Formalization of Infinite Dimension Linear Spaces with Application to Quantum Theory

Mohamed Yousri Mahmoud, Vincent Aravantinos, and Sofiène Tahar

Electrical and Computer Engineering Dept., Concordia University,
1455 De Maisonneuve Blvd. W., Montreal, Canada
{mo_solim,vincent,tahar}@ece.concordia.ca

Abstract. Linear algebra is considered an essential mathematical theory that has many engineering applications. While many theorem provers support linear spaces, they only consider finite dimensional spaces. In addition, available libraries only deal with real vectors, whereas complex vectors are extremely useful in many fields of engineering. In this paper, we propose a new linear space formalization which covers both finite and infinite dimensional complex vector spaces, implemented in HOL-Light. We give the definition of a linear space and prove many properties about its operations, e.g., addition and scalar multiplication. We also formalize a number of related fundamental concepts such as linearity, hermitian operation, self-adjoint, and inner product space. Using the developed linear algebra library, we were able to implement basic definitions about quantum mechanics and use them to verify a quantum beam splitter, an optical device that has many applications in quantum computing.

1 Introduction

Linear algebra is a powerful mathematical tool which is widely used in different engineering areas: digital image processing (where images can be represented as eigenspaces [3]), bioinformatics (where DNA sequences form a vector space [19]), and control systems, e.g., robotics (where the system state is represented as a vector and each operational block as a matrix [2]). Consequently, there exist many computer tools allowing to deal with linear algebra: numerical tools (e.g., Matlab [18]), computer algebra systems (e.g., Maple [1]) and theorem provers (e.g., Coq [16]).

Classically, a *linear space* (or, equivalently *vector space*) is a set paired with two operations (called addition and scalar multiplications) which have to satisfy a particular set of axioms, e.g., closure of the set by these operations, commutativity of addition, or distributivity of scalar multiplication over addition (see, e.g., [4] for details). The concept of *dimension* of a vector space is extremely important: it is a cardinal, which can thus be finite or infinite. The properties of finite-dimension vector spaces can be very different from the ones of infinite-dimension ones. For instance, a finite-dimension linear space always has the same dimension as its dual space, whereas this is not the case in infinite dimension (actually an infinite-dimension linear space always has a smaller dimension as its dual).

In this paper, we present a formalization, in HOL-Light, of complex-valued-function linear spaces. We define the basic types of such linear spaces and prove that they satisfy the axioms of linear spaces. We formalize many concepts such as (linear) operators, inner product, hermitian adjoints, eigenvectors. For all these concepts, we prove basic facts and provide tactics that allow to prove such basic facts in an automated way. We also prove non-basic results such as Pythagorean theorem, Cauchy-Schwarz inequality, or the fact that the eigenvalues of an auto-adjoint operator are real values. Then, we demonstrate the use of our library in practice by applying it to the formalization of basic quantum mechanics concepts. We use this to formalize a quantum beam splitter: a device with two optical inputs and two optical outputs which routes the incoming photons to the output ports [13]. We finally verify that this device preserves energy [14].

To the best of our knowledge, there currently exist only four significant formalizations of linear algebra: two in HOL-Light ([7] and [12]), one in PVS [9], and one in Coq [11]. The three former focus essentially on n-dimensional euclidean and complex spaces, whereas our work generalizes it to (possibly) infinite-dimension vector spaces of complex numbers (more precisely, complex-valued-function spaces). The work in [11] formalizes extensively a chapter of a classical textbook but, as far as we know, it does not handle many other useful concepts like operator algebra, linear operators, hermitian adjoints, eigenvectors or inner product. In a nutshell, the essential difference between this work and ours is that ours is oriented towards applications rather than a systematic formalization of a textbook. Consequently some theorems of purely theoretical interest are proved in [11] but not in ours. On the other hand, we formalized more notions and results that are useful for engineering applications.

The paper is organized as follows. Section 2 presents our HOL-Light formalization of linear algebra. Section 3 shows the usability of our framework by giving a brief summary about quantum mechanics and showing how it can be formalized using our development. It then introduces beam splitters, their formal definition, and the verification that they preserve energy. Finally, Section 4 concludes the paper.

2 Finite/Infinite Dimension Linear Space Formalization

In the following we present our formalization which is a collection of theories consisting in definitions (types, operations, predicates) and theorems over these definitions. This formalization is freely available at [15]. For practical use, most of these theories also come with a dedicated tactic allowing to prove automatically some basic but very useful facts. We believe that this makes our library a practical tool instead of just a set of theorems that can be difficult to manipulate in practice. Indeed, it allows the user to focus on the difficult tasks which involve some complex reasoning while getting rid easily of the simple tasks that are usually a burden to the user of interactive theorem proving.

2.1 Complex Functions Vector Space

In HOL-Light, the current formalization of linear spaces involves only finite real vectors represented by the type $\texttt{real}^{\texttt{N}}$ (i.e., a tuple of N real numbers). Extending this to complex linear spaces is achieved simply by considering the type $\texttt{complex}^{\texttt{N}}$. In order to consider infinite dimension, we take the *function space* of an arbitrary set to complex. This is expressed by the type $\texttt{cfun} = \texttt{A} \rightarrow \texttt{complex}$, where A is a type variable (cfun stands for *c*omplex *fun*ction). This representation allows both for infinite-dimension linear spaces (by taking, e.g., num or real for A) and finite-dimension ones (by taking for A any type with a finite extension). Note that a general formalization would be defined for any field, instead of complex only, however this would require to parameterize the formalization with operations on the corresponding field. This would make the formalization much more complicated for no significant gain, since function spaces over the complex field already cover most of the engineering applications.

