Adventures in the (not so) Complex Space [*]

Emilio Jesús Gallego Arias Pierre Jouvelot

MINES ParisTech, PSL Research University, France

mailto:e+coq2015@x80.org

Abstract

We report on the progress of a constructive mechanization for a small subset of signal processing theory, built upon the SSREFLECT and MATHCOMP libraries.

The development was started to provide mechanized semantics for audio programming languages. Currently, we have formalized several standard properties of the Discrete Fourier Transform, such as its unitary matrix form and its power and convolution theorems. Future goals include transfer functions and *constant overlap-add* processing.

At the workshop, we aim to discuss the needs and limits of our current approach, surveying some mathematical concepts not covered by existing libraries, and similar efforts in other frameworks and theorem provers.

1. Introduction and Motivation

A key component in audio programming and analysis is the Discrete Fourier Transform (DFT). Its efficient implementation — the Fast Fourier Transform — has made practical a myriad of sound-processing algorithms, from convolution-based filtering to envelope extraction.

The need for a mechanized treatment of the DFT and its properties arose in our quest to develop a mechanized semantics for Faust [12], a functional stream-based audio programming language.

Faust's time-domain semantics largely correspond to well-understood notions in the synchronous programming paradigm [3, 2]. However, frequency-domain aspects — central in audio — seem less explored in the literature. Thus, a natural preliminary step was to attempt the mechanization of basic DFT and DSP (Digital Signal Processing) theory.

We have chosen as a reference text the excellent book series by J.O. Smith [18, 17, 19]. Among other nice qualities — like detailed paper proofs — the series is specific to audio applications, reflecting very well the particular, domain-specific considerations of audio DSP. Additionally, the author has ample experience with Faust and stream-based functional audio programming [16].

The first book in the series, "The Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications", presents the basic theory about the DFT: linearity, basis, convolution, and power conservation.

Such theorems are far from mathematically challenging, notwithstanding that they pose a few difficulties in the mechanized constructive setting. In particular, the basic theory of the DFT relies on finite and infinite series, linear algebra, trigonometry, and complex analysis.

The first proof in the book is Euler's formula: $e^{ix} = \cos x + i \sin x$. Indeed, in DSP trigonometry and complex analysis are pervasive, and most definitions crucially depend on complex exponentiation. Thus, the logical path to mechanization seems to require a library supporting complex analysis, such as [8, 5, 13].

However, we soon realized that good support for advanced linear algebra was maybe even more crucial. For that domain, our preferred choice is the linear algebra library of MATHCOMP [11].

Unfortunately, integration of MATHCOMP and constructive complex analysis doesn't seem immediate, due to different definitions of key algebraic structures. A combination of MATHCOMP with classical analysis works, and seemed the most viable choice, even if at the cost of abandoning hopes for a constructive, axiom-free development.

But once we were more familiar with standard DSP practice, it became apparent to us that "true" complex numbers were not a hard requirement for the key parts of the discrete transform theory. Indeed, the SSREFLECT constructive algebraic numbers could pack enough power for our needs. [1] This amounts to work in pre-Hilbert spaces, which seems enough for our case.

Indeed, libraries for algebraic numbers, big operators, and matrix libraries [4, 10] have made our job straightforward, to the point that it has felt like an exercise. It is also remarkable how close our proofs are to the pen and paper versions.

Most of our time was spent in proving some technical lemmas about primitive roots of the unity, and by finding a convenient definition of inner product of (algebraic) vectors. Another remarkable fact of the library is that we didn't use induction in the whole development, "all the required induction was already contained in the provided lemmas".

In the rest of the abstract, we give a quick overview of the development, to briefly conclude with some thoughts on related and planned work.

An version of the code is available at `https://www.cri.ensmp.fr/people/gallego/coq-2015/dft.v`; we warn the reader that it is quite in-flux, we expect it to significantly change in the next weeks. We also hope that the expert reader will excuse our lack of knowledge about the vast MATHCOMP libraries; given its sheer size it could be well the case that we are redefining already present notions nor using it in an optimal way.

2. The Basic Ingredients

Inner Products, Normed Vector Spaces We work over the algebraic numbers $\mathcal{C}$. For any $x \in \mathcal{C}$, $x = \Re x + i\Im x$, where $i^2 = -1$. The conjugation of a number is given by $\overline{x + iy} = x - iy$.

