Implementation of quantum maps by programmable quantum processors

Mark Hillery,[1] Mário Ziman,[2] and Vladimír Bužek[2,3]
[1]*Department of Physics and Astronomy, Hunter College of CUNY, 695 Park Avenue, New York, New York 10021*
[2]*Research Center for Quantum Information, Slovak Academy of Sciences, Dúbravská cesta 9, 842 28 Bratislava, Slovakia*
[3]*Department of Mathematical Physics, National University of Ireland, Maynooth, Ireland*

(Received 7 May 2002; published 4 October 2002)

A quantum processor is a device with a data register and a program register. The input to the program register determines the operation, which is a completely positive linear map, that will be performed on the state in the data register. We develop a mathematical description for these devices, and apply it to several different examples of processors. The problem of finding a processor that will be able to implement a given set of mappings is also examined, and it is shown that, while it is possible to design a finite processor to realize the phase-damping channel, it is not possible to do so for the amplitude-damping channel.

DOI: 10.1103/PhysRevA.66.042302 PACS number(s): 03.67.Lx

I. INTRODUCTION

The coherent control of individual quantum systems is one of the most exciting achievements in physics in the last decade [1]. The possibility of controlling quantum dynamics has far reaching consequences for quantum technologies, in particular, for quantum computing [2]. One of the best known applications of coherent control in quantum physics is the state preparation of an *individual* quantum system. For example, a particular state of the vibrational motion of a trapped ion can be prepared by using a well-defined sequence of external laser pulses. Another possibility is to focus on controlling the dynamics, that is, the unitary evolution operator. One way of doing this is to realize a particular evolution operator by means of a sequence of "elementary" interactions, which are sequentially turned on and off (for more details see Refs. [3,4], and for a specific application to trapped ions see Ref. [5] and references therein).

In the theory of quantum coherent control it is assumed that the control of the dynamics is realized via external classical parameters, such as the intensity of a laser pulse or the duration of an interaction. In this case, the information that controls the quantum system is classical, and it is set by an experimentalist to achieve a single, fixed outcome. This is analogous to programming a computer to perform a single task by setting dials or switches to particular positions, each task requiring different positions.

In this paper we will study a different type of quantum control. We will assume that the information about the quantum dynamics of the system under consideration is not represented by classical external parameters, but rather is encoded in the state of another quantum system. A typical example of such an arrangement is a controlled-NOT (c-NOT) operation (or in general a controlled-U operation). In this case, the specific operation performed on the system, the target, depends on the state of a second quantum system, the control. If the control qubit is in the state $|0\rangle$ the target qubit is left unchanged, but if it is in the state $|1\rangle$, then a NOT operation is applied to the target qubit. This means that this device can perform at least two operations on the target qubit, the identity and NOT. There are, however, further possibilities. Let us suppose that the control qubit is initially in a superposition of $|0\rangle$ and $|1\rangle$, and that we are only interested in the target qubit at the output of the device, so that we trace out the control qubit to obtain the reduced density matrix of the target qubit. The action of the c-NOT gate on the target qubit can then be described as a completely positive, linear map acting on the initial density matrix of the target qubit, with the actual map being determined by the state of the control qubit. We take this device to be a model for a programmable quantum gate array, or quantum processor. Generally speaking, a programmable quantum processor is a device that implements a completely positive linear map, which is determined by the state of one quantum system, on a second quantum system. These processors have two registers, the data register and the program register. The data register contains the quantum system on which the map is going to be applied, and the program register contains the quantum system whose state determines the map. The third element of this device is a fixed array of quantum gates that act on both the program and the data state. The virtue of this arrangement is that we do not have to build a different processor every time we want to realize a new map, we simply change the program state. This allows us greater flexibility than a device in which the map is determined by setting external parameters. For example, it could be the case that we do not even know what the program state is. This would occur when the state of the program register is the output of another quantum device. We will refer to the selection of the program state to perform a desired operation as *quantum programming*.

Programmable quantum processors (gate arrays) were first considered by Nielsen and Chuang [6]. They were interested only in the case in which a unitary operation, rather than a more general completely positive linear map, is performed on the state in the data register. If $|\psi\rangle_d$ is the state of the data register, $|\Xi_U\rangle_p$ a program state that implements the operator U on the data state, and G the overall unitary operation implemented by the fixed gate array, then their processor carries out the transformation

$$G(|\psi\rangle_d\otimes|\Xi_U\rangle_p)=U|\psi\rangle_d\otimes|\Xi'_{U,\psi}\rangle_p, \tag{1.1}$$

where $|\Xi'_{U,\psi}\rangle_p$ is the state of the program register after the transformation G has been carried out. The subscripts U and ψ indicate that this state can depend on both the operation U and the state $|\psi\rangle_d$ of the data register d. They were able to prove a number of results about this device. First, they showed that the output of the program register does not depend on the data register, a fact that follows from the unitarity of G. Second, they proved that the number of possible programs is equal to the dimension of the program register.

Let us assume the first of these results and show how to prove the second. Consider two program states $|\Xi_U\rangle_p$ and $|\Xi_V\rangle_p$ that cause the operators U and V, respectively, to act on the data register. This implies that

$$G(|\psi\rangle_d\otimes|\Xi_U\rangle_p)=U|\psi\rangle_d\otimes|\Xi'_U\rangle_p,$$

$$G(|\psi\rangle_d\otimes|\Xi_V\rangle_p)=V|\psi\rangle_d\otimes|\Xi'_V\rangle_p. \tag{1.2}$$

The unitarity of G implies that

$${}_p\langle\Xi_V|\Xi_U\rangle_p={}_d\langle\psi|V^{-1}U|\psi\rangle_d\,{}_p\langle\Xi'_V|\Xi'_U\rangle_p, \tag{1.3}$$

and if ${}_p\langle\Xi'_V|\Xi'_U\rangle_p\neq 0$, then

$${}_d\langle\psi|V^{-1}U|\psi\rangle_d=\frac{{}_p\langle\Xi_V|\Xi_U\rangle_p}{{}_p\langle\Xi'_V|\Xi'_U\rangle_p}. \tag{1.4}$$

The left-hand side of this equation depends on $|\psi\rangle_d$ while the right does not. The only way this can be true is if

$$V^{-1}U=e^{i\phi}\mathbb{1}, \tag{1.5}$$

for some real ϕ. This means that the operators U and V are the same up to a phase. If we want these operators to be different, we must have that ${}_p\langle\Xi'_V|\Xi'_U\rangle_p=0$, which by Eq. (1.3) implies that ${}_p\langle\Xi_V|\Xi_U\rangle_p=0$. Therefore, the program states corresponding to different unitary operators must be orthogonal. This implies that the dimension of the program register must be greater than or equal to the number of different unitary operators that can be performed on the data register.