We define the linear space operations over the type cfun as follows:

Definition 1.
$\texttt{cfun\_add}\ (\texttt{v}_1 : \texttt{cfun})\ (\texttt{v}_2 : \texttt{cfun})\ : \texttt{cfun} = \lambda\texttt{x} : \texttt{A}.\ \texttt{v}_1\ \texttt{x} + \texttt{v}_2\ \texttt{x}$
$\texttt{cfun\_smul}\ (\texttt{a} : \texttt{complex})\ \texttt{v} = \lambda\texttt{x} : \texttt{A}.\ \texttt{a} * \texttt{v}\ \texttt{x}$

(smul stands for *s*calar *mul*tiplication). These functions just “lift” the corresponding operations over complex numbers to the type cfun. Note that, by convention, all operations dealing with a type t are prefixed with this type (hence every operation dealing with the type cfun starts with the prefix cfun_). One can observe that these definitions match the finite case since, if A is finite, then the above operations correspond to the usual component-wise operations over vectors.

For convenience, we also define the commonly used operations of negation, subtraction and conjugation, as well as the null function:

Definition 2.
$\texttt{cfun\_neg}\ (\texttt{v} : \texttt{cfun})\ : \texttt{cfun} = \texttt{cfun\_smul}\ (-\texttt{Cx}(\&1))\ \texttt{v}$
$\texttt{cfun\_sub}\ (\texttt{v}_1 : \texttt{cfun})\ (\texttt{v}_2 : \texttt{cfun})\ : \texttt{cfun} = \texttt{cfun\_add}\ \texttt{v}_1\ (\texttt{cfun\_neg}\ \texttt{v}_2)$
$\texttt{cfun\_cnj}\ (\texttt{v} : \texttt{cfun})\ : \texttt{cfun} = \lambda\texttt{x} : \texttt{A}.\ \texttt{cnj}\ (\texttt{v}\ \texttt{x})$
$\texttt{cfun\_zero}\ =\ \lambda\texttt{x} : \texttt{A}.\ \texttt{Cx}(\&0)$

where & is the HOL-Light function injecting natural numbers into reals, and Cx injects real numbers into complex numbers.

We can then easily prove that the type cfun with the above operations is a linear space by proving the usual axioms presented in Table 1 (we overload the usual symbols for multiplication, addition, etc. with the above operations for cfun; following HOL-Light notations, % denotes scalar multiplication).

Finally we define the notion of subspace as follows:

Definition 3.
$\texttt{is\_cfun\_subspace}\ (\texttt{spc} : \texttt{cfun} \rightarrow \texttt{bool}) \Leftrightarrow$
$\quad \forall \texttt{x}\ \texttt{y}.\ \texttt{x}\ \texttt{IN}\ \texttt{spc} \wedge \texttt{y}\ \texttt{IN}\ \texttt{spc} \Rightarrow$
$\qquad \texttt{x} + \texttt{y}\ \texttt{IN}\ \texttt{spc} \wedge (\forall\ \texttt{a}.\ \texttt{a}\ \%\ \texttt{x}\ \texttt{IN}\ \texttt{spc}) \wedge \texttt{cfun\_zero}\ \texttt{IN}\ \texttt{spc}$

Table 1. `cfun_add` and `cfun_mul` properties

Property	HOL Theorem
Addition commutativity	$\forall \mathtt{x\ y : cfun.\ x + y = y + x}$
Addition Association	$\forall \mathtt{x\ y\ z : cfun.\ (x + y) + z = x + y + z}$
Left Distributivity	$\forall \mathtt{(a : complex)\ (x : cfun)\ (y : cfun).\ a\ \%\ (x + y) = a\ \%\ x + a\ \%\ y}$
Right Distributivity	$\forall \mathtt{(a\ b : complex)\ (x : cfun).\ (a + b)\ \%\ x = a\ \%\ x + b\ \%\ x}$
Compatibility	$\forall \mathtt{(a\ b : complex)\ (x : cfun).\ a\ \%\ (b\ \%\ x) = (a * b)\ \%\ x}$
Identity Element	$\forall \mathtt{(x : cfun).\ x + cfun\_zero = x}$
Additive Inverse	$\forall \mathtt{(x : cfun).\ x - x = cfun\_zero}$

Around 50 theorems have been proved about this theory. In order to make our formalization easier to use in practice we have developed a tactic `CFUN_ARITH_TAC` which allows to prove many simple facts about the above algebra. Indeed, the axioms of linear spaces are all proved automatically with this tactic, as well as many other theorems. This reduced our formalization from more than 300 lines of code to around 50, thus increasing readability and usability.

2.2 Operators

A very important notion is the one of transformation between vector spaces. Such a transformation is called an *operator*. The type of operators is thus $\mathtt{cop} = (\mathtt{A} \rightarrow \mathtt{complex}) \rightarrow (\mathtt{B} \rightarrow \mathtt{complex})$, for which we define the following standard operations:

Definition 4.
$\mathtt{cop\_add\ (op_1 : cop)\ (op_2 : cop)\ : cop} = \lambda \mathtt{x.\ op_1\ x + op_2\ x}$
$\mathtt{cop\_smul\ (a : complex)\ (op : cop)\ : cop} = \lambda \mathtt{x.\ a\ \%\ op\ x}$

As well as negation, subtraction, conjugate and the null operator which are defined as above (note that the definitions for operators and for complex functions only differ by their type, so that higher-order logic and type polymorphism actually allows us to define general combinators which factorize these definitions; we expanded the use of these combinators for the sake of readability). Moreover, we proved that the set of operators with these operations satisfies all the axioms of a linear space.

The above is very similar to the linear space presented in the previous section, but an essential aspect of operators is the fact that we can also *multiply* them. This multiplication is simply the composition:

Definition 5.
$\mathtt{cop\_mul\ (op_1 : (A \rightarrow complex) \rightarrow (B \rightarrow complex))}$
$\qquad \mathtt{(op_2 : (C \rightarrow complex) \rightarrow (A \rightarrow complex))} = \lambda \mathtt{x.\ op_1\ (op_2\ x)}$

Note that the types of $\mathtt{op_1}$ and $\mathtt{op_2}$ do not need to be the same. Following the conventions applied in HOL-Light for matrix multiplication, this operation is denoted with the infix $**$. Indeed, one can recognize that, when the operator is linear (see next section), then operators amount to matrices in finite dimension.

