Santa Fe Institute Working Paper 06-09-031
arxiv.org/quant-ph/0608206

Computation in Finitary Stochastic and Quantum Processes

Karoline Wiesner[1, 2, *] and James P. Crutchfield[1, 2, †]

[1] *Center for Computational Science & Engineering and Physics Department, University of California Davis, One Shields Avenue, Davis, CA 95616*
[2] *Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501*

(Dated: July 6, 2007)

We introduce *stochastic* and *quantum finite-state transducers* as representations of classical stochastic and quantum finitary processes. Formal *process languages* serve as the literal representation of the behavior of these processes and are recognized and generated by subclasses of stochastic and quantum transducers. We compare deterministic and nondeterministic stochastic and quantum automata, summarizing their relative computational power in a hierarchy of finitary process languages. Quantum finite-state transducers and generators represent a first step toward a computational description of individual closed quantum systems observed over time. They are explored via several physical examples, including the iterated beam splitter, an atom in a magnetic field, and atoms in an ion trap—a special case of which implements the Deutsch quantum algorithm. We show that the behavior of these systems, and so their information processing capacity, depends sensitively on the measurement protocol.

PACS numbers: 05.45.-a 03.67.Lx 03.67.-a 02.50.-r 89.70.+c

Contents

I. Introduction 1
A. Physical Motivations 2
B. Technical Setting 2
C. Results 3
D. Overview 3

II. Finitary Stochastic Processes 3

III. Stochastic Transducers 5
A. Graph Representation 5
B. Word Probabilities 6

IV. Stochastic Recognizers and Generators 6
A. Stochastic Recognizers 6
B. Stochastic Generators 8
C. Properties 9

V. Finitary Quantum Processes 10
A. States in a Quantum System 11
B. Measurement 11

VI. Quantum Transducers 12
A. Word distributions 13
B. Properties 13

VII. Quantum Recognizers and Generators 14
A. Quantum Recognizers 14
B. Alternative Quantum Recognizers 15
C. Quantum Generators 15
D. Density Matrix Formalism 15
E. Hierarchy of Finitary Process Languages 16

VIII. Quantum Generators and Finitary Processes: Examples 18
A. Two-State Quantum Processes 18
1. Iterated Beam Splitter 18
B. Three-State Quantum Processes 20
1. Golden Mean Quantum Machine 20
2. Quantum Even Process 21
C. Four-State Quantum Process 21
1. Quantum Transducer For Trapped Ions 21
2. Deutsch Algorithm as a Special Case 23

IX. Concluding Remarks 24

Acknowledgments 25

References 25

I. INTRODUCTION

Automata theory is the study of abstract computing devices, or *machines*, and the class of functions they can perform on their inputs. In the 1940's and 1950's, simple kinds of machines, so-called *finite-state automata*, were introduced to model brain function [1, 2]. They turned out to be extremely useful for a variety of other purposes, such as studying the lower limits of computational power and synthesizing logic controllers and communication networks. In the late 1950's, the linguist Noam Chomsky developed a classification of *formal languages* in terms of the *grammars* and automata required to recognize them [3]. On the lowest level of Chomsky's hierarchy, for example, whether or not a given sentence obeys the grammatical rules of a language is answered by a finite-state automaton.

Our understanding of the nature of computing has changed substantially in the intervening half century. In

*Electronic address: karoline@cse.ucdavis.edu
†Electronic address: chaos@cse.ucdavis.edu

recent years the study of computation with elementary components that obey quantum mechanical laws has developed into a highly active research area.

The physical laws underlying quantum computation are a mixed blessing. There is a growing body of theoretical results suggesting that a computational device whose components are directly governed by quantum physics may be considerably more powerful than its classical counterpart. Undoubtedly, the most celebrated of these results is Shor's factoring algorithm from 1994 [4]. Other results include Grover's quantum search algorithm from 1996 [5]. These results assume the use of powerful computational architectures, such as quantum Turing machines [6], that are decidedly more powerful than finite-state machines. For a review of theoretical and experimental studies of quantum computation see, for example, Refs. [7, 8].

However, to date, implementation efforts have fallen substantially short of the theoretical promise. So far experimental tests of quantum computation are finite—in fact, very finite. Currently, the largest coherent system of information storage is 7 quantum bits or *qubits* [9]. Quantum finite-state automata have drawn much interest in the last decade for this reason. They reflect the capabilities of currently feasible quantum computers. Thus, the study of finite-state quantum automata is motivated by very practical concerns. As was also true in the first days of digital computers, it is also the starting point for developing a computational hierarchy for quantum dynamical systems.

A. Physical Motivations

Recent developments in experimental quantum computation allow for the preparation and control of individual quantum systems [10] and the control of quantum systems through repeated measurements [11]. One area of investigation that combines the two techniques, but is as yet unrelated to quantum computation, is single molecule spectroscopy [12–14] where the experimentalist attaches a single molecule to a substrate and records its time-dependent fluorescence over milliseconds. This is an example of a single quantum molecular system measured repeatedly over time. Quantum theory, in contrast, often focuses only on predicting the expectation of outcomes from an ensemble of isolated measurements; that is, it predicts an observable's mean value. For molecular and quantum processes this is insufficient, since one needs to describe a system's *behavior*.

Quantum mechanics can be extended to address behavior [15], but the resulting formalism leaves unanswered important questions about a system's computational capacity. That is, given a natural system—say, a molecule that is irradiated and simply behaves in response—what is its capacity to store its history and process that information? Indeed, even if a system is designed to have a desired capacity, a question always remains about whether or not that capacity is actually used during operation. Moreover, for quantum systems, it is essential to include measurement in any description. Observation must be the basis for modeling a quantum process—either its behavior or its computational capacity. Here we introduce a computation-theoretic description of observed quantum processes that, using a combination of tools from quantum mechanics and stochastic processes, attempts to address the issues of control, measurement, and behavior.