In this paper we would like to consider the more general problem of quantum processors that realize completely positive linear maps, not just unitary operators. The paper is organized as follows. In Sec. II we derive a formalism for describing and analyzing quantum processors, and apply it to both pure and mixed program states. In Sec. III we present several classes of quantum processors, while Sec. IV will be devoted to the problem of processor "design," that is, we will discuss specific processors that are able to implement various classes of quantum processes. In the concluding section we will briefly discuss probabilistic quantum processors, which are based on dynamics conditioned on the results of measurements.

II. QUANTUM PROCESSORS

A general quantum processor consists of two registers, a data register and a program register, and a fixed array of quantum gates. The input state that goes into the program register encodes an operation we want to perform on the data register. We would first like to show that the action of the processor can be fully described by a specific set of linear operators.

A. Pure program states

Let $|\psi\rangle_d$ be the input state of the data register, $|\Xi\rangle_p$ be the input program state, and G be the unitary operator that describes the action of the array of quantum gates. If $\{|j\rangle_p|j=1,\ldots,N\}$ is a basis for the space of program states, then we have that

$$G(|\psi\rangle_d\otimes|\Xi\rangle_p)=\sum_{j=1}^{N}|j\rangle_p\,{}_p\langle j|G(|\psi\rangle_d\otimes|\Xi\rangle_p). \tag{2.1}$$

If we define the operator $A_j(\Xi)$, which acts on the data register, by

$$A_j(\Xi)|\psi\rangle_d={}_p\langle j|G(|\psi\rangle_d\otimes|\Xi\rangle_p), \tag{2.2}$$

then we have that

$$G(|\psi\rangle_d\otimes|\Xi\rangle_p)=\sum_{j=1}^{N}A_j(\Xi)|\psi\rangle_d\otimes|j\rangle_p. \tag{2.3}$$

This means that the output density matrix of the data register is given by

$$\rho_d^{out}=\sum_{j=1}^{N}A_j(\Xi)|\psi\rangle_d\,{}_d\langle\psi|A_j^\dagger(\Xi). \tag{2.4}$$

The operator $A_j(\Xi)$ depends on the program state, but it can be expressed in terms of operators that do not. Define the operators

$$A_{jk}=A_j(|k\rangle)={}_p\langle j|G|k\rangle_p, \tag{2.5}$$

where $|k\rangle$ is one of the basis states we have chosen for the space of program states. We have that for any program state $|\Xi\rangle$,

$$A_j(\Xi)=\sum_{k=1}^{N}{}_p\langle k|\Xi\rangle_p A_{jk}. \tag{2.6}$$

This means that the operators A_{jk} completely characterize the processor in the case of pure program states. We shall call these operators the basis operators for the processor. These operators have the following property:

$$\sum_{j=1}^{N}A_{jk_1}^\dagger A_{jk_2}=\sum_{j=1}^{N}\langle k_1|G^\dagger|j\rangle\langle j|G|k_2\rangle=\mathbb{1}_d\delta_{k_1k_2}, \tag{2.7}$$

where we have used the decomposition $\Sigma_j|j\rangle\langle j|=\mathbb{1}_p$.

An obvious question to ask at this point is whether any set of operators satisfying Eq. (2.7) corresponds to a quantum processor. The following construction allows us to show that this is the case [7]. Given a set of N^2 operators acting on $\mathcal{H}_d$, we can construct an operator G acting on the product space $\mathcal{H}_d \otimes \mathcal{H}_p$, where $\mathcal{H}_p$ is an N-dimensional space with basis $\{|k\rangle_p | k=1,\dots,N\}$. We set

$$G = \sum_{j,k=1}^{N} A_{jk} \otimes |j\rangle_{p\,p}\langle k|. \tag{2.8}$$

It is now necessary to verify that G constructed in this way is unitary. Noting that

$$G^\dagger = \sum_{j,k=1}^{N} A_{jk}^\dagger \otimes |k\rangle_{p\,p}\langle j|, \tag{2.9}$$

we see that Eq. (2.7) implies that $G^\dagger G = \mathbb{1}$, so that G preserves the length of vectors and is unitary.

It is possible to express the basis operators for closely related processors in terms of each other. For example, if $\{B_{jk} | j,k=1,\dots,N\}$ are the basis operators for $G^\dagger$, then from Eq. (2.9) we see that $B_{jk} = A_{kj}^\dagger$. If G_1 and G_2 are two processors (unitary operators) with basis operators $\{A_{jk}^{(1)} | j,k=1,\dots,N\}$ and $\{A_{jk}^{(2)} | j,k=1,\dots,N\}$, respectively, then the basis operators C_{jk} for the processor corresponding to the operator $G_1 G_2$ are

$$C_{jk} = \sum_{n=1}^{N} A_{jn}^{(1)} A_{nk}^{(2)}. \tag{2.10}$$

This follows immediately if both G_1 and G_2 are expressed in the form given in Eq. (2.8) and then multiplied together. If we apply this equation to the case $G_1 = G$ and $G_2 = G^\dagger$, and note that $GG^\dagger = \mathbb{1}$, we have that

$$\sum_{j=1}^{N} A_{k_1 j} A_{k_2 j}^\dagger = \mathbb{1}_d \delta_{k_1 k_2}. \tag{2.11}$$

It is clearly possible to generalize Eq. (2.10) to the case when there is a product of more than two operators.