This multiplication has unusual properties, starting with the fact that it is not commutative. It follows that many results that are intuitively true in other contexts are actually false here. For instance, multiplication is only right-distributive over addition, i.e., the following holds:

Theorem 1. $\forall \mathtt{op}_1\ \mathtt{op}_2\ \mathtt{op}_3.\ (\mathtt{op}_1 + \mathtt{op}_2)\ **\ \mathtt{op}_3 = \mathtt{op}_1\ **\ \mathtt{op}_3 + \mathtt{op}_2\ **\ \mathtt{op}_3$

But the following does not:

$$\forall \mathtt{op}_1\ \mathtt{op}_2\ \mathtt{op}_3.\ \mathtt{op}_3\ **\ (\mathtt{op}_1 + \mathtt{op}_2) = \mathtt{op}_3\ **\ \mathtt{op}_1 + \mathtt{op}_3\ **\ \mathtt{op}_2$$

Still, this multiplication has a lot of useful properties that we have proved in our library. The neutral element (both left and right) of this multiplication is the identity function. For convenience, exponentiation has also been defined (note that, here, the operator should have the same domain and range). In total, around 60 theorems have been proved, most of them automatically using our tactic `COP_ARITH_TAC`.

2.3 Linear Operators

Linear operators are of particular interest in our work. They correspond, in the finite-dimension case, to matrices. This notion is easily formalized as follows:

Definition 6.
$\mathtt{is\_linear\_cop}\ (\mathtt{op} : \mathtt{cop}) \Leftrightarrow$
$\quad \forall \mathtt{x}\ \mathtt{y}.\ \mathtt{op}\ (\mathtt{x} + \mathtt{y}) = \mathtt{op}\ \mathtt{x} + \mathtt{op}\ \mathtt{y}\ \wedge \forall \mathtt{a}.\ \mathtt{op}\ (\mathtt{a}\ \%\ \mathtt{x}) = \mathtt{a}\ \%\ (\mathtt{op}\ \mathtt{x})$

Linearity is a powerful property which allows to prove some new properties, in particular about multiplication. For instance, in the case of linear operators, left-distributivity now holds:

Theorem 2. $\forall \mathtt{op}_1\ \mathtt{op}_2\ \mathtt{op}_3.\ \mathtt{is\_linear\_cop}\ \mathtt{op}_3 \Rightarrow$
$\quad \mathtt{op}_3\ **\ (\mathtt{op}_1 + \mathtt{op}_2) = \mathtt{op}_3\ **\ \mathtt{op}_1 + \mathtt{op}_3\ **\ \mathtt{op}_2$

So does the associativity of scalar multiplication on the right of a multiplication:

Theorem 3. $\forall \mathtt{z}\ \mathtt{op}_1\ \mathtt{op}_2.\ \mathtt{is\_linear\_cop}\ \mathtt{op}_1 \Rightarrow$
$\quad \mathtt{op}_1\ **\ (\mathtt{z}\ \%\ \mathtt{op}_2) = \mathtt{z}\ \%\ (\mathtt{op}_1\ **\ \mathtt{op}_2)$

Around 10 additional theorems were proved that deal with the particular properties of linear operators.

In practice, one often has to prove that a given operator is linear. To do this, many congruence results are very useful and have indeed to be proved. We gathered the simplest ones in the following theorem:

Theorem 4.
$\forall \mathtt{op}_1\ \mathtt{op}_2.\ \mathtt{is\_linear\_cop}\ \mathtt{op}_1 \wedge \mathtt{is\_linear\_cop}\ \mathtt{op}_2 \Rightarrow$
$\quad \mathtt{is\_linear\_cop}\ (\mathtt{op}_1 + \mathtt{op}_2) \wedge \mathtt{is\_linear\_cop}\ (\mathtt{op}_1 * \mathtt{op}_2)\ \wedge$
$\quad \mathtt{is\_linear\_cop}\ (\mathtt{op}_2 - \mathtt{op}_1) \wedge \forall \mathtt{a}.\ \mathtt{is\_linear\_cop}\ (\mathtt{a}\ \%\ \mathtt{op}_1)$

The base cases for `cop_zero` and the identity function have also been proved. Together, these theorems allow to prove the most frequently seen situations dealing with linearity. Since the involved reasoning is often very similar, we have again developed a tactic to deal with such situations automatically: `LINEARITY_TAC`.

Finally, the notion of eigenvalues and eigenvectors are very important both in theory and in many applications:

Definition 7.
$\texttt{is\_eigen\_pair}\ (\texttt{op} : \texttt{cop})\ (\texttt{f}, \texttt{v}) \Leftrightarrow$
$\quad \texttt{is\_linear\_cop op} \Rightarrow \texttt{op f} = \texttt{v}\ \%\ \texttt{f}\ \wedge \texttt{f} \neq \texttt{zerofun}$

Here, `f` is called the *eigenfunction*, and `v` the *eigenvalue*. We then proved some useful properties, in particular, the set of all the eigenvectors of a given eigenvalue constitutes a linear space:

Theorem 5. $\forall\texttt{op. is\_linear\_cop op} \Rightarrow$
$\quad \forall\texttt{z. is\_cfun\_subspace}\ (\{\ \texttt{f}\ |\ \texttt{is\_eigen\_pair op}\ (\texttt{f}, \texttt{z})\ \} \cup \{\texttt{cfun\_zero}\})$

2.4 Inner Product

The inner product is very useful both in theory and in practice, in particular in many engineering applications (e.g., digital communication or quantum optics). Since the type `cfun` depends on a type variable `A`, we cannot provide an implementation of the inner product which works with every possible instantiation of `A`. For instance, if `A` is substituted with `num` then we can provide a definition based on some infinite sum, but if it is substituted with `real` then a suitable notion of integration should be defined. This prevents a general definition of inner product. We thus introduce a predicate asserting whether a given function indeed satisfies the axioms of an inner product and then parameterize our formalization with this predicate:

Definition 8.
$\texttt{is\_inprod}\ (\texttt{inprod} : \texttt{cfun} \rightarrow \texttt{cfun} \rightarrow \texttt{complex}) \Leftrightarrow$
$\quad \forall\ \texttt{x y z.}$
$\qquad \texttt{cnj}\ (\texttt{inprod y x}) = \texttt{inprod x y} \wedge$
$\qquad \texttt{inprod}\ (\texttt{x} + \texttt{y})\ \texttt{z} = \texttt{inprod x z} + \texttt{inprod y z} \wedge$
$\qquad \texttt{real}\ (\texttt{inprod x x}) \wedge \&0 \leq \texttt{real\_of\_complex}\ (\texttt{inprod x x}) \wedge$
$\qquad (\texttt{inprod x x} = \texttt{Cx}(\&0) \Leftrightarrow \texttt{x} = \texttt{cfun\_zero}) \wedge$
$\qquad \forall\texttt{a. inprod x}\ (\texttt{a}\ \%\ \texttt{y}) = \texttt{a} * (\texttt{inprod x y})$

where `real x` states that the complex value `x` has no imaginary part, and `real_of_complex` is a function casting such a complex number into a real one.

Around 20 theorems of the inner product have been proved in our formalization, e.g., distributivity with respect to addition, associativity with respect to scalar multiplication (modulo the conjugate when the scalar multiplication occurs on the left), etc. A particularly interesting property is the injectivity of the inner product seen as a curried function:

Theorem 6. $\forall$`inprod. is_inprod inprod` $\Rightarrow$
  $\forall$`x y. inprod x = inprod y` $\Leftrightarrow$ `x = y`

This is a powerful property which allows, in particular, to prove the uniqueness of a hermitian adjoint (see next section).

From the inner product, we can define orthogonality as follows:

Definition 9. `are_orthogonal inprod u v` $\Leftrightarrow$
  `is_inprod inprod` $\Rightarrow$ `inprod u v = Cx(&0)`

We proved some basic properties about orthogonality like the fact that it is symmetric or that scalar multiplication preserves orthogonality. However, we can prove some more difficult and interesting theorems like, e.g., the Pythagorean theorem:

Theorem 7 (Pythagorean).
$\forall$ `inprod u v. is_inprod inprod` $\wedge$ `are_orthogonal inprod u v` $\Rightarrow$
  `inprod (u + v) (u + v) = inprod u u + inprod v v`

or the existence of an orthogonal decomposition of any vector with respect to another one:

Theorem 8 (Decomposition).
$\forall$ `inprod u v. is_inprod inprod` $\Rightarrow$
  `let proj_v =` $\frac{\texttt{inprod v u}}{\texttt{inprod v v}}$ `in`
  `let orthogonal_component = u - proj_v % v in`
    `u = proj_v % v + orthogonal_component` $\wedge$
    `are_orthogonal inprod v orthogonal_component`

These two theorems play a crucial role in particular when proving the Cauchy-Schwarz Inequality, which has itself essential applications in the error analysis of many engineering systems:

Theorem 9 (Cauchy-Schwarz Inequality).
$\forall$ `x y inprod. is_inprod inprod` $\Rightarrow$
  `norm (inprod x y) pow 2` $\leq$
    `real_of_complex (inprod x x) * real_of_complex (inprod y y)`

where `norm` denotes the norm of a complex number. Note that, even without focusing on the infinite-dimension aspect, this theorem is still a not-so-trivial adaptation of the existing results in HOL-Light, since it extends it to *complex* linear spaces.

2.5 Hermitian Adjoint

A very useful notion of linear operators is the one of hermitian adjoint. It is very important theoretically and has many applications, e.g., in quantum mechanics. This operation generalizes the one of conjugate transpose in the finite-dimension case and we formalize it as follows:

Definition 10.
$\texttt{is\_hermitian}\ \texttt{op}_1\ \texttt{op}_2\ \texttt{inprod} \Leftrightarrow$
$\quad\texttt{is\_inprod}\ \texttt{inprod} \Rightarrow$
$\quad\quad\texttt{is\_linear\_cop}\ \texttt{op}_1 \wedge \texttt{is\_linear\_cop}\ \texttt{op}_2\ \wedge$
$\quad\quad\forall\ \texttt{x}\ \texttt{y}.\ \texttt{inprod}\ \texttt{x}\ (\texttt{op}_1\ \texttt{y}) = \texttt{inprod}\ (\texttt{op}_2\ \texttt{x})\ \texttt{y}$

The relation $\texttt{is\_hermitian}\ \texttt{op}_1\ \texttt{op}_2$ holds if and only if op_2 is the hermitian adjoint of op_1. We use a relation instead of a function because the existence of a hermitian operator cannot be proved in a general way: it depends a lot on the underlying space. In particular, this highlights a big difference between the finite and the infinite dimension case: in finite dimension, one can just take the conjugate transpose of the underlying matrix to obtain the hermitian. But in infinite dimension, this is not as simple as that: there is indeed a notion of transpose operator, but it yields an operator in the *dual space* of the original vector space. If there is an isomorphism between this dual space and the original vector space, then one can obtain a satisfying definition of hermitian, however, in infinite dimension, there is not always such an isomorphism. However, in any case, if there is a hermitian operator, then it is unique, as proved by the following theorem:

Theorem 10.
$\forall\texttt{op}_1\ \texttt{op}_2\ \texttt{op}_3\ \texttt{inprod}.$
$\quad\texttt{is\_hermitian}\ \texttt{op}_1\ \texttt{op}_2\ \texttt{inprod} \wedge \texttt{is\_hermitian}\ \texttt{op}_1\ \texttt{op}_3\ \texttt{inprod}$
$\quad\quad\Rightarrow \texttt{op}_2 = \texttt{op}_3$

We also proved some other properties of the hermitian, such as for instance the symmetry of its relation:

Theorem 11.
$\forall\texttt{inprod}\ \texttt{op}_1\ \texttt{op}_2.$
$\quad\texttt{is\_hermitian}\ \texttt{op}_1\ \texttt{op}_2\ \texttt{inprod} \Leftrightarrow \texttt{is\_hermitian}\ \texttt{op}_2\ \texttt{op}_1\ \texttt{inprod}$

Seeing the hermitian as a function, this proves the usual property that taking the hermitian of the hermitian is the identity.

