB* and *scaleC*, which fix both the vector spaces and the field, and the map $f$. Now we can interpret $\mathbb{F}^n$ (where $\mathbb{F}$ is a field) as a vector space over $\mathbb{F}$ and prove the linear interpretation for $\mathbb{F}^n$ (the corresponding linear map is the multiplication of a matrix by a vector):

```
interpretation vec: vector_space "op *s :: 'a::field ⇒ 'a^'b ⇒ 'a^'b"
interpretation vec: linear "op *s" "op *s" "(λx. A *v (x::'a::field^_))"
```

After reproducing in the new locale the lemmas involved in the proof, we prove the generalized version. Note the differences between both statements:

```
theorem matrix_works:
assumes "linear (op *s) (op *s) f"
shows "matrix f *v x = f (x::'a::field^'n)"
```

## 2.2 The Generalization of the Gauss-Jordan algorithm

Our aim is to generalize the Gauss-Jordan algorithm to generate verified code for matrices with elements belonging to a generic field. The algorithm itself just requires type classes (the *field*) so code generation will work; nevertheless, proving its correctness needs generalizations of properties using locales. In Section 2.1, we have shown an example of how to carry out this generalization. As Harrison pointed out [1], in many cases the proof is essentially the same. However, the procedure is not immediate and almost every demonstration involves subtle design decisions: introduce new locales, syntactic details, interpretations inside the lemma to reuse previous facts, change the types properly and so on. In broad terms, we have carried out four kinds of generalizations in the *HMA* to achieve verified execution over matrices with elements belonging to a field:

1. Lemmas involving real vector spaces (a type class) are generalized to arbitrary vector spaces (a locale).
2. Lemmas involving Euclidean spaces (a type class) are generalized to finite-dimensional vector spaces (a locale).
3. Lemmas involving real matrices are generalized to matrices over any field (thanks to the previous two points).
4. Lemmas about determinants of matrices with coefficients in a real vector space are proven for matrices with coefficients in a commutative ring.

In *HMA* the first time that the notion of a finite basis appeared was in the *euclidean_space* class. Now, we have introduced a new locale *finite_dimensional_vector_space* and generalized several proofs from the *euclidean_space* class to that locale. Thanks to those generalizations, some lemmas that were stated in *HMA* only over real matrices are now proven over more general types. Let us take a look at the following lemma, which claims that a matrix is invertible iff its determinant is not null. The following version is the original one available in *HMA*, stated for integral domains:

```
lemma det_identical_rows:
  fixes A :: "'a::linordered_idom^'n^'n"
  assumes ij: "i ≠ j" and r: "row i A = row j A"
  shows "det A = 0"
proof-
  have tha: "⋀(a::'a) b. a = b ⟹ b = - a ⟹ a = 0"  by simp
  have th1: "of_int (-1) = - 1" by simp
  let ?p = "Fun.swap i j id"
  let ?A = "χ i. A $ ?p i"
  from r have "A = ?A" by (simp add: vec_eq_iff row_def Fun.swap_def)
  then have "det A = det ?A" by simp
  moreover have "det A = - det ?A" by (simp add: det_permute_rows[OF
    permutes_swap_id] sign_swap_id ij th1)
  ultimately show "det A = 0" by (metis tha)
qed
```

The original statement comes from Harrison's formalization [1], where the lemma is demonstrated over real matrices. The previous proof follows the one presented in most of the literature. Essentially, in the proof it is deduced that $\det A = -\det A$ and thus $\det A = 0$. But such a property does not hold in rings which characteristic is 2 (such as $\mathbb{Z}_2$). For instance, in [6] the statement is presented for commutative rings but it is proven without taking into account rings with characteristic 2. The same appears in [7], but the author warns that the demonstration fails in the case of $\mathbb{Z}_2$ matrices. To generalize the result to an arbitrary ring, we had to change totally the proof and work over permutations.[1]

Not only change some proofs, sometimes we have to introduce new definitions. For instance, to multiply a matrix by a scalar. *HMA* works with real matrices, so the next operation is used: `(op *`$_R$`)::real ⇒ 'a ⇒ 'a`. In the generalization, we would like to multiply a matrix of type `'a^'n^'m` by an element of type `'a`. We cannot use `(op *`$_R$`)` to do that. The most similar operation presented in *HMA* is: `(op *s)::'a ⇒ 'a^'n ⇒ 'a^'n`.