B. General program states

Suppose the program is represented by a mixed state $\varrho_p = \Sigma_{kl} R_{kl} |k\rangle\langle l|$. Then for the induced mapping we have

$$\varrho_d^{out} = \sum_{klmn} R_{kl} A_{mk} \varrho_d^{in} A_{nl}^\dagger \mathrm{Tr}_p(|m\rangle_p\langle n|) = \sum_{klm} R_{kl} A_{mk} \varrho_d A_{ml}^\dagger. \tag{2.12}$$

We shall denote by $\mathcal{C}_G$ the set of completely positive linear maps realizable by using the fixed processor G and any mixed state in the program space as a program.

Let us now address the question of whether it is possible to find a second processor G' that can realize any map in the set $\mathcal{C}_G$ using only pure state programs. Any mixed state in $\mathcal{H}_p$ can be purified, but the purification is not unique [2,8]. We begin by defining a new program space $\mathcal{H}_{p'} = \mathcal{H}_p \otimes \mathcal{H}_p$ and choosing the purification in the following way:

$$\varrho_p = \sum_k \lambda_k |\chi_k\rangle\langle\chi_k|$$
$$\to |\Phi\rangle_{p'} = \sum_k \sqrt{\lambda_k} |\chi_k\rangle_p \otimes |k\rangle, \tag{2.13}$$

where ϱ_p has been written in terms of its spectral decomposition. We define the unitary operator corresponding to the new processor, which acts on the space $\mathcal{H}_d \otimes \mathcal{H}_{p'}$, by

$$G' \coloneqq G \otimes \mathbb{1}. \tag{2.14}$$

The conjecture is that the processor G' with the pure program state $|\Phi\rangle_{p'}$ will produce the same mapping as the processor G with the mixed program state ϱ_p. If this is true, then we will have shown that by using only pure program states with the processor G' we can implement the entire class of superoperators $\mathcal{C}_G$.

In order to prove this we have to show that

$$\mathrm{Tr}_p G \varrho_d \otimes \varrho_p G^\dagger = \mathrm{Tr}_{p'} G' \varrho_d \otimes \varrho_{p'} G'^\dagger \tag{2.15}$$

for all ϱ_d. The right-hand side of this equation can be rewritten as

$$\begin{aligned}
&\mathrm{Tr}_{p'} G' \varrho_d \otimes \varrho_{p'} G'^\dagger \\
&= \mathrm{Tr}_{p'}\left[\sum_{kl} \sqrt{\lambda_k \lambda_l} (G \varrho_d \otimes |\chi_k\rangle\langle\chi_l| G^\dagger) \otimes |k\rangle\langle l|\right] \\
&= \sum_{kl} \sqrt{\lambda_k \lambda_l} \mathrm{Tr}_p[(G \varrho_d \otimes |\chi_k\rangle\langle\chi_l| G^\dagger) \delta_{kl}] \\
&= \mathrm{Tr}_p\left[G \varrho_d \otimes \left(\sum_k \lambda_k |\chi_k\rangle\langle\chi_k|\right) G^\dagger\right] \\
&= \mathrm{Tr}_p G \varrho_d \otimes \varrho_p G^\dagger,
\end{aligned} \tag{2.16}$$

which proves Eq. (2.15). Therefore, we can conclude that it is possible to "mimic" mixed program states for a given processor by introducing a larger program space $\mathcal{H}_{p'}$ and a new processor mapping $G' = G \otimes \mathbb{1}$.

C. Correspondence between programs and mappings

We have just seen that two different programs on two different processors can lead to the same mapping, and now we would like to examime whether different programs on the same processor can produce identical mappings. We shall show that this can occur by means of a simple example. Let Q_p be a projection operator on the program space whose range has dimension D, where $1 < D < N$, and let U_1 and U_2 be two different unitary operators on the data space. Consider the processor given by

$$G = U_1 \otimes Q_p + U_2 \otimes (\mathbb{1}_p - Q_p). \tag{2.17}$$

Any program state in the range of Q_p produces the mapping U_1 on the data state, and there are clearly an infinite number of these. Therefore, we can conclude that there are processors for which many program states produce the same operation on the data state.

We shall now show that the opposite can also occur, i.e., that there exists a processor for which every program state (mixed or pure) encodes a different superoperator We will present an example which illustrates that. To do so, we utilize results of Ref. [9] where the unitary transformation

$$G = \cos\phi \mathbb{1} + i \sin\phi S \tag{2.18}$$

was introduced. The swap operator $S = \Sigma_{kl}|kl\rangle\langle lk|$ is defined in any dimension. The so-called *partial swap* transformation G acts on two qudits (d-dimensional systems). Let us restrict our attention to qubits, and identify one of the qubits with the data register and the other with the the program system. In Ref. [9] it was shown that if the program system is prepared in the state $\varrho_p \equiv \xi$, then the induced map (superoperator) T_ξ is contractive with its fixed point equal to ξ [10]. Since each contractive superoperator has only a single fixed point, we can conclude that different program states $\xi \neq \xi'$ induce different superoperators, i.e., $T_\xi \neq T_{\xi'}$. As a result we can conclude that in the processor given by Eq. (2.18), for any value of the parameter ϕ, the correspondence between programs and mappings is one to one. Finally, we note that while here we considered only qubits the results in this section also hold for qudits [10].

D. Equivalent processors

We shall regard two processors G_1 and G_2 as essentially equivalent if one can be converted into the other by inserting *fixed* unitary gate arrays at the input and output of the program register, that is, if

$$G_2 = (\mathbb{1}_d \otimes U_{p1}) G_1 (\mathbb{1}_d \otimes U_{p2}), \tag{2.19}$$

where U_{p1} and U_{p2} are unitary transformations on the program space. If this equation is satisfied, then the processors defined by the two gate arrays will perform the same set of operations on data states, but the program states required to perform a given operation are different, and the outputs of the program registers will be different as well. If Eq. (2.19) holds, then for the basis operators $A_{jk}^{(i)}$ ($i = 1,2$) associated with the two processors we have

$$A_{jk}^{(2)} = \sum_{m,n=1}^{N} (U_{p1})_{jm} (U_{p2})_{nk} A_{mn}^{(1)}. \tag{2.20}$$

Therefore, we can regard two processors whose set of operators $A_{jk}^{(i)}$ are related by the above equation as equivalent.

