Functional framework for representing and transforming quantum channels

Jarosław Adam Miszczak
Institute of Theoretical and Applied Informatics,
Polish Academy of Sciences
Bałtycka 5, 44-100 Gliwice, Poland

miszczak@iitis.pl

Abstract

We develop a framework which aims to simplify the analysis of quantum states and quantum operations by harnessing the potential of function programming paradigm. We show that the introduced framework allows a seamless manipulation of quantum channels, in particular to convert between different representations of quantum channels, and thus that the use of functional programming concepts facilitates the manipulation of abstract objects used in the language of quantum theory.

For the purpose of our presentation we will use *Mathematica* computer algebra system. This choice is motivated twofold. First, it offers a rich programming language based on the functional paradigm. Second, this programming language is combined with powerful symbolic and numeric manipulation capabilities.

Keywords
quantum channels, functional programming, scientific computing

1 Introduction

Functional programming is frequently seen as an attractive alternative to the traditional methods used in scientific computing, which are based mainly on the imperative programming paradigm [3]. Among the features of functional languages which make them suitable for the use in this area is the easiness of execution of the functional code in the parallel environments.

The main aim of this work is to show that the functional programming concepts facilitate the use of abstract objects used in the language of quantum theory. We develop a framework which aims to simplify the analysis of quantum states and quantum operations by harnessing the potential of functional programming paradigm. For the purpose of our presentation we will use *Mathematica* computer algebra system. This choice is motivated twofold. First, it offers a rich programming language based on the functional paradigm. Second, this programming language is combined with powerful symbolic and numeric manipulation capabilities.

During the last few years a number of simulators of quantum information processing has been developed using *Mathematica* computing system [8, 5, 7, 2]. Unfortunately, these packages do not use functional programming capabilities of this system and are focused on pure states and unitary operations. Moreover, they focus on the quantum mechanical systems which can be represented using state vectors and include only a basic functionality required for the purpose of manipulating and analyzing quantum states.

In this paper we follow the pragmatic approach and we provide a set of useful constructions which can be helpful for the analysis of quantum channels. At the same time we advocate the use of functional programming in this approach. We argue that by using the functional language elements provided by *Mathematica* one can easily and efficiently convert between different representations of quantum channels.

2 Functional syntax for quantum channels

2.1 Notation

In the following we assume that the quantum systems are represented by finite-dimensional density matrices, *i.e.* positive semidefinite complex matrices with unit trace. The space of density matrices of dimension d is denoted by Ω_d. We use **res**, **unres** operations [6] for converting between matrix and vector forms of states and operators. In the *Mathematica* language function **res** is defined as a synonym for a built-in function **Flatten**

```
Res = Function[m, Flatten[m]];
```

This function transforms a matrix `m` into a vector in a row order. Function **unres**, which is a reverse transformation, is defined in *Mathematica* as

```
Unres = Function[m, Partition[m, Sqrt[Length[m]]]];
```

and it uses built-in function **Partition** to get back from a one-dimensional list to a matrix. As the space of density matrices is unitary with Hilbert-Schmidt scalar product, we introduce a function

```
HSInner = Function[x, Function[y, Tr[x.ConjugateTranspose[y]]]];
```

which, thanks to the curried form, allows using a partial application in the application of this scalar product.

Unfortunately *Mathematica* does not provide a straightforward support for the partial application of functions. The language does not allow using functions with too few parameters and one has to explicitly use empty slots (`#`-signs) to define a partially applied function. For this reason in order to use the functional version of some procedures, it is necessary to provide a curried version of these functions.

2.2 Simple channels

Let us illustrate the above considerations with the simplest example – the transposition map. This map is defined as

$$\rho \mapsto \rho^T, \tag{1}$$

and can be expressed in *Mathematica* as

```
trans = Function[x, Transpose[x]]
```

or using more compact syntax as `trans = Transpose[#]&`.

One should note that this map is not completely positive, hence it does not represent a valid quantum channel. Nevertheless, it is useful for presenting basic transformations which can be performed on quantum channels.

If we would like to apply this function on some state ρ we simply write `trans[`ρ`]`. In many situations however, one needs to apply a map on a list `rhos` of states or matrices. In this case we simply map the functions representing the map on the list using **Map** function as

```
Map[trans, rhos]
```

or using more compact syntax as `trans /@ rhos`.

2.3 Channels with parameters

In order to use channels defined by parametrized expression, one can employ partially applied functions. The simplest example of such channels is a depolarizing channel $\Psi_{D(p,d)}$ defined as

$$\Psi_{D(p,d)}(\rho) = (1-p)\rho + p\frac{1}{n}\mathbb{1}_n, \tag{2}$$

where we assume that $\rho \in \mathbb{M}_n$ and $\mathbb{1}_n$ denotes the identity matrix of the appropriate size.