An intriguing, but seemingly unrelated area of research in quantum mechanics is quantum chaos. Since any quantum dynamical system is described by the Schrödinger equation, which is linear, no chaotic behavior can arise. However, quantum systems that exhibit chaotic behavior in the classical limit, also show signatures of chaos in a semi-classical regime [16]. It turns out that measurement interaction leads to genuinely chaotic behavior in quantum systems, even far from the semi-classical limit [17]. Studies of quantum chaos are, in effect, extensions of the theory of nonlinear classical dynamics. The behavior of nonlinear classical dynamical systems can be coded by finite-state automata, using *symbolic dynamics* [18]. The quantum automata introduced in the following provide a similar kind of link for quantum systems; see, in particular, Ref. [19].

B. Technical Setting

In the following we develop a line of inquiry complementary to both quantum computation and quantum dynamical systems by investigating the intrinsic computation of quantum processes. *Intrinsic computation* in a dynamical system is an inherent property of the behavior the system generates [20]. One asks three basic questions of the system: First, how much historical information is stored in the current state? Second, in what architecture is that information stored? Finally, how is the stored information transformed to produce future behavior? This approach has been used to analyze intrinsic computation in classical dynamical systems and stochastic processes [21–23].

We view the present contribution as a direct extension of this prior work and, also, as complementary to the current design and theoretical-engineering approach to quantum computation. Specifically, we focus on the dynamics of quantum processes, rather than on methods to construct devices that implement a desired function and express the intrinsic information processing using various kinds of finite-memory devices. We emphasize the effects of measurement on a quantum system's behavior and so, in this way, provide a somewhat different view of quantum dynamical systems for which, typically, observation is often ignored. An information-theoretic analysis using the resulting framework can be found in Refs. [19, 24].

Most directly, we are interested, as natural scientists are, in *behavior*—how a system state develops over time.

In the computation-theoretic setting this translates into a need to represent behaviors as formal *process languages* and to model these via *generators*. In contrast, the conventional setting for analyzing the computational power of automata centers around detecting membership of words in a language. As a consequence, the overwhelming fraction of existing results on automata concerns devices that *recognize* an input string—and on problems that can be recast as such. A second, substantial fraction of work focuses on transducers, mostly for human-language recognition tasks. Automata that spontaneously generate outputs are much less often encountered, if at all, in the theory of computation. Nonetheless, generators are necessary if one wants to model physical processes using dynamical systems. In particular, as we hope to show, process languages and generators are key tools for answering questions about the information processing capabilities inherent in natural processes [24, 25].

C. Results

In this vein, we introduce a computation-theoretic model for quantum dynamical systems. Although quantum mechanical systems have received much attention in the last few years, there is a dearth of formal results on computational models of the behavior generated by quantum processes. The following provides such models at the lowest, finite-memory level of an as-yet partially understood quantum-computation hierarchy.

The computation-theoretic models are analyzed as if they are stochastic processes. The results give a way to represent and analyze computation in natural quantum mechanical systems, providing the foundation for methods to quantify intrinsic computation of quantum dynamical systems [24]. Quantum systems are prevalent in the molecular world, as we noted above. During its temporal evolution any such system stores some amount of historical information and uses this to generate its future behavior. With computation-theoretic models of quantum dynamics in hand, such as the ones we use here, a process's computational capacity can be analyzed information theoretically [24]. This is a goal with experimental consequences.

While such results should be useful for the design of quantum systems for computational tasks, design is not our goal in the following. Rather, the focus is on developing a finite-memory computational model for quantum processes and on how it can be used as a tool for identifying finite-memory processes in nature. As a practical consequence, since today all experiments testing quantum computation support only (very) finite memory and since the experiments are physical processes (in the sense in which we use the phrase), the results should be of immediate use in analyzing experimental systems.

D. Overview

Due to the range of topics, in the following we give a self-contained treatment. We review what is needed from automata, formal languages, and quantum theory, though familiarity with those areas is helpful. Citations to reference texts are given at the appropriate points.

Our approach will make most sense, especially to those unfamiliar with the theory of formal languages, if we devote some time to reviewing basic automata theory and its original goals. This also allows us to establish, in a graded fashion, the necessary notation for the full development, clearly identifying which properties are quantum mechanical and which, in contrast, are essentially classical (and probabilistic). In addition, it illustrates one of the principle benefits of discrete computation theory: i.e., the classification of devices that implement different kinds of computation. Those for whom automata and formal languages are well known, though, should appreciate by the end of the review the physical and dynamical motivations, since these will be expressed within the existing frameworks of discrete computation and stochastic processes.

To lay the foundations for a computational perspective on quantum dynamical systems we introduce a class of finite-state automata called *quantum finite-state transducers*. In the next sections we introduce the concept of *process languages*, building on formal language theory. We then present stochastic finite-state transducers and their subclasses—stochastic recognizers and generators—as classical representations of process languages. The relationship between automata and languages is discussed in each case and we provide an overview (and introduce notation) that anticipates their quantum analogs. We then introduce quantum finite-state transducers and their subclasses—quantum recognizers and generators—and discuss their various properties. Finally, we illustrate the main ideas by analyzing specific examples of quantum dynamical systems that they can model.

II. FINITARY STOCHASTIC PROCESSES

Consider the temporal evolution of the state of some natural system. The evolution is monitored by a series of measurements—numbers registered in some way, perhaps continuously, perhaps discretely. Each such measurement can be taken as a random variable. The distribution over sequences of these random variables is what we refer to as a *stochastic process*. An important question for understanding the structure of natural systems is what kinds of stochastic processes there are.

The class of *finitary* stochastic processes was introduced to identify those that require only a finite amount of internal resources to generate their behavior. This property is important in several settings. In symbolic dynamical systems, for example, it was shown that the *sofic subshifts* have a form of infinite correlation in their

temporal behaviors despite being finitely specified [26]. The information-theoretic characterization of stochastic processes [27], as another example, defines finitary processes as those with a bounded value of mutual information between past and future behaviors. Here, we remain close to these original definitions, giving explicit structural models, both classical and quantum, for finitary processes.

In this, we use formal language theory. Our use of formal language theory differs from most, though, in how it analyzes the connection between a language and the systems that can generate it. In brief, we observe a system through a finite-resolution measuring instrument, representing each measurement with a *symbol* σ from discrete *alphabet* Σ. The temporal behavior of a system, then, is a string or a *word* consisting of a succession of measurement symbols. The collection of all (and only) those words is the *language* that captures the possible, temporal behaviors of the system.