If processors 1 and 2 are equivalent, then they will implement the same set of superoperators, i.e., $\mathcal{C}_{G_1} = \mathcal{C}_{G_2}$. In order to see this, suppose that, when the state $|\Xi_1\rangle_p$ is sent into the program register of processor 1, the map T_{Ξ_1}, with program operators $A_j^{(1)}(\Xi_1)$, is performed on the data state. Now consider what happens when we send the state $|\Xi_2\rangle_p = U_{p2}^{-1}|\Xi_1\rangle_p$ into the input of the program register of processor 2. This will produce the mapping T_{Ξ_2} on the data state of processor 2. The relation between the program operators $A_j^{(1)}(\Xi_1)$ and $A_j^{(2)}(\Xi_2)$ is

$$A_j^{(2)}(\Xi_2) = \sum_{k=1}^{N} (U_{p1})_{jk} A_1^{(1)}(\Xi_1). \tag{2.21}$$

The operators $A_j^{(1)}(\Xi_1)$ are Kraus operators for the mapping T_{Ξ_1} and the operators $A_j^{(2)}(\Xi_2)$ are Kraus operators for the mapping T_{Ξ_2}. The above equation implies that the mappings are identical, $T_{\Xi_1} = T_{\Xi_2}$ [7]. Therefore, any superoperator that can be realized by processor 1 can be realized by processor 2. Similarly, it can be shown that any superoperator that can be realized by processor 2 can also be realized by processor 1. This shows that the two processors implement the same set of superoperators.

A special case of this type of equivalence occurs when the two processors are simply related by a change of the basis in the program space, i.e., when $U_{p1} = U_{p2}^{-1}$. It is possible to derive conditions that the basis operators of the two processors must satisfy if the processors are to be equivalent in this more restricted sense. These follow from the fact that the trace is independent of the basis in which it is taken. If $U_{p1} = U_{p2}^{-1}$, then $\mathrm{Tr}_p(G_1) = \mathrm{Tr}_p(G_2)$, which implies that

$$\sum_{j=1}^{N} A_{jj}^{(1)} = \sum_{j=1}^{N} A_{jj}^{(2)}. \tag{2.22}$$

We also have that $\mathrm{Tr}_p(G_1^n) = \mathrm{Tr}_p(G_2^n)$, which for the case $n = 2$ gives us

$$\sum_{j,k=1}^{N} A_{jk}^{(1)} A_{kj}^{(1)} = \sum_{j=1}^{N} A_{jk}^{(2)} A_{kj}^{(2)}. \tag{2.23}$$

Clearly, by taking higher values of n, we can derive additional equivalence conditions.

We end this section by summarizing some of our results so far.

(i) For a given processor G any member of the class of all possible completely positive linear maps realizable by G, $\mathcal{C}_G$, can be expressed in terms of the operators A_{jk}.

(ii) We can mimic the action induced by any mixed program state by a pure program state in a larger program space.

(iii) For any two mappings realized by the processor G and the pure state programs $|\Xi_1\rangle_p$ and $|\Xi_2\rangle_p$ the identity

$$\sum_k A_k^\dagger(\Xi_1) A_k(\Xi_2) = \langle \Xi_1|\Xi_2\rangle \mathbb{1}_d \tag{2.24}$$

holds. This follows directly from Eqs. (2.6) and (2.7).

III. CLASSES OF PROCESSORS

In this section we will examine several different kinds of quantum processors. These will serve to illustrate some of the general considerations in the previous sections.

A. *U* processors

Let us suppose that the eigenvectors of the unitary operator G that describes the fixed array of gates are tensor products. In particular, suppose that we have a single orthonormal basis for $\mathcal{H}_p$, $\{|k\rangle_p|k=1,\ldots,N\}$ and a collection of orthonormal bases for $\mathcal{H}_d$, $\{|\phi_{mk}\rangle_d|k=1,\ldots,N,m=1,\ldots,M\}$, where M is the dimension of $\mathcal{H}_d$. For each value of k, the vectors $\{|\phi_{mk}\rangle_d|m=1,\ldots,M\}$ form an orthonormal basis for $\mathcal{H}_d$. We call a processor a *U processor* if the eigenvectors of G, $|\Phi_{mk}\rangle_{dp}$, are of the form

$$|\Phi_{mk}\rangle_{dp}=|\phi_{mk}\rangle_d\otimes|k\rangle_p. \tag{3.1}$$

In this case the operators A_{jk} are given by $A_{jk}=\delta_{jk}U_j$ where U_j is unitary (its eigenstates are just $\{|\phi_{mj}\rangle_d|m=1,\ldots,M\}$). This is the type of processor that was studied by Chuang and Nielsen [6], and we recall that the dimension of $\mathcal{H}_p$ is equal to the number of unitary operators that this type of processor can perform. The processor acts on the state $|\psi\rangle_d\otimes|j\rangle_p$ as

$$G(|\psi\rangle_d\otimes|j\rangle_p)=(U_j|\psi\rangle_d)\otimes|j\rangle_p, \tag{3.2}$$

where $|\psi\rangle_d$ is an arbitrary data state.

For a general pure program state $|\Xi\rangle_p=\sum_j\alpha_j|j\rangle_p$ the encoded mapping, or superoperator, T_Ξ, is given by the expression $T_\Xi[\varrho_d]=\sum_j|\alpha_j|^2U_j\varrho_dU_j^\dagger$. In the case of a mixed program state $\varrho_p=\sum_{jk}R_{jk}|j\rangle\langle k|$ the data state is transformed as $T_{\varrho_p}[\varrho_d]=\sum_jR_{jj}U_j\varrho_dU_j^\dagger$. Comparing these two cases we conclude that we can always mimic a mixed program state by a pure one, in particular, it is enough to set $\alpha_j=\sqrt{R_{jj}}$. Hence, for this type of processor we can consider only pure program states without any loss of generality.

Finally, we note that for all program states $|\Xi\rangle_p$

$$T_\Xi\left[\frac{1}{d}\mathbb{1}_d\right]=\sum_j|\alpha_j|^2U_j\frac{1}{d}\mathbb{1}_dU_j^\dagger=\frac{1}{d}\mathbb{1}_d. \tag{3.3}$$

This implies that each element of $\mathcal{C}_G$ is *unital*, i.e., it maps the identity operator into itself.