Finally, we prove some congruence theorems which allow to prove, in many cases, that a given operator is the hermitian of another:

Theorem 12.
$\forall\texttt{inprod}\ \texttt{op}_1\ \texttt{op}_2\ \texttt{op}_3\ \texttt{op}_4\ \texttt{a}.$
$\quad\texttt{is\_hermitian}\ \texttt{op}_1\ \texttt{op}_2\ \texttt{inprod} \wedge \texttt{is\_hermitian}\ \texttt{op}_3\ \texttt{op}_4\ \texttt{inprod}\ \Rightarrow$
$\quad\quad\texttt{is\_hermitian}\ (\texttt{op}_1 + \texttt{op}_3)\ (\texttt{op}_2 + \texttt{op}_4)\ \texttt{inprod}\ \wedge$
$\quad\quad\texttt{is\_hermitian}\ (\texttt{op}_1 - \texttt{op}_3)\ (\texttt{op}_2 - \texttt{op}_4)\ \texttt{inprod}\ \wedge$
$\quad\quad\texttt{is\_hermitian}\ (\texttt{op}_1 * \texttt{op}_3)\ (\texttt{op}_4 * \texttt{op}_2)\ \texttt{inprod}\ \wedge$
$\quad\quad\texttt{is\_hermitian}\ (\texttt{a}\ \%\ \texttt{op}_1)\ (\texttt{cnj}\ \texttt{a}\ \%\ \texttt{op}_2)\ \texttt{inprod}$

Finally, we also provide a more "computational" version of these congruence theorems:

Theorem 13.
$\forall$a b inprod op$_1$ op$_2$ op$_3$ op$_4$ op$_5$.
  is_hermitian op$_1$ op$_2$ inprod $\wedge$ is_hermitian op$_3$ op$_4$ inprod $\wedge$
  is_hermitian (a % op$_1$ + b % op$_3$) op$_5$ inprod $\Rightarrow$
    op$_5$ = cnj a % op$_2$ + cnj b % op$_4$

In total, around 10 theorems were proved about hermitian operators.

2.6 Self-adjoint Operators

We conclude the overview of our library by presenting the notion of self-adjoint operator, which simply denotes operators which are their own hermitian adjoint:

Definition 11. `is_self_adjoint op inprod` $\Leftrightarrow$ `is_hermitian op op inprod`

Once again, we have proved many congruence theorems allowing to deal with most self-adjoint operators that are encountered in proofs. Most of them are similar to the ones for the hermitians, only the case of scalar multiplication should be handled with a little bit of care, since we must require that the scalar is a real number:

Theorem 14.
$\forall$ inprod op a. is_inprod inprod $\wedge$ real a
  $\Rightarrow$ is_self_adjoint(a % op) inprod

Some other results are a less obvious and very useful, for instance:

Theorem 15.
$\forall$ inprod op x y.
  is_inprod inprod $\wedge$ is_linear_op op $\wedge$
  inprod (op x) y = $-$(inprod x (op y)))
    $\Rightarrow$ is_self_adjoint (ii % op) inprod

Proving that a given operator is self-adjoint using all these theorems is such a common task that we have developed a dedicated tactic for it: `SELF_ADJOINT_TAC` [15].

We finally give two examples of non-trivial theorems which involve many of the concepts presented until now. The first one states that any eigenvalue of a self-adjoint operator is real:

Theorem 16.
$\forall$ inprod op. is_inprod inprod $\wedge$ is_self_adjoint op inprod $\Rightarrow$
  $\forall$z. is_eigen_value op z $\Rightarrow$ real z

where `is_eigen_value z` is true if and only if there exists an eigenfunction such that `z` is its corresponding eigenvalue. Another result states that the eigenfunctions of a self-adjoint operator are orthogonal if the corresponding eigenvalues are different:

Theorem 17.
$\forall$ `inprod op` $\mathtt{f_1}$ $\mathtt{f_2}$ $\mathtt{z_1}$ $\mathtt{z_2}$.
  `is_inprod inprod` $\wedge$ `is_self_adjoint op inprod` $\wedge$ $\mathtt{z_1} \neq \mathtt{z_2}$ $\wedge$
  `is_eigen_pair op` $(\mathtt{f_1}, \mathtt{z_1})$ $\wedge$ `is_eigen_pair op` $(\mathtt{f_2}, \mathtt{z_2})$
    $\Rightarrow$ `are_orthogonal inprod` $\mathtt{f_1}$ $\mathtt{f_2}$

This concludes the presentation of our current formalization. In order to show its usefulness, we now give a sophisticated application by formalizing (basics of) Quantum Mechanics and applying this to the verification of a device called a *beam splitter*.

3 Application to Quantum Theory

In this section we briefly introduce quantum mechanics, how it can be mathematically represented using inner product spaces, and how we propose to formalize it using the results of the previous section.

3.1 Quantum Mechanics

It is assumed that the description of any physical system starts with a *state*. From this state, one can obtain the *coordinates* of the system: e.g., the position of a moving particle, or the temperature of a given system. Coordinates are the atomic pieces of information of the system. Being given the state of a system, one can also derive the values of other quantities called *observables*: e.g., the energy of the system. Observables are similar to coordinates except that they are not atomic, i.e., they can be derived from coordinates. In classical physics, the measurement of a system state (and thus observables) and its evolution are deterministic, whereas they are only probabilistic in quantum physics [6]. Consequently, whereas the state of the system is a set of real numbers in classical physics, it is a probability distribution in quantum mechanics. In both cases, coordinates and observables are functions which take the system state as input. However, in classical physics, the output of this function is a real number, but it is a probability distribution in quantum mechanics.