Definition. *A* formal language $\mathcal{L}$ *is a set of* words $w = \sigma_0\sigma_1\sigma_2\ldots$ *each of which consists of a finite series of symbols* $\sigma_t \in \Sigma$ *from a discrete alphabet* Σ.

In the following λ denotes the empty word. Σ^* denotes the set of all possible words, including λ, of any length formed using symbols in Σ. We denote a word of length L by $\sigma^L = \sigma_0\sigma_1\ldots\sigma_{L-1}$, with $\sigma_t \in \Sigma$. The set of all words of length L is Σ^L.

Since a formal language, as we use the term, is a set of observed words generated by a process, then each *subword* $\sigma_t\sigma_{t+1}\ldots\sigma_{u-1}\sigma_u$, $t \leq u$, $t, u = 0, 1, \ldots, L-1$, of a word σ^L has also been observed and is considered part of the language. This leads to the following definition.

Definition. *A language* $\mathcal{L}$ *is* subword closed *if, for each* $w \in \mathcal{L}$*, all of* w*'s subwords* $\mathrm{sub}(w)$ *are also members of* $\mathcal{L}$*:* $\mathrm{sub}(w) \subseteq \mathcal{L}$.

Finally, we imagine that a physical system can run for an arbitrarily long time and so the language describing its behaviors has words of arbitrary length. In this way, a subword-closed formal language—as a set of arbitrarily long series of measurements—represents the allowed (and, implicitly, disallowed) behaviors of a system.

Beyond a formal language listing which words (or behaviors) occur and which do not, we are also interested in the probability of their occurrence. Let $\Pr(w)$ denote the probability of word w, then we have the following.

Definition. *A* stochastic language $\mathcal{S}$ *is a formal language with a* word distribution $\Pr(w)$ *that is normalized at each length* L*:*

$$\sum_{\{\sigma^L \in \Sigma\}} \Pr(\sigma^L) = 1 \ , L = 1, 2, 3, \ldots \tag{1}$$

with $0 \leq \Pr(\sigma^L) \leq 1$.

Definition. *The* joint probability *of symbol* σ *following word* w *is written* $\Pr(w\sigma)$.

Definition. *The* conditional probability $\Pr(\sigma|w)$ *of symbol* σ *given the preceding observation of word* w *is*

$$\Pr(\sigma|w) = \Pr(w\sigma)/\Pr(w) \ . \tag{2}$$

For purposes of comparison between various computational models, it is helpful to refer directly to the set of words in a stochastic language $\mathcal{S}$. This is the *support* of a stochastic language:

$$\mathrm{supp}\,(\mathcal{S}) = \{w \in \mathcal{S} : \ \Pr(w) > 0\} \ . \tag{3}$$

These lead us, finally, to define the main object of study.

Definition. *A* process language $\mathcal{P}$ *is a stochastic language that is subword closed. It obeys the consistency condition* $\Pr(\sigma^L) \geq \Pr(\sigma^L\sigma)$.

A process language represents all of a system's possible behaviors, $w \in \mathrm{supp}\,(\mathcal{P})$, and their probabilities $\Pr(w)$ of occurrence. In its completeness it could be taken as a model of the system, but at best it is a rather prosaic and unwieldy representation. Indeed, a *model* of a process is usually intended to be a more compact description than a literal listing of observations. In the best of circumstances a model's components capture some aspect of a system's structure and organization. Here we will be even more specific, the models that we will focus on not only have to describe a process language, but they will also consist of two structural components: states and transitions between them. (One should contrast the seeming obviousness of the latter with the fact that there are alternative computational models, such as grammars, which do not use the concept of state.)

To illustrate process languages we give an example in Fig. 1, which shows a language—from the *Golden Mean Process*—and its word distribution at different word lengths. In this process language $\Sigma = \{0, 1\}$ and word 00 and all words containing it have zero probability. Moreover, if a 1 is seen, then the next $\sigma \in \Sigma$ occurs with fair probability.

Figure 1 plots the base-2 logarithm of the word probabilities versus the binary string σ^L, represented as the base-2 real number $0.\sigma^L = \sum_{t=0}^{L-1} \sigma_t 2^{-t-1} \in [0, 1]$. At length $L = 1$ (upper leftmost plot) both words 0 and 1 are allowed but have different probabilities. At $L = 2$ the first disallowed string 00 occurs. As L grows an increasing number of words are forbidden—those containing the shorter forbidden word 00. As $L \to \infty$ the set of allowed words forms a self-similar, uncountable, closed, and disconnected (Cantor) set in the interval $[0, 1]$ [18]. Note that the language is subword closed. The process's name comes from the fact that the logarithm of the number of allowed words grows exponentially with L at a rate given by the logarithm of the golden mean $\phi = \frac{1}{2}(1 + \sqrt{5})$.

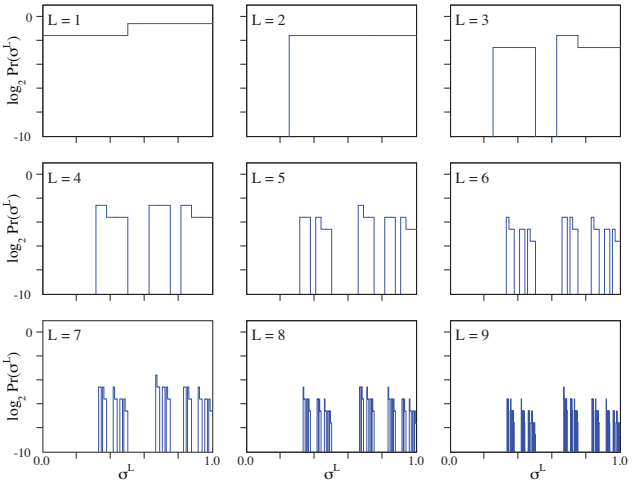

FIG. 1: Example of a process language: In the Golden Mean Process, with alphabet $\Sigma = \{0, 1\}$, word 00 and all words containing it have zero probability. All other words have nonzero probability. The logarithm base 2 of the word probabilities is plotted versus the binary string σ^L, represented as base-2 real number "$0.\sigma^L$". To allow word probabilities to be compared at different lengths, the distribution is normalized on $[0, 1]$—that is, the probabilities are calculated as densities.