Assume $x, y \in \mathcal{C}^n$, elements of the vector space obtained by n copies of $\mathcal{C}$. We will identify $\mathcal{C}^n$ with time-sampled signals of n samples. Then the inner product of x and y is $\langle x, y\rangle = \sum_{m=0}^{n-1} x(m)\overline{y(m)}$, and a norm is then defined as $\|x\| = \sqrt{\langle x, x\rangle}$.

[*] Research funded by the ANR FEEVER project.

[1] Indeed, in a presentation preceding this work, the first author was asked whether the algebraic numbers would suffice. We wrongly responded no; however there are still corners of the theory that may be difficult to capture without working in a complete metric space.

We write $x \perp y$ for $\langle x, y\rangle = 0$, and we say that x and y are orthogonal.

We assume our vector spaces to be cyclic, that is, for $x \in \mathcal{C}^n$ we perform indexing modulo n. We also have a shift operator, $\textsc{Shift}_k\, x(n) = x(n-k \mod n)$.

Roots of the Unity A n-root of the unity is any z such that it is a solution of the polynomial $z^n = 1$. Informally, a primitive root of the unity is a generator of all the other roots. Given ω a primitive root, some properties of interest are $\sum_{m=0}^{n-1} \omega^m = 0$, $|\omega| = 1$, etc... Given $s_k \in \mathcal{C}^n$ such that $s_k(n) = \omega^{kn}$, we have $s_k \perp s_l \iff k \neq l$.

The Discrete Fourier Transform We define the DFT as:

$$\textsc{Dft}_k\, x = \sum_{m=0}^{n-1} x(m)\omega^{mk}.$$

The definition in matrix form is also concise and convenient:

```
Definition W := \matrix_(i < n, j < n) ' ω ^+ (i*j).
```

Linearity follows directly, while the shifting lemma:

$$\textsc{Dft}_k(\textsc{Shift}_m\, x) = \omega^{mk}\, \textsc{Dft}_k\, x$$

can be proved in 3 lines.

Convolution The relation of DFT with convolution is a key property; using the efficient form of the DFT (FFT), we can perform convolution — and thus FIR filtering — in $\mathcal{O}(n \log n)$ vs $\mathcal{O}(n^2)$.

Definition 1 (Circular Convolution)**.**

$$(x \circledast y)(k) = \sum_{m=0}^{n-1} x(m)y(k-m)$$

Theorem 1 (Circular Convolution)**.**

$$\textsc{Dft}_k\ (x \circledast y) = \textsc{Dft}_k\, x \cdot \textsc{Dft}_k\, y$$

Commutativity of convolution needs 3 lines of proof; the convolution theorem, 5.

Unitary Property Define the *Hermitian transpose* of a matrix $\mathbf{A}^* \equiv \overline{\mathbf{A}^T}$. A matrix is unitary if $\mathbf{A}^*\mathbf{A} = \mathbf{A}\mathbf{A}^* = \mathbf{I}$. This states that the columns of $\mathbf{A}$ are orthogonal. For the DFT matrix $\mathbf{W}$, we have $\mathbf{W}^*\mathbf{W} = \mathbf{W}\mathbf{W}^* = n\mathbf{I}$, and thus its normalized form $\tilde{\mathbf{W}} = \frac{1}{\sqrt{n}}\mathbf{W}$ is unitary.

The Power Theorem In particular, this implies the power theorem:

$$\langle x, y\rangle = \frac{1}{N}\langle \textsc{Dft}\, x, \textsc{Dft}\, y\rangle$$

of which an easy corollary is Parseval's:

$$\|x\|^2 = \frac{1}{N}\|\textsc{Dft}\, x\|^2.$$

3. What's next?

We are still working on completing and cleaning up the development: improving naming, trying to find the best interfaces for some of our tasks — namely exponentiation modulo, inner products, Vandermonde or unitary matrices — and exploring other related finite transforms like [6]. Studying the relation of our approach to rational trigonometry could be interesting too.

Algebraic Signal Processing The next candidate for mechanization is the Z-transform, a generalization of the Fourier transform that replaces the roots of unity by an arbitrary complex number z. The Z-transform reduces analysis of linear systems to their "transfer functions" and "region of convergence", that is to say, values of z for which the transform converges.

The Z-transform has been recently mechanized in a classical setting [15], allowing the verification of linear systems, although no convolution is included.

Unfortunately for us, the Z-transform is essentially defined as an infinite time summation, which doesn't translate well to the approach we have followed here. Notwithstanding, there exists work in algebraic signal processing [1, 14] that allows the definition of a finitary version of the Z-transform based on polynomial algebras. The concrete details are outside of the scope of this abstract, but the basic idea is to view linear filters as an algebra A, with filtering defining an A-module over the vector space V of signals. In this setting, the values $z \in ROC$ act as generators for the polynomial algebras, with elements in the sequence space ℓ_1 for the infinite time case, etc...