B. *Y* processors

A second possibility is to consider a situation that is in some ways the reverse of the one we just examined. We have a single orthonormal basis for $\mathcal{H}_d$, $\{|m\rangle_d|m=1,\ldots,M\}$, and a set of orthonormal bases for $\mathcal{H}_p$, $\{|\chi_{mk}\rangle_p|k=1,\ldots,N\}$, where $m=1,\ldots,M$ labels the bases and the index k labels the individual basis elements. We again assume that the eigenvectors of G, $|\Phi_{mk}\rangle_{dp}$ are tensor products, but now they are given by

$$|\Phi_{mk}\rangle_{dp}=|m\rangle_d\otimes|\chi_{mk}\rangle_p. \tag{3.4}$$

In this case the processor can be expressed as $G=\sum_m|m\rangle_d\langle m|\otimes U_m$, where U_m is unitary and has eigenvectors $\{|\chi_{mk}\rangle_p|k=1,\ldots,N\}$. We find the operators A_{jk} by first choosing a single orthonormal basis in $\mathcal{H}_p$, $\{|k\rangle_p\}$, and computing

$$\begin{aligned}A_{jk}&={}_p\langle j|G|k\rangle_p\\&=\sum_m|m\rangle\langle m|\langle j|U_m|k\rangle=\sum_m(U_m)_{jk}|m\rangle\langle m|.\end{aligned} \tag{3.5}$$

The maps produced by Y processors are unital, as can be seen from

$$\begin{aligned}\sum_jA_{jk_1}A_{jk_2}^\dagger&=\sum_j\sum_{mn}(U_m)_{jk_1}(U_n^\dagger)_{jk_2}|m\rangle\langle m|n\rangle\langle n|\\&=\sum_{jm}(U_m)_{jk_1}(U_m^\dagger)_{jk_2}|m\rangle\langle m|\\&=\delta_{k_1k_2}\sum_m|m\rangle\langle m|=\delta_{k_1k_2}\mathbb{1}.\end{aligned} \tag{3.6}$$

The action of a Y processor is particularly simple if all of the operators U_m have some common eigenstates, and the program state is one of them. Suppose that $U_m|\Xi\rangle_p=e^{i\phi_m}|\Xi\rangle_p$, then

$$G\left(\sum_mc_m|m\rangle_d\otimes|\Xi\rangle_p\right)=\left(\sum_mc_me^{i\phi_m}|m\rangle_d\right)\otimes|\Xi\rangle_p. \tag{3.7}$$

In summary, we can say that both the U and Y processors are controlled-U gates; in the U processor, the control system is the program and the target is the data, and in the Y processor, it is the target that is the program and the control that is the data.

C. *U′* processors

Let us consider a simple modification of the U processor, which we shall call the U' processor. Suppose we have two different orthonormal bases of $\mathcal{H}_p$, $\{|k\rangle_p\}$, and $\{|\chi_k\rangle_p\}$. We define a U' processor to have a unitary operator G of the form

$$G=\sum_kU_k\otimes|k\rangle_p\langle\chi_k|. \tag{3.8}$$

This looks like a new kind of processor, but it is actually equivalent to a U processor. This can be seen immediately if we realize that there exists a unitary operator U_p acting on $\mathcal{H}_p$ such that $|\chi_k\rangle_p=U_p|k\rangle_p$. Therefore, we have that

$$G=\left(\sum_k|k\rangle_p\langle k|\right)(\mathbb{1}_d\otimes U_p^\dagger), \tag{3.9}$$

so that G is, in fact, equivalent to a U processor.

D. *Y′* processors

Now let us try a modification of the Y processor in the same spirit as the one we just made to the U processor. Suppose we have two different orthonormal bases of $\mathcal{H}_d$,

$\{|m\rangle_p\}$, and $\{|\phi_m\rangle_d\}$. We define a Y' processor to have a unitary operator G of the form

$$G=\sum_m |m\rangle_d\langle\phi_m|\otimes U_m . \tag{3.10}$$

For the operators A_{jk} we obtain

$$A_{jk}={}_p\langle j|G|k\rangle_p=\sum_m |m\rangle\langle\phi_m|(U_m)_{jk} . \tag{3.11}$$

This type of processor is not equivalent to a Y processor. It does, however, share the property of producing unital maps as can be seen from

$$\begin{aligned}\sum_j A_{jk_1}A^\dagger_{jk_2}&=\sum_{j,m,n} |m\rangle\langle\phi_m|\phi_n\rangle\langle n|(U_m)_{jk_1}(U^\dagger_n)_{k_2j}\\ &=\sum_{j,m} |m\rangle\langle m|(U^\dagger_m)_{k_2j}(U_m)_{jk_1}\\ &=\delta_{k_1k_2}\sum_m |m\rangle\langle m|\\ &=\delta_{k_1k_2}\mathbb{1}_d ,\end{aligned} \tag{3.12}$$

which implies that for any program state the identity on $\mathcal{H}_d$ is mapped into itself.

E. Covariant processors

Another class of processors that may be of interest are *covariant* processors. Covariance has proven to be an important property in the study of quantum machines. Covariant processors have the property that if the processor maps the input data state $\varrho_{in}=|\psi\rangle_{d\,d}\langle\psi|$, which we shall assume is a qudit, onto the output density matrix ρ_{out}, then it maps the input state $U|\psi\rangle_d$ onto the output density matrix $U\rho_{out}U^{-1}$, for all $U\in\mathcal{G}$, where $\mathcal{G}$ is a subgroup of SU(D), for some subset $\mathcal{S}$ of all possible program states [11]. This relation implies that if $|\Xi\rangle\in\mathcal{S}$, then the operators $A_j(\Xi)$ satisfy the relation

$$\sum_{j=1}^N UA_j(\Xi)\varrho_{in}A_j^\dagger(\Xi)U^{-1}=\sum_{j=1}^N A_j(\Xi)U\varrho_{in}U^{-1}A_j^\dagger(\Xi), \tag{3.13}$$

for all $U\in\mathcal{G}$. Let us now consider the case $\mathcal{G}=$SU(D). If we take ρ_{in} to be $\mathbb{1}_d/d$, we find

$$\sum_{j=1}^N UA_j(\Xi)A_j^\dagger(\Xi)U^{-1}=\sum_{j=1}^N A_j(\Xi)A_j^\dagger(\Xi). \tag{3.14}$$