III. STOCHASTIC TRANSDUCERS

The process languages developed above require a new kind of finite-state machine that can represent them. And so, our immediate goal is to construct a consistent formalism for machines that can recognize, generate, and transform process languages. We refer to the most general of the following representations as *stochastic transducers*. We will then specialize these transducers into recognizers and generators.

A few comments on the various models of stochastic transducers introduced by other authors will help to introduce our approach, which has the distinct goal of representing process languages. Paz defines *stochastic sequential machines* that are, in effect, transducers [28]. Rabin defines *probabilistic automata* that are stochastic sequential machines with no output [29]. Both models are capable of generating process languages, though not recognizing them. Neither, though, considers process languages or the "generation" of any language for that matter. Vidal et al define *stochastic transducers*, though based on a different definition of stochastic language [30]. As a result, their stochastic transducers cannot represent process languages.

In our definition of a *stochastic transducer* we follow Paz's definition of stochastic sequential machines.

Definition. *A* stochastic finite-state transducer *(ST) is a tuple* $\{S, X, Y, \{T(y|x)\}\}$ *where*

1. *S is a finite set of states. S includes a start state s_0.*
2. *X and Y are finite alphabets of input and output symbols, respectively.*
3. *$\{T(y|x) : x \in X, y \in Y\}$ is a set of square substochastic matrices of* dim $|S|$*, one for each input/output pair $y|x$.*
4. *The matrix entries $T_{ij}(y|x)$ define the conditional probability, when in state i, of going to state j and reading in symbol x and emitting symbol y.*

Generally, an ST operates by reading in symbols that, along with the current state, determine the next state(s) and output symbol(s). At each step a symbol $x \in X$ is read from the input word. The transducer stochastically chooses a transition $T_{ij}(y|x) > 0$, emits symbol $y \in Y$, and updates its state from i to j. An ST thus maps an input word to one or more output words. Unless otherwise explicitly stated, in our models there is no delay between reading an input symbol and producing the associated output symbols.

STs are our most general model of finitary (and nonquantum) computation. They are structured so that specialization leads to a graded family of models of increasing sophistication.

A. Graph Representation

The set $\{T(y|x)\}$ can be represented as a directed graph $\mathcal{G}(T)$ with the nodes corresponding to states—the matrix row and column indices. An edge connects two nodes and corresponds to an element $T_{ij} > 0$ that gives the nonzero transition probability from state i to state j. Edges are labeled $x|p|y$ with the input symbol $x \in X$, output symbol $y \in Y$, and transition probability $p = T_{ij}(y|x)$. Since an ST associates outputs with transitions, in fact, what we have defined is a *Mealy ST*, which differs from the alternative, and equivalent, *Moore ST* in which an output is associated with a state [28].

Definition. *A* path *in a machine is a sequence of edges with $T_{ij} > 0$.*

Definition. *A directed graph $\mathcal{G}$ is* connected *if there is at least one path between every pair of states.*

Definition. *A directed graph $\mathcal{G}$ is* strongly connected *if for every pair of states, i and j, there is at least one path from i to j and at least one from j to i.*

The states in the graph of an ST can be classified as follows, refining the definitions given by Paz [28, p. 85].

Definition. *A state j is a* consequent *of state i if there is a path beginning at i and ending at j.*

Definition. *A state is called* transient *if it has a consequent of which it is not itself a consequent.*

Definition. *A state is called* recurrent *if it has at least one consequent of which it is itself a consequent.*

Note that transient and recurrent states can be overlapping sets. We therefore make the following distinctions.

Definition. *A state is called* asymptotically recurrent *if it is recurrent, but not transient.*

Definition. *A state is called* transient recurrent *if it is transient and recurrent.*

Generally speaking, an ST starts in a set of transient states and ultimately transits to one or another of the asymptotically recurrent subsets. That is, there can be more than one set of asymptotically recurrent states. Unless stated otherwise, though, in the following we will consider STs that have only a single set of asymptotically recurrent states.

B. Word Probabilities

Before discussing the process languages associated with an ST we must introduce the matrix notation required for analysis. To facilitate comparing classical stochastic models and their quantum analogs, we use Dirac's bra-ket notation: Row vectors $\langle\cdot|$ are called *bra* vectors; and column vectors $|\cdot\rangle$, *ket* vectors.

Notation. *Let* $|\eta\rangle = (1, 1, \ldots, 1, 1)^T$ *denote a column vector with* $|S|$ *components that are all* 1*s.*

Notation. *Let* $\langle\pi| = (\pi_0, \pi_1, \ldots, \pi_{|S|-1})$ *be a row vector whose components,* $0 \leq \pi_i \leq 1$*, give the* probability of being in state i*. The vector is normalized in probability:* $\sum_{i=0}^{|S|-1} \pi_i = 1$*. The* initial state distribution*, with all of the probability concentrated in the start state, is denoted* $\langle\pi^0| = (1, 0, \ldots, 0)$.

For a series of L input symbols the action of the corresponding ST is a product of transition matrices:

$$T(y^L|x^L) = T(y_0|x_0)T(y_1|x_1)\cdots T(y_{L-1}|x_{L-1}) \ ,$$

whose elements $T_{ij}(y^L|x^L)$ give the probability of making a transition from state i to j and generating output y^L when reading input x^L.

Starting in state distribution $\langle\pi^0|$, the state distribution $\langle\pi(y^L|x^L)|$ after reading in word x^L and emitting word y^L is

$$\langle\pi(y^L|x^L)| = \langle\pi^0|\, T(y^L|x^L) \ . \tag{4}$$

This can then be used to compute the probability of reading out word y^L conditioned on reading in word x^L:

$$\Pr(y^L|x^L) = \langle\pi(y^L|x^L)|\eta\rangle \ . \tag{5}$$

IV. STOCHASTIC RECOGNIZERS AND GENERATORS

We are ready now to specialize this general architecture into classes of recognizing and generating devices. In each case we address those aspects that justify our calling them models; viz., we can calculate various properties of the process languages that they represent directly from the machine states and transitions, such as the word distribution and statistical properties that derive from it.

