TOWARDS A VERIFIED SMITH NORMAL FORM ALGORITHM IN ISABELLE/HOL

JOSE DIVASÓN AND JESÚS ARANSAY

Abstract. In this note we report on a project to get a verified program to compute the Smith normal form of a matrix in Isabelle/HOL. The presented approach tries to tackle the problems that arise in an environment without dependent types as well as we try to reuse previous developments, keeping also the focus of the formalization on its full generality.

Introduction

A matrix over a Bézout domain is in Smith normal form, from here on *SNF*, if its diagonal elements α_i satisfy $\alpha_i|\alpha_{i+1}$ and the rest of elements of the matrix are zeros. Over such a structure, there exists an algorithm to transform a matrix A into its corresponding SNF S by means of invertible matrices, i.e., there exist invertible matrices P and Q such that $S = PAQ$. This well-known canonical form possesses many applications, such as the computation of determinants and solving a system of diophantine equations. It is also useful for computing the homology of a chain complex and of a simplicial complex. More generally, it is useful to compute persistent homology, which can be applied to process big volume of data. Thus, it is interesting to apply formal methods and get a verified algorithm to compute this form, trying to minimize bugs and errors that can cause important losses. In this work, we present a work in progress to formalize the SNF in Isabelle/HOL.

Preliminaries

Isabelle is a generic theorem prover which has been instantiated to support different object-logics. The most widespread of them is HOL. Isabelle's version of HOL (usually called Isabelle/HOL) corresponds to Church's simple type theory extended with polymorphism, Haskell-style type classes and type definitions. The HOL Analysis library (from here on, *HA*) is a huge Isabelle library which contains many theoretical results in mathematical fields such as Analysis, Topology and Linear Algebra. Its vector representation is based on the ideas by Harrison [6], where a vector $v \in \mathbb{R}^n$ is represented as a function of type $\alpha \Rightarrow \mathbb{R}$ where α is a type with n elements, that is, a finite type of cardinality n. Similarly a matrix $A \in \mathbb{R}^{n\times m}$ is represented by a function of type $\alpha \Rightarrow \beta \Rightarrow \mathbb{R}$ where β is a finite type with m elements. However, a subject that had not been explored either in the HA library (or any HOL prover) was to provide an executable implementation of such a representation. To this end, in our previous work [1] we developed a framework where algorithms over matrices can be formalized, executed, refined, and coupled with their mathematical meaning. The basic idea of the framework comes from the data refinement strategy: soundness of algorithms

This work is funded by the Spanish projects MTM2014-54151-P and MTM2017-88804-P.

should be proven in an abstract type, where proofs are feasible but does not achieve a good performance, or even prevent code execution. Execution is carried out in other more efficient concrete structure, which is connected to the abstract representation by means of morphisms. In our case, we provide a refinement to immutable arrays, which correspond to *Vector* in SML and to *IArray.array* in Haskell. This representation defines polymorphic vectors, immutable sequences with constant-time access, and thus, an efficient representation for vectors and matrices. Code from the formalizations using this framework can be exported to functional languages, such as SML and Haskell, by means of the Isabelle code generation tool. This framework was successfully applied to the formalization of various linear algebra algorithms, such as the Gauss-Jordan algorithm [1] and the echelon form [2]. However, there is no formalization of the SNF or any result involving homology, neither using this framework nor any other representation in Isabelle. It does exist a formalization in Coq [3].