For our concern, the interesting aspect of quantum mechanics is that the involved probability distributions form an infinite-dimension (complex) inner product space: The state of a quantum system can be mathematically represented as a complex-valued function and coordinates (and observables) can be represented by (self-adjoint) operators. In practice, one is very often interested in the expected value of such an observable: This can be represented by the norm canonically associated with the inner product.

We thus have all the tools required to formalize these concepts. Note that we formalize only some basics of quantum mechanics. However, those definitions are sufficient to define formally the quantum system presented in the next section and to do simple verification tasks on it. We start by defining the type `qstate` as an abbreviation for `cfun` (note that this type contains a type variable: this variable can be instantiated differently depending on the considered system). The *space* of the possible values for states is defined as $\texttt{qspace} = (\texttt{qstate} \to \texttt{bool}) \times (\texttt{cfun} \to \texttt{cfun} \to \texttt{complex})$, where the first element of the pair is the considered set of possible states, and the second one is an inner product to be associated with this set. In order to ensure that a given value of type `qspace` indeed represents a valid quantum space, we define the following predicate:

Definition 12.
$\texttt{is\_qspace}\ ((\texttt{vs}, \texttt{inprod}) : \texttt{qspace}) \Leftrightarrow$
$\quad \texttt{is\_cfun\_subspace vs} \wedge \texttt{is\_inprod inprod}$

Being given a space, we can define coordinates and observables: As mentioned above, these are mathematically represented by self-adjoint operators. They thus have the type $\texttt{qstate} \to \texttt{qstate}$. Being given a quantum state space, we have to ensure that an observable (or coordinate) is self-adjoint and that the result of its application remains in the state space. This is achieved by the following predicate:

Definition 13.
$\texttt{is\_observable}\ (\texttt{op} : \texttt{qstate} \to \texttt{qstate})\ ((\texttt{vs}, \texttt{inprod}) : \texttt{qspace}) \Leftrightarrow$
$\quad \texttt{is\_qspace}\ (\texttt{vs}, \texttt{inprod}) \wedge \texttt{is\_self\_adjoint op inprod} \wedge$
$\quad \forall\ \texttt{x}.\ \texttt{x} \in \texttt{vs} \Rightarrow \texttt{op x} \in \texttt{vs}$

Now, verifying a device requires that we formalize a model of it. Mathematically, a device is just a quantum system, we thus formalize this notion. A system is built of a state space, coordinates, and a function describing the evolution of the state. First of all, we should notice that coordinates depend on time, which we consider here to be a real number, so their type is actually $\texttt{coord} = \texttt{time} \to (\texttt{qstate} \to \texttt{qstate})$ (for readability, `time` is defined as an abbreviation of `real`). The evolution of the system is actually fully expressed by the expression of its total energy (called the "Hamiltonian"). Since the total energy is an observable, which also depends on time, it also has the type $\texttt{time} \to \texttt{qstate} \to \texttt{qstate}$. So, finally, the type of quantum systems is defined as:

$$\texttt{qsys} = \texttt{qspace} \times \texttt{coord list} \times (\texttt{time} \to \texttt{qstate} \to \texttt{qstate})$$

To ensure that we have a valid system, we define again a predicate (`qs` stands for *q*uantum *s*ystem, `cs` for *c*oordinate*s*, and `H` for *H*amiltonian):

Definition 14.
$\texttt{is\_qsys}\ (\texttt{qs}, \texttt{cs}, \texttt{H}) \Leftrightarrow$
$\quad \texttt{is\_qspace qs} \wedge \forall \texttt{t} : \texttt{time}.\ \texttt{is\_observable}\ (\texttt{H t})\ \texttt{qs} \wedge$
$\quad \texttt{ALL}\ (\lambda \texttt{c}.\ \texttt{is\_observable}\ (\texttt{c t})\ \texttt{qs})\ \texttt{cs}$

where `ALL P l` is true if and only if every element of `l` satisfies the predicate `P`.

Using all these notions and our library, we could prove the famous uncertainty principle:

Theorem 18 (Uncertainty Principle).
$\forall$`obs1 obs2 ((spc, inprod) : qspace) t qst.`
`is_observable obs1 (spc, inprod)` $\wedge$ `is_observable obs2 (spc, inprod)` $\wedge$
`qst` $\in$ `spc` $\wedge$ `qst` $\neq$ `cfun_zero` $\Rightarrow$

$$\left(\frac{\texttt{expectation inprod qst (commutator op1 op2)}}{\texttt{Cx(\&2)*ii}}\right) \texttt{ pow 2}$$

$\leq$ `real_of_complex (variance inprod qst op1)`
$*$ `real_of_complex (variance inprod qst op2)`

where `expectation inprod qst op` returns the expected value of an operator `op` seen as a statistical measurement in a given state `qst`. This is classically defined in quantum mechanics using the inner product as `inprod qst (op qst)`. Similarly, the variance can be computed using the inner product, which yields the function `variance`. Finally `commutator` $\texttt{op}_1\ \texttt{op}_2 = \texttt{op}_1 * *\texttt{op}_2 - \texttt{op}_2 * *\texttt{op}_1$. We refer the reader to [6] for detailed explanations about the uncertainty principle. Note that the proof of this result makes an essential use of the Cauchy-Schwarz inequality (Theorem 9) and of our automation tactics `LINEARITY_TAC` and `SELF_ADJOINT_TAC`.

This concludes our formalization of quantum mechanics basics. Note that this could not have been done with the current library of linear algebra in HOL Light [8], because of the lack of (complex-valued) function space formalization. Neither could it be developed in Coq using [11] because it lacks many of the notions we used here: operators, inner product, self-adjoint.

In the next section, we present the formalization of a quantum single-mode electromagnetic field, i.e., the inputs and outputs of a beam splitter.