Generally speaking, a recognizer reads in a word and has two possible outputs for *each* symbol being read in: *accept* or *reject*. This differs from the common model [31] of reading in a word of finite length and only at the end deciding to accept or reject. This aspect of our model is a consequence of reading in process languages which are subword closed.

In either the recognition or generation case, we will discuss only models for arbitrarily long, but finite-time observations. This circumvents several technical issues that arise with recognizing and generating infinite-length strings, which is the subject of ω-language theory of Büchi automata [32].

Part of the burden of the following sections is to introduce a number of specializations of stochastic machines. Although it is rarely good practice to use terminology before it is defined, in the present setting it will be helpful when tracking the various machine types to explain our naming and abbreviation conventions now.

In the most general case—in particular, when the text says nothing else—we will discuss, as we have just done, *machines*. These are input-output devices or transducers and we will denote this in any abbreviation with a capital T. These will be specialized to *recognizers*, abbreviated R, and *generators*, denoted G. Within these basic machine types, there will be various alternative implementations. We will discuss *stochastic* (S) and *quantum* (Q) versions. Within these classes we will also distinguish the additional subsidiary property *determinism*, denoted D.

As we noted above, the entire development concerns machines with a finite set of states. And so, we will almost always drop the adjectives "finite-state" and "finitary", unless we wish to emphasize these aspects in particular.

A. Stochastic Recognizers

Stochastic devices and how they recognize an input have been variously defined since the first days of automata theory. Rabin defined *probabilistic automata* in 1963 [29]. The probabilistic aspect arose from the construction of stochastic matrices for each input symbol. For a given state and input symbol the machine would stochastically transition to a next state. Acceptance of an input string x^L *with cut point* λ was then defined by repeatedly reading in the same string and determin-

ing that the acceptance probability was above threshold: $p(x) > \lambda$. Acceptance *with isolated cut point* was defined for some $\delta > 0$ with $|p(x) - \lambda| \geq \delta$.

The difference from our recognition via a word-probability threshold is the normalization over accepted strings of the same length. Thus, Rabin's probabilistic automata do not recognize stochastic languages as defined above, but merely assign a number between 0 and 1 to each word being read in. The same is true for the *stochastic sequential machines* defined by Paz [28].

Here we introduce a *stochastic recognizer* that applies isolated cut-point recognition to process languages.

Definition. *A* stochastic finite-state recognizer (SR) *is an ST with a reject state s_r, and $Y = \{accept, reject\}$. In addition:*

1. *No transitions are allowed from the reject state to any state but itself.*
2. *A stochastic matrix is obtained by summing over accept-transitions:*

$$\sum_{x \in X, j \in S} T_{ij}(accept|x) = 1 \ , \ \forall i \in S \ . \tag{6}$$

Condition 2 is necessary for the recognition of process languages. Due to the definition of $T(y|x)$, the probabilities of accepting a given input symbol and of accepting any other input symbol sum to 1. This is expressed in the following two conditions that are implied by the SR definition:

$$\begin{aligned} \Pr(accept|x) + \Pr(reject|x) &= 1 \\ \Pr(accept|x) + \Pr(accept|x' \neq x) &= 1 \end{aligned} \tag{7}$$

In the following we will not explicitly represent the reject state in the discussion of SRs. It can be simply thought of as a probability sink. An SR's operation is completely defined by the non-reject states due to the conditions in Eqs. (7). Thus, in stead of writing $T(accept|x)$, in the following we simplify notation and simply write $T(x)$.

The state-to-state transition matrix for an SR is stochastic:

$$T = \sum_{x \in X} T(x) \ . \tag{8}$$

Definition. *Given a process language $\mathcal{P}$, an SR* accepts *a word $w \in \mathcal{P}$ with threshold $0 \leq \delta \leq 1$, if and only if*

$$|\Pr(w) - \langle \pi^0 | T(w) | \eta \rangle \, | \leq \delta \ . \tag{9}$$

The first criterion for accepting a word is that the word leads the machine through a series of transitions with positive probability. That is, it accepts the support of the language. The second criterion is that the probability of accepting the word is equal to the word's probability within a threshold δ. Thus, an SR not only tests for membership in a formal language, it also recognizes a *function*: the probability distribution of the language. For example, if $\delta = 0$ the SR accepts exactly a process language's word distribution. If $\delta > 0$ it accepts the probability distribution with some fuzziness, still rejecting all probability-0 words. As mentioned before, recognition happens at each time step. This means that in practice the experimenter runs an ensemble of SRs on the same input. The frequency of acceptance can then be compared to the probability of the input string computed from the $T(x)$.

Definition. *The* stationary state distribution $\langle \pi^s |$, *which gives the asymptotic state visitation probabilities, is determined by the left eigenvector of $T(x)$:*

$$\langle \pi^s | = \langle \pi^s | \, T(x) \ , \tag{10}$$

normalized in probability: $\sum_{i=0}^{|S|-1} \pi_i^s = 1$.

For a series $x_0 x_1 \cdots x_{L-1}$ of input symbols the action of the corresponding SR upon acceptance is a product of transition matrices:

$$T(x^L) = T(x_0) T(x_1) \cdots T(x_{L-1}) \ ,$$

whose elements $T_{ij}(x^L)$ give the probability of making a transition from state i to j and generating output *accept* when reading input x^L.

If the SR starts in state distribution $\langle \pi^0 |$, the probability of accepting x^L is

$$\Pr(x^L) = \langle \pi^0 | T(x^L) | \eta \rangle \ . \tag{11}$$

The state distribution $\langle \pi(x^L) |$ after accepting word x^L starting in state distribution $\langle \pi^0 |$ is

$$\langle \pi(x^L) | = \langle \pi^0 | \, T(x^L) \ . \tag{12}$$

These seemingly simple expressions—e.g., for the probability of a single word—are actually costly to compute since the number of elements to be summed increases exponentially with L.

We have the following special class of stochastic recognizers.