Towars a verified computation of the Smith Normal Form

Most of the algorithms to compute the SNF of a matrix are based on submatrices [9]. Unfortunately, submatrices are a delicate issue in the HA library. Let A be an $n \times n$ integer matrix. Such a matrix A would be modeled in the HA library by means of a function of type $\alpha \Rightarrow \alpha \Rightarrow \mathbb{Z}$, where α is a finite type of cardinality equal to n. Let us say that we want to obtain the $(n-1) \times (n-1)$ submatrix from the first element of A, that is, getting rid of the last row and last column. Since Isabelle does not feature dependent types, we cannot use the size of the matrix in the definition. Indeed, we cannot generate a type of the desired cardinality, that is, it is not possible to define a function of type $\beta \Rightarrow \beta \Rightarrow \mathbb{Z}$ imposing cardinality of β to be $n-1$. To sum up, the output type of the function submatrix clearly depends on the input term (the original matrix), which is not expressible in Isabelle/HOL as it does not allow dependent types. A possible workaround is to complete with zeros the deleted elements, and model the output also as a matrix of the same dimension, that is, a function of type $\alpha \Rightarrow \alpha \Rightarrow \mathbb{Z}$. A similar approach of filling with zeros was already introduced by Obua on its formalization of bounds for real linear programs [8]. However, this approach has some drawbacks: we are not really representing a submatrix and internally performance will decrease, one just has to think what happens when taking a small submatrix of a very big matrix. This makes the formalization of the SNF to be harder than usual, and even if one achieves it, performance would be poor due to the overhead produced by the workaround of filling with zeros. As an alternative, we propose the following strategy.

(1) Change completely of framework and representation, and implement an algorithm to compute the SNF of a matrix in the new library developed by Thiemann and Yamada [10]. Soundness of the algorithm would be proved in it.
(2) Define in the HA library the basic concepts of homology and the definition of SNF.
(3) Connect both libraries and generate statements in both worlds, by means of the lifting and transfer package as well as the use of local type definitions [7].

Thiemann and Yamada already faced the problem of working with submatrices when formalizing Jordan normal forms of matrices, a kind of forms whose construction is done by means of block matrices. As a solution, they propose a new matrix representation, from here on *JNF*, which is indeed an abstraction of the HA representation, but flexible for dimensions. A vector $(v_0, \ldots, v_{n-1})$ is represented by a pair (n, v), where n is the dimension and v the

characteristic function (from natural numbers to the type of the elements of the vector), i.e., $v\ i = v_i$. They provide a similar representation for matrices, based on a triple (n, m, f) with the number of rows, number of columns and the characteristic function for a matrix. They prove again many properties of linear algebra and matrices based on such a representation, and they indeed connect it to immutable arrays using our libraries to be able to export efficient code. This new library is far from having all theorems presented in the HA library, but it is growing nowadays and was used successfully to provide executable algorithms for computing Jordan blocks of matrices. Our approach would consist of using this new library to formalize the SNF: there won't be problems with submatrices since they are easily supported by the library, and indeed, they are already defined in the JNF development. However, many results are present in the HA library, and indeed our framework is based on it, so formalizing the SNF with JNF framework would make it isolated. Moreover, we aim to formalize many results of homology based on the HA library, but then we could not connect it with the computable side of a formalized SNF, since it would have been done in other library. We propose the following solution: use the lifting and transfer package to connect both developments. It is relatively easy to do a proof in the HA world based on a result in JNF: from a HA-statement involving vectors or matrices, by means of transfer rules we can obtain the corresponding statement in JNF, prove it in such a world and then return the result to HA, since we already know the type dimension of the vector and matrices involved. However, the other way is not that easy and at some point it would be necessary for our development. As we have said, Isabelle does not allow dependent types. Then, after some technical work it would also be possible to transfer a statement from a JNF statement to a HA-statement, but possibly with an extra assumption in the lemmas relating a type with the dimension: $n = \mathit{CARD}(\alpha)$. One example of this can be seen in the following lemma, obtained from a development of the Berlekamp–Zassenhaus factorization algorithm for integer polynomials [5]. It just states that the output of Berlekamp's algorithm to factorize polynomials modulo a prime p is correct. Let us remark that p is demanded to be a prime number equal to the cardinality of a finite type.

assumes *finite_field_factorization′* (*ff_ops* p) $g = (c, gs)$
  and $p = \mathit{CARD}(\alpha :: \mathit{prime\_card})$ ***and*** `(* More premises *)`
shows $\mathit{unique\_factorization}_p\ f\ (c, \mathit{mset}\ fs)$