Because this holds for all $U\in$SU(D), Schur's lemma implies that

$$\sum_{j=1}^N A_j(\Xi)A_j^\dagger(\Xi)=c\mathbb{1}, \tag{3.15}$$

where c is a constant. Taking the trace of both sides of Eq. (3.15) we find

$$\mathrm{Tr}\left(\sum_{j=1}^N A_j(\Xi)A_j^\dagger(\Xi)\right)N=c\,\mathrm{Tr}(\mathbb{1})=cN, \tag{3.16}$$

so that $c=1$. Because this relation holds for any program state, we have that

$$\sum_{j=1}^N A_{jk_1}A^\dagger_{jk_2}=\delta_{k_1k_2}\mathbb{1}_d , \tag{3.17}$$

which implies that the maps produced by a processor that is covariant with respect to SU(D) are unital.

Let us briefly consider an example in order to show that a nontrivial covariant processor with respect to SU(2) exists. We shall examine a processor provided by the quantum information distributor [12]. The program state of this device consists of two qubits and the data state is one qubit. The unitary operator, G can be implemented by a sequence of four controlled-NOT gates. A controlled-NOT gate acting on qubits j and k, where j is the control bit and k is the target bit, is described by the operator

$$D_{jk}|m\rangle_j|n\rangle_k=|m\rangle_j|m\oplus n\rangle_k , \tag{3.18}$$

where $m,n=0$ or 1, and the addition is modulo 2. If we denote the data qubit as qubit 1 and the two program qubits as qubits 2 and 3, then the operator G for this processor is

$$G=D_{31}D_{21}D_{13}D_{12} . \tag{3.19}$$

For the set of program states, $\mathcal{S}$, we shall consider two-qubit states of the form

$$|\Xi\rangle=\alpha|\Xi_{00}\rangle_{23}+\beta|\Phi\rangle_{23} , \tag{3.20}$$

where

$$\begin{aligned}|\Xi_{00}\rangle&=\frac{1}{\sqrt{2}}(|0\rangle_2|0\rangle_3+|1\rangle_2|1\rangle_3),\\ |\Phi\rangle&=\frac{1}{\sqrt{2}}|0\rangle_2(|0\rangle_3+|1\rangle_3),\end{aligned} \tag{3.21}$$

α and β are real, and $\alpha^2+\beta^2+\alpha\beta=1$. If the data register at the input is described by the state ϱ_{in}, then at the output of the processor we find the data register in the state

$$\rho_{out}=(1-\beta^2)\varrho_{in}+\frac{\beta^2}{2}\mathbb{1}_d . \tag{3.22}$$

The action of this processor is clearly covariant with respect to any transformation in SU(2).

IV. PROCESSOR DESIGN

In the previous sections we have studied sets of superoperators that a given processor can perform. We would now like to turn the problem around and suppose that we have a given set of superoperators, and our aim is to construct a processor that will be able to execute them. We already know that it is impossible to find a processor that will perform all superoperators. In particular, if the set of superoperators we are trying to implement contains an uncountable set of unitary superoperators, then the set of superoperators cannot be performed by a single processor.

Here we will ask a more modest question: Under what circumstances are we able to find a processor that will perform some one-parameter set of superoperators? In particular, suppose that we have the superoperators T_θ, where the parameter θ varies over some range, and that these operators have a Kraus representation $\{B_j(\theta)|j=1,\dots,M\}$ such that

$$T_\theta[\rho]=\sum_{j=1}^{M} B_j(\theta)\rho B_j^\dagger(\theta). \tag{4.1}$$

Our aim is to find a unitary operator G and a set of program states $|\Xi(\theta)\rangle_p$ so that

$$T_\theta[\rho_d]=G(\rho_d\otimes|\Xi(\theta)\rangle_{p\,p}\langle\Xi(\theta)|)G^\dagger. \tag{4.2}$$

The operators $A_j(\Xi)$ that represent the action of the processor on the data states when the program state is $|\Xi\rangle$ are now functions of θ and we shall denote them as $A_j(\theta)$. Our processor then transforms the input data state ρ_d into the output state $\rho_d^{(out)}$:

$$\rho_d^{(out)}=\sum_{j=1}^{N} A_j(\theta)\rho_d A_j^\dagger(\theta). \tag{4.3}$$

We note that the operators $\{A_j(\theta)|j=1,\dots,N\}$ also constitute a Kraus representation of the superoperator T_θ. The Kraus representation of a superoperator is not unique; any two different Kraus representations of the same superoperator, $\{B_j|j=1,\dots,M\}$ and $\{C_j|j=1,\dots,N\}$, where $N\geqslant M$, are related as follows [7]:

$$C_j=\sum_{k=1}^{N} U_{kj}B_k, \tag{4.4}$$

where U_{kj} is a unitary matrix. It is understood that if $N>M$ then zero operators are added to the set $\{B_j|j=1,\dots,M\}$ so that the two sets of operators have the same cardinality.

In what follows we will study two single-qubit quantum channels, the phase-damping channel and the amplitude-damping channel. We will show that the first can be realized by a finite quantum processor, while the second cannot.

A. Phase-damping channel

The phase-damping channel is described by the map T_θ that is determined by the Kraus operators $B_1(\theta)=\sqrt{\theta}\mathbb{1}$ and $B_2(\theta)=\sqrt{1-\theta}\sigma_z$, where both σ_z and $\mathbb{1}$ are unitary operators, and $0\leqslant\theta\leqslant1$ [13,2]. Hence for the phase-damping map we find

$$T_\theta[\varrho_d]=\theta\mathbb{1}\varrho_d\mathbb{1}+(1-\theta)\sigma_z\varrho_d\sigma_z^\dagger, \tag{4.5}$$

where ϱ_d is the input qubit state. We can design the corresponding processor using Eq. (3.2), that is,

$$G^{phase}|\phi\rangle_d\otimes|k\rangle_p=(U_k|\phi\rangle_d)\otimes|k\rangle_p, \tag{4.6}$$

where $k=1,2$ and $U_1=\mathbb{1},U_2=\sigma_z$. The program state in which the required transformation T_θ is encoded is given by $|\Xi(\theta)\rangle_p=\sqrt{\theta}|0\rangle_p+\sqrt{1-\theta}|1\rangle_p$. Note that in this case the program operators $A_j(\theta)$ for $j=1,2$, are equal to the corresponding Kraus operators, i.e., $A_j(\theta)=B_j(\theta)$. Therefore, we can execute the entire one-parameter set of superoperators T_θ merely by changing the program state we send into the processor, and the dimension of the program space is 2.