We cannot reuse it because is thought to multiply a vector (and not a matrix) by a scalar. Then, we define the multiplication of a matrix by a scalar as follows:

```
definition matrix_scalar_mult :: "'a ⇒ 'a^'n^'m ⇒ 'a^'n^'m"
(infixl "*k" 70) where "k *k A ≡ (χ i j. k * A $ i $ j)"
```

The statements for the real matrix version and the general one are different:

```
lemma scalar_matrix_vector_assoc:
  fixes A :: "real^'m^'n"
  shows "k *R (A *v v) = k *R A *v v"

 lemma scalar_matrix_vector_assoc:
  fixes A :: "'a::field^'m^'n"
  shows "k *s (A *v v) = k *k A *v v"
```

[1] We followed the proof presented in `http://hobbes.la.asu.edu/courses/site/442-f09/dets.pdf`

Some other particularities arose in the generalization. For instance, we had to completely change other demonstration: the *row rank* and the *column rank* of a matrix are equal. We had followed an elegant proof but only valid for real matrices, see [9]. We based its generalization on the output of the Gauss-Jordan algorithm (a reduced row echelon form) following [11]. This change forced us to completely reorganize the files of our development. Another example arises in systems of linear equations: in the real field there could be infinite solutions, but in other fields such as $\mathbb{Z}_2$ there is always finitely many solutions.

Finally, we have generalized more than 2500 lines of code: about 220 theorems and 9 definitions, introducing 6 new locales, 3 new sublocales and 8 new interpretations. The generalized version of the Gauss-Jordan formalization was published in the AFP [10]. Moreover, the generalizations are useful for another contribution of ours: the Rank-Nullity Theorem [8].

## 3 Conclusions

The generalization of *HMA* is useful and desirable, but doing it can be overwhelming at a first glance. The process can be partially automated with suitable scripts, but the full goal cannot be discharged automatically and it requires to make some design decisions. The careful combination of locales, type classes and interpretations has been shown to be a sensible methodology. A remarkable number of proofs have been reused in this way. This contribution shows that the aim is feasible and the generalization has served for our purposes of executing a verified version of the Gauss-Jordan algorithm over fields such as $\mathbb{Z}_2$ and $\mathbb{Q}$.

## References

1. Harrison, J.: The HOL Light Theory of Euclidean Space. J. Autom. Reasoning. Vol. 50-2. pp. 173–190 (2013)
2. Heras, J. et al.: Towards a certified computation of homology groups for digital images. CTIC 2012. LNCS, vol. 7309. pp. 49–57. Springer (2012)
3. Avigad, J., Hölzl, J. and Serafin, L.: A formally verified proof of the Central Limit Theorem. CoRR, (2014)
4. Hölzl, J., Immler, F. and Huffman, B.: Type Classes and Filters for Mathematical Analysis in Isabelle/HOL. ITP 2013. LNCS, vol. 7998. pp. 279–294. Springer (2013)
5. `http://www.unirioja.es/cu/jodivaso/Isabelle/Gauss-Jordan-2013-2/`
6. Axler, S.: Linear Algebra Done Right. Second Edition. Springer (2004)
7. Strang, G.: Introduction to Linear Algebra. Wellesley - Cambridge Press (2009)
8. Aransay, J. and Divasón, J.: Rank-Nullity Theorem in Linear Algebra. AFP (2013)
9. Mackiw, G.: A Note on the Equality of the Column and Row Rank of a Matrix. Mathematics Magazine, Vol. 68-4. pp. 285–286 (1995)
10. Aransay, J. and Divasón, J.: Gauss-Jordan Algorithm and its Applications. AFP (2014)
11. `http://www.math4all.in/public_html/linearalgebra/chapter3.4.html`
12. Durán, A. J., Pérez, M. and Varona, J. L.: The Misfortunes of a Trio of Mathematicians Using Computer Algebra Systems. Can We Trust in Them? Notices of the AMS. Vol. 51-10. pp. 1249–1252 (2014)