3.2 Single-Mode Electromagnetic Field

A single-mode field is an electromagnetic field with a single resonance frequency. This is the simplest model of a light beam. Such a field constitutes a quantum system according to the definition that we have given above. We should thus specify its coordinates and Hamiltonian (we do not specify the state space in order to keep our formalization general). The coordinates of an electromagnetic field consists in its amount of charges $q(t)$ and the intensity of its flux $p(t)$. In quantum mechanics, operators are usually written with a circumflex, so the quantum versions of these coordinates are written $\hat{p}(t)$ and $\hat{q}(t)$. The Hamiltonian is then defined as:

$$\hat{H}(t) = \frac{\omega^2}{2}\hat{q}(t)^2 + \frac{1}{2}\hat{p}(t)^2$$

where ω is the resonance frequency. In order to keep explicit the resonance frequency, we define a type dedicated to single-mode fields by `sm` $=$ `qsys` $\times$ `real` (`sm` stands for *s*ingle-*m*ode) where the first component is the system itself and the second one is the frequency. Once again we collect in a predicate all the conditions required for a value of type `sm` to represent a valid single-mode field:

Definition 15.

$\texttt{is\_sm}\ ((\texttt{qs}, \texttt{cs}, \texttt{H}), \omega : \texttt{sm}) \Leftrightarrow$
$\quad \texttt{is\_qsys}\ (\texttt{qs}, \texttt{cs}, \texttt{H}) \wedge 0 < \texttt{omega}\ \wedge \texttt{LENGTH cords} = 2\ \wedge$
$\quad \texttt{let p} = \texttt{EL 0 cs and q} = \texttt{EL 1 cs in}$
$\quad \forall \texttt{t} : \texttt{time. H t} = \texttt{Cx}(\frac{\omega^2}{2})\ \%\ ((\texttt{q t})\ \texttt{pow 2}) + \texttt{Cx}(\frac{1}{2})\ \%\ ((\texttt{p t})\ \texttt{pow 2})$

where `EL i l` is the $\texttt{i}^{\texttt{th}}$ element of a list `l`. Here, we assert that the system should indeed be a valid system, that the frequency should be positive and there should be two coordinates. We fix the first coordinate to be the charge and the second one to be the intensity.

Using our library, we can already prove a couple of useful theorems about single mode fields. For instance, we can prove that the Hamiltonian is linear:

Theorem 19.

$\forall \texttt{qs cs H}\ \omega\ \texttt{t. is\_sm}\ ((\texttt{qs}, \texttt{cs}, \texttt{H}), \omega) \Rightarrow \texttt{is\_linear\_cop}\ (\texttt{H t})$

And even that it is self-adjoint:

Theorem 20.

$\forall \texttt{qs cs H}\ \omega\ \texttt{t. is\_sm}\ ((\texttt{qs}, \texttt{cs}, \texttt{H}), \omega) \Rightarrow \texttt{is\_self\_adjoint}\ (\texttt{H t})$

Both theorems were proved automatically by using our tactics `LINEARITY_TAC` and `SELF_ADJOINT_TAC`.

3.3 Beam Splitter

A beam splitter is a generic name for an optical device which takes two input light beams and outputs two other beams. It can route the input beams towards the output in different ways, depending on the type of beam splitter which is considered. For instance, as its name suggests, a typical behavior is to "split" a single input beam, i.e., one can have a configuration where, if there is only one incident beam, then half of the photons are routed towards one output beam, and the other half is routed towards the other one. However other beam splitters can have other behaviors, e.g., beam phase shifting [5]. Note that beam splitters play an important role in some implementations of quantum computers [10], e.g., in [17]. In this section, we provide a general specification for a beam splitter and prove that any device satisfying this specification preserves the energy from the input to the output beams.

Again, we first define a dedicated type for beam splitters. The behavior of a beam splitter, which determines the route of photons, can be modeled by four parameters, given as complex numbers. This yields the following definition:

$$\texttt{bmsp} = \texttt{complex} \times \texttt{complex} \times \texttt{complex} \times \texttt{complex} \times \texttt{sm} \times \texttt{sm} \times \texttt{sm} \times \texttt{sm}$$

The four values of type `sm` represent the two input and two output single-mode fields, respectively. We then define a predicate ensuring that a value of type `bmsp` indeed represents a real beam splitter.

Definition 16.

$$
\begin{array}{l}
\texttt{is\_bmsp}\ (\texttt{b}_1,\texttt{b}_2,\texttt{b}_3,\texttt{b}_4,\texttt{in\_port}_1,\texttt{in\_port}_2,\texttt{out\_port}_1,\texttt{out\_port}_2) \Leftrightarrow \\
\quad \texttt{is\_sm in\_port}_1 \wedge \texttt{is\_sm in\_port}_2 \ \wedge \texttt{is\_sm out\_port}_1 \wedge \texttt{is\_sm out\_port}_2 \\
\quad \wedge \texttt{b}_1 * \texttt{cnj b}_1 + \texttt{b}_2 * \texttt{cnj b}_2 = \texttt{Cx}\ (\&1) \ \wedge \texttt{b}_3 * \texttt{cnj b}_3 + \texttt{b}_4 * \texttt{cnj b}_4 = \texttt{Cx}\ (\&1) \\
\quad \wedge \texttt{b}_1 * \texttt{cnj b}_3 + \texttt{b}_2 * \texttt{cnj b}_4 = \texttt{Cx}\ (\&0) \ \wedge \texttt{cnj b}_1 * \texttt{b}_3 + \texttt{cnj b}_2 * \texttt{b}_4 = \texttt{Cx}\ (\&0) \\
\quad \wedge\ \forall\ \texttt{t} : \texttt{time}. \\
\qquad \texttt{p}_{\texttt{out}_1}\ \texttt{t} = \texttt{b}_1\ \%\ \texttt{p}_{\texttt{in}_1}\ \texttt{t} + \texttt{b}_2\ \%\ \texttt{p}_{\texttt{in}_2}\ \texttt{t} \ \wedge \texttt{q}_{\texttt{out}_1}\ \texttt{t} = \texttt{b}_1\ \%\ \texttt{q}_{\texttt{in}_1}\ \texttt{t} + \texttt{b}_2\ \%\ \texttt{q}_{\texttt{in}_2}\ \texttt{t} \\
\qquad \wedge \texttt{p}_{\texttt{out}_2}\ \texttt{t} = \texttt{b}_3\ \%\ \texttt{p}_{\texttt{in}_1}\ \texttt{t} + \texttt{b}_4\ \%\ \texttt{p}_{\texttt{in}_2}\ \texttt{t} \ \wedge \texttt{q}_{\texttt{out}_2}\ \texttt{t} = \texttt{b}_3\ \%\ \texttt{q}_{\texttt{in}_1}\ \texttt{t} + \texttt{b}_4\ \%\ \texttt{q}_{\texttt{in}_2}\ \texttt{t}
\end{array}
$$