Definition. *A* stochastic deterministic finite-state recognizer *(SDR) is a stochastic finite-state recognizer whose substochastic transition matrices $T(x)$ have at most one nonzero element per row.*

A word that is accepted by an SDR is associated with one and only one series of transitions. This allows us to give an efficient expression for the word distribution of the language exactly ($\delta = 0$) recognized by an SDR:

$$\Pr(x^L) = T_{s_0 s_1}(x_0) T_{s_1 s_2}(x_1) \cdots T_{s_{L-1} s_L}(x_{L-1}) \ , \tag{13}$$

where $s_0 s_1 \ldots s_L$ is the unique series of states along the path selected by x^L and where by $T_{ij}(x)$ we refer only to

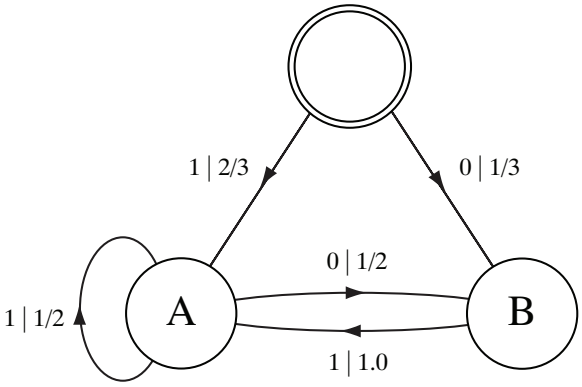

FIG. 2: Stochastic deterministic recognizer for the Golden Mean process language of Fig. 1. The edges are labeled $x|p$, where $x \in X$ and $p = T_{ij}(x)$. The start state $\langle \pi^0 | = (1, 0, 0)$ is double circled. The reject state and all transitions to it are omitted; as is the output *accept* on all edges.

the single component of $T(x)$ for the transition selected by x.

There is an important difference here with Eq. (11). Due to determinism, the computational cost for computing the word probability $\Pr(x^L)$ from SDRs increases only linearly with L; whereas it is exponential for SRs.

Figure 2 shows an example of an SDR that recognizes the Golden Mean process language. That is, it rejects any word containing two consecutive 0s and accepts any other word with respective nonzero probability. This leads, in turn, to the self-similar structure of the support of the word probability distribution noted in Fig. 1.

A useful way to characterize this property is to list a process language's *irreducible forbidden words*—the shortest disallowed words. In the case of the Golden Mean formal language, this list has one member: $\mathcal{F} = \{00\}$. Each irreducible word is associated with a family of longer words containing it. This family of forbidden words forms a Cantor set in the space of sequences, as described above. (Recall Fig. 1).

If we take the threshold to be $\delta = 0$, then the SDR recognizes only the process language shown in Fig. 1. If $\delta = 1$, in contrast, the SDR would accept process languages with any distribution on the Golden Mean process words. That is, it always recognizes the language's support.

One can easily calculate word probabilities and state distributions for the Golden Mean Process using the SDR's matrix representation.

$$T(0) = \begin{pmatrix} 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix} \text{ and } T(1) = \begin{pmatrix} 0 & \frac{2}{3} & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 \end{pmatrix} . \quad (14)$$

We use Eq. (11) with the start state distribution $\langle \pi^0 | = (1, 0, 0)$ to calculate the $L = 1$ word probabilities. Eq. (13) would be equally applicable.

$$\begin{aligned} \Pr(0) &= \langle \pi^0 | T(0) | \eta \rangle = \tfrac{1}{3} \ , \\ \Pr(1) &= \langle \pi^0 | T(1) | \eta \rangle = \tfrac{2}{3} \ . \end{aligned} \quad (15)$$

At $L = 3$ one finds for $\sigma^3 = 011$:

$$\begin{aligned} \Pr(011) &= \langle \pi^0 | T(011) | \eta \rangle \\ &= \langle \pi^0 | T(0) T(1) T(1) | \eta \rangle \\ &= \tfrac{1}{6} . \end{aligned} \quad (16)$$

In fact, all $L = 3$ words have the same probability, except for $\sigma^3 = 101$, which has a higher probability: $\Pr(101) = \frac{1}{3}$. (Cf. the $L = 3$ word distribution in Fig. 1.)

The conditional probability of a 1 following a 0, say, is calculated in a similarly straightforward manner:

$$\begin{aligned} \Pr(1|0) &= \frac{\Pr(01)}{\Pr(0)} \\ &= \frac{\langle \pi^0 | T(0) T(1) | \eta \rangle}{\langle \pi^0 | T(0) | \eta \rangle} \\ &= 1 \ . \end{aligned} \quad (17)$$

Whereas, the probability $\Pr(0|0)$ of a 0 following a 0 is zero, as expected.

B. Stochastic Generators

As noted in the introduction, finite-state machines generating strings of symbols can serve as useful models for structure in dynamical systems. They have been used as computational models of classical dynamical systems for some time; see Refs. [18, 21, 23, 25, 33–36], for example.

As we also noted, automata that only generate outputs are less often encountered in formal language theory [31] than automata operating as recognizers. One reason is that redefining a conventional recognizer to be a device that generates output words is incomplete. A mechanism for choosing which of multiple transitions to take when leaving a state needs to be specified. And this leads naturally to probabilistic transition mechanisms, as one way of completing a definition. We will develop finite-state generators by paralleling the development of recognizers in the previous section.

Definition. *A* stochastic finite-state generator (SG) *is an ST with* $|X| = 1$.

The input symbol can be considered a clock signal that drives the machine from state to state. The transition matrices can be simplified to $T(y|x) = T(y)$.

The state-to-state transition probabilities of an SG are given by the stochastic *state-to-state transition matrix*:

$$T = \sum_{y \in Y} T(y) \ . \quad (18)$$

The word probabilities are computed in the exact same way as those of an SR, detailed in Section IV A: One simply exchanges input symbols x with output symbols y. We obtain the following equation for word probabilities:

$$\Pr(y^L) = \langle \pi^0 | T(y^L) | \eta \rangle \ . \tag{19}$$

We define the following special class of SGs.

Definition. *A* stochastic deterministic finite-state generator *(SDG) is a stochastic finite-state generator in which each matrix $T(y)$ has at most one nonzero entry per row.*

As with recognizers, given the generator's state and an output symbol, the next state is uniquely determined. And, again, it is less costly to compute word probabilities.

$$\Pr(y^L) = T_{s_0 s_1}(y_0) T_{s_1 s_2}(y_1) \cdots T_{s_{L-1} s_L}(y_{L-1}) \ , \tag{20}$$

Given an initial state distribution, a sum is taken over those states, weighted by their probability. Even so, the computation increases only linearly with word length.