Using the notation introduced in Section **??**, this channel can be represented by a function

```
dep = Function[d, Function[p, Function[x,
    (1-p)x + p IdentityMatrix[d]/d
]]];
```

Here we follow the convention that the function parameters should be organized in such a way, that by providing all but one of them, we obtain a function accepting quantum state as an argument. In the above case the first two parameters represent the dimension and the reliability of the channel (the probability of introducing no errors).

Function `dep` requires three arguments and its application on state ρ is achieved by first declaring the instance of the channel for a fixed dimension (*e.g* d=4)

```
dep4 = dep[4];
```

and next using this function with a specific probability `p`

```
dep4[p][ρ];
```

However, one can use `dep` function to define the expression in which only two arguments are provided

```
g = (dep[#1][p][#2]) &
```

and this allows obtaining a general definition of the depolarizing channel with a fixed parameter `p`, identical to the following definition

```
Function[d, Function[x, (1-p) x + p IdentityMatrix[d]/d]];
```

Function `g` accepts two arguments representing the dimension and the input state. Its application on some state $\rho \in \mathbb{M}_4$ reads

```
g[4][ρ];
```

and this syntax allows the selection of an argument which should be fixed during the manipulation.

3 Representations of quantum channels

3.1 Natural representation

As channels are linear mappings, it is possible, at least in finite-dimensional case, to represent them by matrices. Let us assume that we are dealing with $d = n \times n$ dimensional matrices.

The base in n^2-dimensional space $\mathbb{M}_n$ is given by matrices, which can be obtained by using `Unres` operations on the base vectors in the d-dimensional space $\mathbb{C}^d, d = n^2$, as

```
base = Map[Unres[UnitVector[d, #]] &, Range[d]];
```

where `Range[d]` returns a list containing numbers $1, 2, \ldots, d$. In the following we assume that the d-dimensional matrix base can be obtained using function `BaseMatrices[d]` defined as

```
BaseMatrices =  Function[d, Map[Unres[UnitVector[d, #]] &, Range[d]]];
```

If the list `fBase` contains the images of the quantum channel `f` on the base

```
fBase = f /@ base
```

then the natural representation can be calculated by unreshaping the images of the map on the base matrices in $\mathbb{M}_{d^2}$,

```
{Res /@ fBase}
```

Combining this into one function gives

```
NaturalRepresentation = Function[f, Function[d,
    With[{base=BaseMatrices[d^2]}, Map[Res[f[#]]&, base]]
];
```

We denote the natural representation of the channel Φ by $\mathcal{M}_\Phi$, assuming that this matrix is obtained in the standard basis. Matrix $\mathcal{M}_\Phi$ is sometimes called a supermatrix for the channel Φ.

The above considerations can be summarized as the following definition.

Definition 1 (Natural representation) *For a given channel Ψ, the natural representation of Φ by $\mathcal{M}_\Phi$ is defined as*

$$(\mathcal{M}_\Phi)_{i\cdot} = \mathbf{res}\,\Phi(b_i) \tag{3}$$

where $(A)_{i\cdot}$ denotes i-th column of the matrix A and b_i, $i = 1, n^2$ denotes base matrices in $\mathbb{M}_n$.

For example, in order to obtain the matrix representation of the depolarizing channel `dep` acting on one qubit, one should use `NaturalRepresentation` function as

```
NaturalRepresentation[dep[2][p]][2]
```

In a similar manner one can check that the natural representation of the one-qubit transposition channel `trans`

```
NaturalRepresentation[trans][2]
```

is equal to the SWAP gate.

3.2 General natural representation

Clearly one can represent a given channel in a matrix form using not only a canonical base, but any orthonormal basis in $\mathbb{C}^{n^2}$. In this situation one cannot use the method described above as it relies on the special form of the canonical base matrices.

The straightforward method of calculating a matrix representation, is based on the formula

$$(M_\Phi^b)_{ij} = \mathbf{tr}[b_i \Phi(b_j)^\dagger], \tag{4}$$

where $b_i, i = 1, \ldots, n^2$ denotes the base.

Definition 2 (General natural representation) *For a given channel Ψ, the general natural representation of Φ in base b is defined as*

$$(\mathcal{M}_\Phi^b)_{ij} = \mathbf{tr}[\Phi(b_i) b_j^\dagger], \tag{5}$$

where b_i, $i = 1, n^2$ denote base matrices in $\mathbb{M}_n$.