The challenge is to get rid of this premise, being substituted by *prime* p in that case. This was impossible before, but now we can do it by means of Isabelle's recent addition of local type definitions, a soundness extension to Isabelle/HOL's logic. Fortunately, this approach and the necessary connection between both libraries were already formalized in the mechanized proof of the Perron–Frobenius theorem [4], were some results were proven in HA, transferred to JNF representation, proved some properties there and finally *untransferred* the final result to HA. Our idea would be similar. We aim to take advantage of the best of each side: theorems and proofs involving homology would be proven in the HA-world to reuse existing results and our framework, the SNF would be defined and proved in the JNA world, where it is possible to work easily with submatrices, and finally all the results and soundness would be put together by means of the lifting and transfer package, the existing connection in the Perron–Frobenius development [4] and the use of local type definitions. This way, the homology results would be stated in the HA library whereas the SNF algorithm would be

implemented (and executed) in the JNF library, but its soundness in both worlds. Then the mathematical connection between homology and the output of the SNF would be preserved. In addition, thanks to this approach we can follow a similar strategy to our previous work of the echelon form of a matrix [2] to prove the existence of the SNF in general over any Bézout domain, and then later derive an executable algorithm parametrized by executable operations to compute Bézout coefficients. Then, our formalization would not be limited to integer matrices, but it would also allow other structures such as polynomial matrices. The idea of using functions as parameters in the algorithm is very interesting, since it could also allow us to formalize at once different versions (we impose some conditions for such parameters, but that specification can be satisfied by different implementations). For instance, we could parameterize the algorithm by a pivot function, to tackle the different ways of selecting pivots in each step. Moreover, also thanks to our framework and the connection to HA theorems, it seems to be feasible to prove the uniqueness of this normal form with the presented strategy.

Conclusions

We have presented an approach to verify the SNF in Isabelle/HOL, and then, a way to get verified algorithms to compute homology groups. Although the approach is far from being trivial and involves different techniques, it seems a feasible way to avoid the limitations of the existing libraries in an environment without dependent types. Indeed, a similar approach was successfully used in other contexts, such as the formalization of the Perron–Frobenius theorem and the Berlekamp–Zassenhaus algorithm.

References

[1] J. Aransay and J. Divasón. Formalisation in higher-order logic and code generation to functional languages of the Gauss-Jordan algorithm. *J. Funct. Program.*, 25, 2015.
[2] J. Aransay and J. Divasón. Formalisation of the computation of the echelon form of a matrix in Isabelle/HOL. *Formal Asp. Comput.*, 28(6):1005–1026, 2016.
[3] G. Cano, C. Cohen, M. Dénès, A. Mörtberg, and V. Siles. Formalized linear algebra over Elementary Divisor Rings in Coq. *Logical Methods in Computer Science*, 12(2), 2016.
[4] J. Divasón, S. J. C. Joosten, O. Kuncar, R. Thiemann, and A. Yamada. Efficient certification of complexity proofs: formalizing the Perron-Frobenius theorem. In *Proceedings of the 7th Conference on Certified Programs and Proofs*, pages 2–13, 2018.
[5] J. Divasón, S. J. C. Joosten, R. Thiemann, and A. Yamada. A formalization of the Berlekamp-Zassenhaus factorization algorithm. In *Proceedings of the 6th Conference on Certified Programs and Proofs*, pages 17–29, 2017.
[6] J. Harrison. The HOL Light Theory of Euclidean Space. *J. Autom. Reasoning*, 50(2):173 – 190, 2013.
[7] O. Kuncar and A. Popescu. From Types to Sets by Local Type Definitions in Higher-Order Logic. In *Proceedings of the 7th Interactive Theorem Proving International Conference*, pages 200–218, 2016.
[8] S. Obua. Proving Bounds for Real Linear Programs in Isabelle/HOL. In *Proceedings of the Theorem Proving in Higher Order Logics International Conference*, pages 227–244, 2005.
[9] A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2000.
[10] R. Thiemann and A. Yamada. Formalizing Jordan normal forms in Isabelle/HOL. In *Proceedings of the 5th Conference on Certified Programs and Proofs*, pages 88–99, 2016.

Universidad de La Rioja, Edificio CCT – C/Madre de Dios 53, 26006 Logroño, La Rioja, Spain.
E-mail address: {jose.divason,jesus-maria.aransay}@unirioja.es