In the following we concentrate on deterministic finite-state generators. As an example, consider the generator for the Golden Mean process language. Its matrix representation is the same as for the Golden Mean recognizer given in Eqs. (14) and Fig. 2. Due to the latter's determinism, one can construct a generator simply by swapping input symbols to output symbols. (We return to the relationship between recognizers and equivalent generators shortly.) It turns out this is the smallest generator, but the proof of this will be presented elsewhere.

One can easily calculate word probabilities and state distributions for the Golden Mean Process using the SDG's matrix representation. We here introduce a way of computing these probabilities using the asymptotically recurrent states only. This is done using the stationary state distribution and the transition matrices restricted to the asymptotically recurrent states. The method is useful whenever the start state is not known, but the asymptotic behavior of the machine is. The transition matrices for the SDG, following Eqs. (14), become:

$$T(0) = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & 0 \end{pmatrix} \text{ and } T(1) = \begin{pmatrix} \frac{1}{2} & 0 \\ 1 & 0 \end{pmatrix} . \tag{21}$$

The stationary state distribution $\langle \pi^s |$ is calculated as the left eigenvector of the state-to-state transition matrix T, Eq. (18):

$$\langle \pi^s | = \langle \tfrac{2}{3}, \tfrac{1}{3} | \ . \tag{22}$$

Thus, each state is assigned a probability $\Pr(i) = \pi^s_i$, where π^s_i is the i^{th} component of $\langle \pi^s |$.

Assuming that the initial state is not known and that the process has been running for a long time, we use Eq. (19) with the stationary distribution $\langle \pi^s |$ to calculate the $L = 1$ word probabilities:

$$\Pr(0) = \langle \pi^s | T(0) | \eta \rangle = \tfrac{1}{3} \ , \tag{23a}$$

$$\Pr(1) = \langle \pi^s | T(1) | \eta \rangle = \tfrac{2}{3} \ . \tag{23b}$$

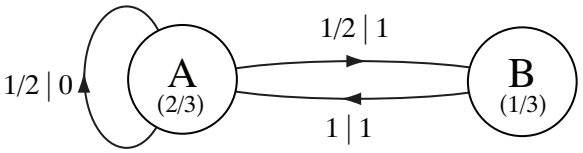

FIG. 3: A deterministic generator of the Even Process: Blocks of an even number of 1s are separated by 0s. Only the asymptotically recurrent states are shown. Edges are labeled $p \,|\, y$, where $y \in Y$ and $p = T_{ij}(y)$. The numbers in parentheses give a state's asymptotic probability.

At $L = 3$ one finds for $\sigma^3 = 011$:

$$\begin{aligned} \Pr(011) &= \langle \pi^s | T(011) | \eta \rangle \\ &= \langle \pi^s | T(0) T(1) T(1) | \eta \rangle \\ &= \tfrac{1}{6} . \end{aligned} \tag{24}$$

In fact, all $L = 3$ words have the same probability, except for $\sigma^3 = 101$, which has a higher probability: $\Pr(101) = \frac{1}{3}$. (Again, cf. the $L = 3$ word distribution in Fig. 1.)

Note that these are the same results that we calculated for the Golden Mean Process recognizer in the previous section. There, however, we used a different initial distribution. The general reason why these two calculations lead to the same result is not as obvious as one might think.

As a second example of a generator consider the Even Process whose language consists of blocks of even numbers of 1s bounded by 0s. The substochastic transition matrices for its recurrent states are

$$T(0) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix} \text{ and } T(1) = \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & 0 \end{pmatrix} . \tag{25}$$

The corresponding graph is shown in Fig. 3. Notice that the state-to-state transition matrix T is the same as the previous model of the Golden Mean Process. However, the Even Process is substantially different; and its SDG representation lets us see how. The set of irreducible forbidden words is countably infinite [26]: $\mathcal{F} = \{01^{2k+1}0 : \ k = 0, 1, 2, \ldots\}$. Recall that the Golden Mean Process had only a single irreducible forbidden word $\{00\}$. One consequence is that the words in the Even Process have a kind of infinite correlation: the "evenness" of the length of 1-blocks is respected over arbitrarily long words. This makes the Even Process effectively non-finite: As long as a sequence of 1s is produced, memory of the initial state distribution persists. Another difference is that the support of the word distribution has a countable infinity of distinct Cantor sets—one for each irreducible forbidden word. Thus, the Even Process falls into the broader class of *finitary* processes.

C. Properties

We can now describe the similarities and differences between stochastic and other kinds of recognizers and

between the various classes of generators.

Recall that $\mathcal{P}(\mathcal{M})$ denotes the process language associated with (recognized or generated by) automaton $\mathcal{M}$.

The relationships between the languages associated with the various machine types follow rather directly from their definitions. Essentially, we swap input and output alphabets and reinterpret the same transition matrices, either as specifying $x|p$ or $p|y$ as required. All, that is, except for the last two results, which may be unexpected.

Proposition 1. *For every SR,* supp $\mathcal{L}(SR)$ *is a regular language.*

Proof. *The graph of an SR, removing the probabilities, defines a finite-state recognizer and accepts, by definition, a regular language [31]. This regular language is the support of $\mathcal{L}(SR)$ by construction.*

Proposition 2. *For every SR, $\mathcal{L}(SR)$ is a process language.*

Proof. *The first property to establish is that the set of words recognized by an SR is subword closed: if $\Pr(y^L) > 0$, then all $w \in \mathrm{sub}(y^L)$ have $\Pr(w) > 0$. This is guaranteed by definition, see Condition 1 of the SR definition.*

The second property to establish is that the word distribution $\Pr(x^L)$ is normalized at each L. This follows from Condition 2 of the same definition.