B. Amplitude-damping channel

The amplitude-damping map S_θ is given by the Kraus operators $B_1(\theta)=|0\rangle\langle0|+\sqrt{1-\theta}|1\rangle\langle1|$ and $B_2(\theta)=\sqrt{\theta}|0\rangle\langle1|$, where again $0\leqslant\theta\leqslant1$. In designing a processor to realize this channel, we would again like to assume that the program operators are the same as the Kraus operators $B_1(\theta)$ and $B_2(\theta)$. In this case, however, we have a problem. The program operators must satisfy Eq. (2.24), but

$$\sum_{j=1}^{2} B_j^\dagger(\theta_1)B_j(\theta_2)=|0\rangle\langle0|+[\sqrt{\theta_1\theta_2}+\sqrt{(1-\theta_1)(1-\theta_2)}]|1\rangle\langle1|, \tag{4.7}$$

and the right-hand side of this equation is not, in general, proportional to the identity.

What we now must do is try to find a Kraus representation for this channel that does satisfy Eq. (2.24). In particular, we assume that

$$C_k(\theta)=\sum_{k=1}^{N} U_{kj}(\theta)B_j(\theta), \tag{4.8}$$

where $U(\theta)$ is an $N\times N$ unitary matrix, and $B_j(\theta)=0$ for $j>2$. In addition, we want

$$\sum_{j=1}^{N} C_j^\dagger(\theta_1)C_j(\theta_2)=f(\theta_1,\theta_2)\mathbb{1}, \tag{4.9}$$

where $f(\theta_1,\theta_2)$ is a function whose magnitude is less than or equal to 1. The operators $C_j(\theta)$ would then be candidates for the program operators $A_j(\theta)$. What we will show is that there is no Kraus representation with N finite that satisfies these conditions. Because the number of program operators is equal to the dimension of the program space, this will show that there is no finite quantum processor that can realize the family of superoperators that describes the amplitude-damping channel.

If Eq. (4.9) is to hold, then the coefficients of $|0\rangle\langle0|$ and $|1\rangle\langle1|$ must be the same. Inserting the explicit expressions for $C_j(\theta)$ in terms of $B_1(\theta)$ and $B_2(\theta)$, this condition becomes

$$[1-\sqrt{(1-\theta_1)(1-\theta_2)}]\sum_{j=1}^{N} U_{1j}^{*}(\theta_1)U_{1j}(\theta_2)$$
$$=\sqrt{\theta_1\theta_2}\sum_{j=1}^{N} U_{2j}^{*}(\theta_1)U_{2j}(\theta_2). \tag{4.10}$$

We can now make use of the fact that the rows of a unitary matrix constitute orthonormal vectors and the Schwarz inequality to show that the magnitude of the sum on the right-hand side of this equation is less than or equal to 1. This give us that

$$\left|\sum_{j=1}^{N} U_{1j}^{*}(\theta_1)U_{1j}(\theta_2)\right|\leqslant\frac{\sqrt{\theta_1\theta_2}}{1-\sqrt{(1-\theta_1)(1-\theta_2)}}. \tag{4.11}$$

We now need the result that if $\{v_j|j=1,\ldots,N\}$ are vectors of length 1, and $|\langle v_j|v_k\rangle|<1/(N-1)$, then $\{v_j|j=1,\ldots,N\}$ are linearly independent [14]. The proof is quite short, so we give it here. If the vectors are linearly dependent, then there are constants c_j, at least some of which are not zero, such that

$$\sum_{j=1}^{N} c_j|v_j\rangle=0. \tag{4.12}$$

Taking the inner product of both sides with $|v_k\rangle$ we find that

$$|c_k|=\left|\sum_{j\neq k} c_j\langle v_k|v_j\rangle\right|$$
$$<\frac{1}{N-1}\sum_{j\neq k}|c_j|. \tag{4.13}$$

Summing both sides of the above inequality over k gives us that

$$\sum_{k=1}^{N}|c_k|<\frac{1}{N-1}\sum_{k=1}^{N}\sum_{j\neq k}|c_j|=\sum_{k=1}^{N}|c_k|, \tag{4.14}$$

which is clearly impossible. Therefore, the vectors must be linearly independent.

This can now be applied to the first row of the unitary matrix $U(\theta)$, which we can think of as an N-component normalized vector, which we shall call $u_0(\theta)$. What we will show is that we can find arbitrarily many of these vectors whose inner products can be made arbitrarily small. The result in the previous paragraph then implies that these vectors are linearly independent, but this contradicts the fact that they lie in an N-dimensional space. Hence, there must be an infinite number of Kraus operators, and the program space must be infinite dimensional.

In order to study the inner products of the vectors $u_0(\theta)$ for different values of θ, we need to examine the function appearing on the right-hand side of Eq. (4.11):

$$g(\theta_1,\theta_2)=\frac{\sqrt{\theta_1\theta_2}}{1-\sqrt{(1-\theta_1)(1-\theta_2)}}. \tag{4.15}$$

Using the fact that if $0\leqslant\theta\leqslant 1$, then $\sqrt{1-\theta}\leqslant 1-(\theta/2)$, we have that for $0\leqslant\theta_j\leqslant 1$, $j=1,2$,

$$g(\theta_1,\theta_2)\leqslant\frac{2\sqrt{\theta_1\theta_2}}{\theta_1+\theta_2-(\theta_1\theta_2/2)}. \tag{4.16}$$

Finally, noting that for θ_1 and θ_2 between 0 and 1

$$\frac{\theta_1+\theta_2}{\theta_1+\theta_2-(\theta_1\theta_2/2)}\leqslant\frac{4}{3}, \tag{4.17}$$

we see that

$$g(\theta_1,\theta_2)\leqslant\frac{8\sqrt{\theta_1\theta_2}}{3(\theta_1+\theta_2)}. \tag{4.18}$$