This definition can be implemented using **`Outer`** function as

```
Function[f, Function[b,
    Outer[HSInner[#1][#2]&, Map[f,b], b, 1]
]];
```

where `base` is a given base or, alternatively, by using **`Map`** function as

```
Function[f, Function[b,
    Map[Map[#, b] &, Map[HSInner, Map[f, b]]]
]];
```

This method requires n^4 multiplications of $n \times n$ matrices and is highly inefficient.

The simplest method is to reconstruct a change of basis matrix M_B,

```
M_B = Map[Res, b]
```

and use it to obtain M_Φ^b as

$$M_\Phi^b = M_B M_\Phi M_B^\dagger. \tag{6}$$

3.3 Choi-Jamiołkowski representation

Complete positivity, one of the requirements for the map between finite-dimensional spaces can be formulated using Choi-Jamiolkowski representation of a map [4, 1]. This representation in the context of quantum channels is known as Jamiolkowski isomorphism and here the image of this isomorphism is denoted as $\mathcal{J}_\Phi$.

The Choi-Jamiolkowski representation is closely related to the natural representation. The natural representation of the channel acting on $n \times n$-dimensional matrices is always obtained with respect to some bases $\{b_i\}_{i=1,n^2}$, where n^2 is the dimension of the state space.

If one uses base b_i to obtain the natural representation of the channel Φ resulting in matrix M_Φ^b, then the Choi-Jamiolkowski matrix for this channel is obtained as

$$\{\mathcal{J}_\Phi^b\}_{i,j} = \mathbf{tr}[M_\Phi^b (b_i \otimes b_j)], \tag{7}$$

for $i, j = 1, n^2$.

Definition 3 (Choi-Jamiolkowski matrix) *Let $\{b_i\}$ be a base in $\mathbb{C}^{n^2}$. The Choi-Jamiolkowski matrix corresponding to a general natural representation in base $\{b_i\}$ is defined as*

$$\{\mathcal{J}_\Phi^b\}_{i,j} = \mathbf{tr}[M_\Phi^b(b_i \otimes b_j)]. \tag{8}$$

The Choi-Jamiolkowski representation of the channel Φ can be also obtained using several other methods. One of the simplest formulas is the one expressing $\mathcal{J}_\Phi$ as a sum

$$\mathcal{J}_\Phi = \sum_{i=1}^{d} \Phi(e_i) \otimes e_i. \tag{9}$$

Assuming that `base` represents matrix base in d-dimensional space, this representation can be used by mapping

```
cjBase = Map[KroneckerProduct[f[#], #] &, base]
```

and accumulating the results

```
Plus /@ cjBase
```

Combining the above into one function gives

```
ChoiJamiolkowskiRepresentation = Function[f, Function[d,
    With[{base=BaseMatrices[d]},
        Map[Plus,[Map[KroneckerProduct[f[#],#]&, base]]]
]];
```

The Choi-Jamiolkowski representation of a channel is related to the natural representation, one can easily construct a Choi-Jamiolkowski matrix corresponding to a given generalized natural representation.

Acknowledgements This work was supported by the Polish Ministry of Science and Higher Education under the grant number IP2011 036371 and by the Polish National Science Centre under the grant number DEC-2011/03/D/ST6/00413. Author would like to acknowledge stimulating discussions with Z. Puchała, P. Gawron, D. Kurzyk, V. Jagadish and P. Zawadzki.

References

[1] M.-D. Choi. Completely positive linear maps on complex matrices. *Linear Algebr. Appl.*, 10(3):285–290, 1975.

[2] J. L. Gómez-Muñoz and F. Delgado-Cepeda. Quantum 2.3 for Mathematica 8, 2011-. Software available on-line at `http://homepage.cem.itesm.mx/lgomez/`.

[3] K. Hinsen. The promises of functional programming. *Comput. Sci. Eng.*, 11(4):86–90, 2009.

[4] A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Rep. Math. Phys.*, 3(4):275–278, 1972.

[5] B. Juliá-Díaz, J.M. Burdis, and F. Tabakin. QDENSITY—a Mathematica quantum computer simulation. *Comput. Phys. Commun.*, 174(11):914934, 2006.

[6] J.A. Miszczak. Singular value decomposition and matrix reorderings in quantum information theory. *Int. J. Mod. Phys. C*, 22(9):897–918, 2011.

[7] F. Tabakin and B. Juliá-Díaz. QCWAVE – a Mathematica quantum computer simulation update. *Comput. Phys. Commun.*, 182(8):16931707, 2011.

[8] H. Touchette and P. Dumais. The quantum computation package for Mathematica 4.0. Software available on-line at `http://crypto.cs.mcgill.ca/QuCalc/`, 2000-.