where $\texttt{p}_{\texttt{in}_\texttt{x}}$ and $\texttt{q}_{\texttt{in}_\texttt{x}}$ denote the charge and flux intensity in the x^{th} input beam, respectively, and the same holds with the `out` index for the output beams. The first line ensures that all the involved light beams are indeed single-mode fields. The four following lines impose general constraints on the configuration of the device. Finally, the last four lines provide the relation that holds between the light beams, according to the parameters.

Finally, using our formalization of linear algebra and quantum mechanics, we could prove that any beam splitter is an energy lossless device, i.e., the total energy of input ports is equal to the total energy of output ports. Formally:

Theorem 21.
$\forall\ \texttt{bs. is\_bmsp bs} \Rightarrow \texttt{H}_{\texttt{in}_1} + \texttt{H}_{\texttt{in}_2} = \texttt{H}_{\texttt{out}_1} + \texttt{H}_{\texttt{out}_2}$

where $\texttt{H}_b$ is the Hamiltonian of the light beam b. This result was proved in around 200 lines of HOL-Light proof script, which is quite small for an application requiring so many layers of formalization. P

4 Conclusion

Linear algebra is extremely useful in many engineering disciplines. However the developments currently available in theorem provers do not allow to tackle many of these fields due to the lack of support for the required concepts (function spaces, inner products, self-adjoints, etc.). In particular, in HOL-Light, only euclidean spaces are formalized thus preventing the application to many areas. In this paper, we presented a formalization of linear algebra which targets engineering applications rather than a purely theoretical development. Notably, we tried to emphasize the practical usability by providing tactics which allow to solve many small but commonly-encountered problems. Using this formalization, we were able to define some basic notions of quantum mechanics and to apply it to the verification that any beam splitter is an energy lossless device. In our opinion, this demonstrates that our library is general and practical enough to tackle complex problems that make use of linear algebra.

Furthermore, this work yields a lot of potential future research. We plan first to develop the linear algebra library even more by adding other useful notions of linear algebra: e.g., dual spaces or decomposition according to a basis. We also consider providing implementations of some specific instantiations of the theory presented here, depending on the value of the variable `A` in the type `cfun`. This would yield the development of some specific theories that could be especially useful to particular areas like, e.g., electromagnetic. Finally, our successful experiments with the formalization of quantum mechanics encourages to go further in this direction, by developing a theorem-proving framework that would allow easy but safe verification of quantum optics devices. This would have applications both in the verification of optics-related technologies, and in quantum computer engineering.

References

1. Aladev, V.Z.: Computer Algebra Systems: A New Software Toolbox For Maple. Computer Mathematics Series. Fultus Books (2004)
2. Bakshi, U.A., Bakshi, V.: Modern Control Theory. Technical Publications (2009)
3. Chandrasekaran, S., Manjunath, B.S., Wang, Y.F., Winkeler, J., Zhang, H.: An Eigenspace Update Algorithm for Image Analysis. Graphical Models and Image Processing 59(5), 321–332 (1997)
4. Dettman, J.W.: Introduction to Linear Algebra. Dover Books on Mathematics Series. Dover (1974)
5. Fox, M.: Quantum Optics: An Introduction. Oxford Master Series in Physics. Oxford University Press (2006)
6. Griffiths, D.J.: Introduction to Quantum Mechanics. Pearson Prentice Hall (2005)
7. Harrison, J.: HOL Light: A Tutorial Introduction. In: Srivas, M., Camilleri, A. (eds.) FMCAD 1996. LNCS, vol. 1166, pp. 265–269. Springer, Heidelberg (1996)
8. Harrison, J.: A HOL Theory of Euclidean Space. In: Hurd, J., Melham, T. (eds.) TPHOLs 2005. LNCS, vol. 3603, pp. 114–129. Springer, Heidelberg (2005)
9. Herencia-Zapana, H., Jobredeaux, R., Owre, S., Garoche, P.-L., Feron, E., Perez, G., Ascariz, P.: PVS linear algebra libraries for verification of control software algorithms in C/ACSL. In: Goodloe, A.E., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 147–161. Springer, Heidelberg (2012)
10. Hirvensalo, M.: Quantum Computing. Natural Computing Series. Springer (2004)
11. Stein. J.: `http://coq.inria.fr/pylons/contribs/view/LinAlg/trunk`
12. Khan Afshar, S., Aravantinos, V.: `http://hvg.ece.concordia.ca/code/hol-light/complex-vectors`
13. Leonhardt, U.: Quantum Physics of Simple Optical Instruments. Reports on Progress in Physics 66(7), 1207 (2003)
14. Leonhardt, U.: Essential Quantum Optics: From Quantum Measurements to Black Holes. Cambridge University Press (2010)
15. Mahmoud, M.Y., Aravantinos, V.: `http://hvg.ece.concordia.ca/code/hol-light/qoptics/qalgebra.ml`
16. The Coq development team: The Coq Proof Assistant Reference Manual. LogiCal Project, Version 8.0 (2004)
17. Ralph, T.C., Gilchrist, A., Milburn, G.J., Munro, W.J., Glancy, S.: Quantum Computation with Optical Coherent States. Physical Review A 68, 042319 (2003)
18. Strang, G.: Introduction to Linear Algebra. Wellsley-Cambrige Press (2003)
19. Vinga, S., Almeida, J.: Alignment-Free Sequence Comparison – A Review. Bioinformatics 19(4), 513–523 (2003)