Going to the Infinite It would be very interesting to extend our formalization to the infinite-dimensional case. However, we would lose the great convenience that the big operators and linear algebra libraries offer.

Extending big operators to handle infinite sums is far from trivial, we believe that mainly due to convergence issues. Indeed, in [15] linearity theorems for the Z-transform only hold if $z \in ROC$. A possibility would be to pack convergence proofs in the operator itself, or to extend the base type to include elements capturing divergence, as done in other approaches.

Efficient, Certified Computation It could be worth to try to use existing tools for effective linear algebra [9] to achieve verified computation of the DFT. It would mainly require computation over the algebraic numbers; the rest would be captured by the existing matrix multiplication. Going beyond that and refining the algorithm to the FFT [7] is possible, but likely to be quite challenging.

References

[1] *Algebraic Theory of Signal Processing*. 2015. URL: `https://www.ece.cmu.edu/~smart/research.html` (visited on 04/01/2015).

[2] C. Auger. "Compilation Certifiée de SCADE/LUSTRE". `http://tel.archives-ouvertes.fr/tel-00818169/`. Thèse de Doctorat. Université Paris-Sud, Feb. 2013.

[3] G. Berry. "The foundations of Esterel". In: *Proof, Language, and Interaction, Essays in Honour of Robin Milner*. The MIT Press, 2000, pp. 425–454.

[4] Y. Bertot et al. "Canonical Big Operators". In: *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings*. Vol. 5170. Springer, 2008, pp. 86–101.

[5] S. Boldo, C. Lelay, and G. Melquiond. "Coquelicot: A User-Friendly Library of Real Analysis for Coq". English. In: *Mathematics in Computer Science* (2014), pp. 1–22.

[6] J. C. Brown. "Calculation of a constant Q spectral transform". In: *The Journal of the Acoustical Society of America* 89.1 (1991), pp. 425–434.

[7] V. Capretta. "Certifying the Fast Fourier Transform with Coq". In: *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001, Edinburgh, Scotland, UK, September 3-6, 2001, Proceedings*. Vol. 2152. Springer, 2001, pp. 154–168.

[8] L. Cruz-Filipe, H. Geuvers, and F. Wiedijk. "C-CoRN, the Constructive Coq Repository at Nijmegen". In: *Mathematical Knowledge Management, Third International Conference, MKM 2004, Bialowieza, Poland, September 19-21, 2004, Proceedings*. Vol. 3119. Springer, 2004, pp. 88–103.

[9] M. Dénès, A. Mörtberg, and V. Siles. “A refinement-based approach to computational algebra in COQ”. In: *ITP - 3rd International Conference on Interactive Theorem Proving - 2012*. Vol. 7406. Springer. Princeton, USA: Springer, 2012, pp. 83–98.

[10] G. Gonthier. “Point-Free, Set-Free Concrete Linear Algebra”. In: *Interactive Theorem Proving - Second International Conference, ITP 2011, Berg en Dal, The Netherlands, August 22-25, 2011. Proceedings*. Vol. 6898. Springer, 2011, pp. 103–118.

[11] G. Gonthier, A. Mahboubi, and E. Tassi. *A Small Scale Reflection Extension for the Coq system*. Research Report RR-6455. 2008.

[12] GRAME. *The Faust Project*. 2014. URL: `http://faust.grame.fr/` (visited on 04/01/2015).

[13] C. Lelay. *More than real analysis in Coq*. 2014. URL: `http://www.ens-lyon.fr/LIP/AriC/MSC2014/clelay.pdf`.

[14] M. Püschel and J. M. F. Moura. “Algebraic Signal Processing Theory”. In: *CoRR* abs/cs/0612077 (2006).

[15] U. Siddique, M. Y. Mahmoud, and S. Tahar. “On the Formalization of Z-Transform in HOL”. In: *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*. Vol. 8558. Springer, 2014, pp. 483–498.

[16] J. O. Smith III. *Audio Signal Processing in Faust*. June 8, 2013. URL: `https://ccrma.stanford.edu/~jos/aspf/`.

[17] J. O. Smith III. *Introduction to Digital Filters: with Audio Applications*. W3K Publishing, Oct. 2007.

[18] J. O. Smith III. *Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications*. 2nd. W3K Publishing, Apr. 2007.

[19] J. O. Smith III. *Spectral Audio Signal Processing*. W3K Publishing, Dec. 2011.