We can make use of this bound, if we choose, for any positive integer M, the sequence $\zeta_n=[1/(16M^2)]^n$, where $n=1,\ldots$. If $\theta_1=\zeta_n$ and $\theta_2=\zeta_m$ where $m>n$, then

$$g(\theta_1,\theta_2)\leqslant\frac{8}{3}\frac{1}{(4M)^{m-n}}. \tag{4.19}$$

The vectors $\{u_0(\zeta_m)|m=1,\ldots,M\}$ have pairwise inner products whose magnitudes are less than $1/M$, and, therefore, they are linearly independent. As these vectors have N components, if we choose $M>N$ we have a contradiction. This, as we stated before, implies that the number of Kraus operators is infinite, and that the amplitude-damping channel cannot be realized by a finite quantum processor.

V. CONCLUSION

In this paper we have presented a theory of programmable quantum processors that allows us to realize completely positive maps on quantum systems. We have introduced several classes of quantum processors and have discussed the design of processors to realize particular classes of superoperators. In our discussion we focused on the situation when no measurements are performed on the program register.

In concluding this paper let us briefly comment on the fact that, if we allow dynamics conditioned on the results of measurements on the program register, additional classes of maps can be realized. One version of quantum processors with conditional dynamics, whose operating principle is that of quantum teleportation, was discussed by Nielsen and Chuang [2]. Here we shall present a different example. Consider a processor consisting or a single C-NOT gate in which the program register consists of the control qubit, and the data register consists of the data qubit. If the program qubit is initially in the state

$$|\Xi\rangle_p=\alpha|0\rangle+\beta|1\rangle, \tag{5.1}$$

and the data qubit in the state $|\psi\rangle_d$, then the output of our simple processor is the state

$$|\Phi_{out}\rangle_{dp} = \alpha|\psi\rangle_d|0\rangle_p + \beta\sigma_x|\psi\rangle_d|1\rangle_p . \tag{5.2}$$

If we trace out the program register we obtain the output density matrix

$$\rho_{out}^{(1)} = |\alpha|^2\rho_{in} + |\beta|^2\sigma_x\rho_{in}\sigma_x , \tag{5.3}$$

where $\rho_{in}=|\psi\rangle_d\langle\psi|$. If, on the other hand, we measure the output of the program register in the $|\pm x\rangle$ basis, where

$$|\pm x\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle), \tag{5.4}$$

and accept the output of the data register only if we get $|+x\rangle$, then we find for the output state of the data register

$$\rho_{out}^{(2)} = K(\alpha\mathbb{1} + \beta\sigma_x)\rho_{in}(\alpha^*\mathbb{1} + \beta^*\sigma_x), \tag{5.5}$$

where K is a normalization constant. We note that the sets of mappings realized by the two different procedures are not the same.

While the addition of conditional measurements to quantum processors allows us to realize a different set of mappings, there is, however, a cost. The procedure has a certain probability of failing, although we do know whether it has succeeded or not. The failure probability depends on both the program and on the data state.

It was shown by Vidal and Cirac that it is possible to increase the probability of success by increasing the dimensionality of the program register [14]. They started with a single C-NOT processor (in their case the control qubit was the data qubit and the target qubit was the program qubit) that implemented the one-parameter set of unitary operations $U(\alpha)=\exp(i\alpha\sigma_z)$ on the data qubit. The probability of success is 1/2. By increasing the size of the program to two qubits and adding a Toffoli gate, they were able to increase this probability to 3/4. Adding yet more qubits to the program and gates to the processor allowed them to make the success probability as close to 1 as they wished.

Another type of probabilistic quantum processor, based on the quantum cloning circuit, was studied by us in an earlier paper [15]. Its qubit version (it can be generalized to qudits) can implement any linear operator (up to normalization) on the input qubit state. There are still many open questions with respect to probabilistic quantum processors, and we will study some of them in a forthcoming publication [16].

ACKNOWLEDGMENTS

We thank Matyas Koniorczyk for stimulating discussions. This work was supported in part by the European Union projects EQUIP (Grant No. IST-1999-11053), QUBITS (Grant No. IST-1999-13021), QUEST (Grant No. HPRN-CT-2000-00121), and by the National Science Foundation under Grant No. PHY-9970507.

[1] S. Lloyd and L. Viola, Phys. Rev. A **65**, 010101 (2002).
[2] See, for example, M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, England, 2000).
[3] G. Harel and V. M. Akulin, Phys. Rev. Lett. **82**, 1 (1999).
[4] S. Lloyd and S. L. Braunstein, Phys. Rev. Lett. **82**, 1784 (1999).
[5] B. Hladký, G. Drobný, and V. Bužek, Phys. Rev. A **61**, 022102 (2000).
[6] M. A. Nielsen and I. L. Chuang, Phys. Rev. Lett. **79**, 321 (1997).
[7] J. Preskill, http://www.theory.caltech.edu/people/ preskill
[8] A. Uhlmann, Rep. Math. Phys. **9**, 273 (1976); **24**, 229 (1986).
[9] M. Ziman, P. Štelmachovič, V. Bužek, M. Hillery, V. Scarani, and N. Gisin, Phys. Rev. A **65**, 042105 (2002).
[10] D. Nagaj, P. Štelmachovič, V. Bužek, and M. S. Kim (unpublished).
[11] Whether there are any nontrivial covariant processors in the case that $\mathcal{S}=\mathcal{H}_p$ is an open question.
[12] S. Braunstein, V. Bužek, and M. Hillery, Phys. Rev. A **63**, 052313 (2001).
[13] A. S. Holevo, *Probabilistic and Statistical Aspects of Quantum Theory* (North-Holland, Amsterdam, 1982).
[14] G. Vidal and J. I. Cirac, e-print quant-ph/0012067; see also G. Vidal, L. Masanes, and J. I. Cirac, e-print quant-ph/0102037.
[15] M. Hillery, V. Bužek, and M. Ziman, Phys. Rev. A **65**, 022301 (2002).
[16] M. Hillery *et al.* (unpublished).