Proposition 3. *SGs and SRs are equivalent: They recognize and generate the same set of languages, respectively: $\mathcal{P}(SG) = \mathcal{P}(SR)$.*

Proof. *Consider SG's transition matrices $T(y)$ and form a new set $T(accept|x)$ in which $X = Y$. The $T(accept|x)$ together with $T(reject|x)$, where $T(reject|x)_{is_r} = 1 - \sum_{j \in S} T(accept|x)_{ij}$, define an SR that recognizes $\mathcal{P}(SG)$. It follows that $\mathcal{P}(SG) \subseteq \mathcal{P}(SR)$.*

Now consider SR's transition matrices $T(accept|x)$ and form a new set $T(y)$ in which $X = Y$. The $T(y)$ define an SG that generates $\mathcal{P}(SR)$. It follows that $\mathcal{P}(SG) = \mathcal{P}(SR)$.

Corollary 1. *For every SG,* supp $\mathcal{L}(SG)$ *is a regular language.*

Corollary 2. *For every SG, $\mathcal{L}(SG)$ is a process language.*

Corollary 3. *SDGs and SDRs are equivalent: They recognize and generate the same set of languages, respectively: $\mathcal{P}(SDG) = \mathcal{P}(SDR)$.*

These equivalences are intuitive and expected. They do not, however, hint at the following, which turn on the interplay between nondeterminism and stochasticity.

Proposition 4. *There exists an SG such that $\mathcal{P}(SG)$ is not recognized by any SDR.*

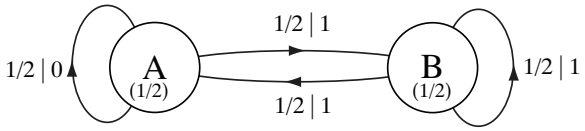

FIG. 4: A nondeterministic generator that produces a process language not recognized by any (finite-state) SDR. Only asymptotically recurrent states are shown. Edges are labeled $p \,|\, y$, where $y \in \{0, 1\}$ and $p = T_{ij}(y)$.

Proof. *We establish this by example. Consider the nondeterministic generator in Fig. 4, the* Simple Nondeterministic Source *(SNS). To show that there is no possible construction of an SDR we argue as follows. If a* 0 *appears, then the generator is in state* A*. Imagine this is then followed by a block 1^k. At each k the generator is in either state* A *or* B*. The probability of seeing a* 0 *next is ambiguous (either* 0 *or* $1/2$*) and depends on the exact history of internal states visited. Deterministic recognition requires that a recognizer be in a state in which the probability of the next symbol is uniquely given. While reading in* 1*s the recognizer would need a new state for each* 1 *connecting to the same state (state* A*) on a* 0*. Since this is true for all k, there is no finite-state SDR that recognizes the SNS's process language.*

Ref. [27] gives an SDR for this process that is minimal, but has a countably infinite number of states. Note that supp $\mathcal{P}(SNS)$ is the support of the Golden Mean process language.

Corollary 4. *There exists an SR such that $\mathcal{P}(SR)$ is not generated by any SDG.*

These propositions say, in essence, that deterministic machines generate or recognize only a subset of the finitary process languages. In particular, Props. 3, 4, and Cor. 3 imply proper containment: $\mathcal{P}(SDR) \subset \mathcal{P}(SG)$, and $\mathcal{P}(SDG) \subset \mathcal{P}(SR)$. This is in sharp contrast with the standard result in formal language theory: deterministic and nondeterministic automata recognize the same class of languages—the regular languages [31].

This ends our development of classical machines and their various specializations. We now enter the discussion of their quantum analogs, using a strategy that will be familiar by now. The reader should have a working knowledge of quantum theory at the level of, say, Ref. [37].

V. FINITARY QUANTUM PROCESSES

Consider any quantum system and its temporal evolution. As in the case of stochastic processes, the evolution of a quantum system is monitored by a series of measurements—numbers registered in some way. Each measurement can be taken to be the realization of a random variable. The distribution over sequences of these

random variables is what we call a *quantum process*. We will consider the finitary version of quantum processes in the same sense as used for the classical stochastic processes: The internal resources used during the evolution are finitely specified.

A. States in a Quantum System

Quantum mechanics is sometimes referred to as a generalization of classical probability theory with noncommuting probabilities. It is, therefore, helpful to compare classical stochastic automata and quantum automata and, in particular, to contrast the corresponding notions of state. The goal is to appreciate what is novel in quantum automata.

In the classical (stochastic) automaton setting an automaton has internal states and also a distribution over them. The distribution can be taken to be a "state" of the automaton. One interpretation of this "state" comes from considering how an observer monitors a series of outputs from a stochastic generator and predicts, with each observed symbol, the internal state the automaton is in. This prediction is a distribution over the internal states—one that represents the observer's best guess of the automaton's current internal state. The distribution is, in a sense, the "state" of the best predictor.

Similarly, there are several kinds of "state" that one might identify in a quantum automaton. Each quantum automaton will consist of *internal states* and we will take the state of the automaton to be a "superposition" over them; we call the latter the *state vector*. The crucial difference with classical (stochastic) automata is that this superposition over internal states is not a probability distribution. Internal states have complex amplitudes associated with them and therefore they potentially interfere. This, in turn, can affect the stochastic language associated with the quantum automaton.

In the vocabulary of quantum mechanics, at any moment in time a given quantum automaton is in a *pure state*, which is simply a superposition of its internal states. An observer's best guess as to the automaton's current state is a probability distribution over state vectors—the well known *density matrix*. Whenever the distribution contains more than one positive element we speak of a *mixed state*; otherwise, it is a *pure state*.

One can imagine, for example, a collection of individual quantum automata, each in a (pure) state, that is specified by a distribution of weights. One can also imagine a single quantum automaton being in different pure states at different moments in time. The "average" state then is also a mixed state. It is the latter picture that will be adopted here.

The various notions of "state" involved in a quantum automaton already hints at the relationship between states of an automaton and the state of a quantum system. The latter is completely described by its *state vector* which is a unit vector in the system's state space. This unit vector can be represented as a sum of *basis states* that span the state space. Choosing as a basis the eigenstates of an observable we find a simple correspondence between a state vector of a quantum system (a superposition of basis states) and a state of a quantum automaton (a distribution over internal states). Thus, we will use the terms *internal states* (of an automaton) and *basis states* (of a quantum state space) interchangeably, since they are constructed to be interchangeable. By similar reasoning, the terms *state vector* (of a quantum system) and *state* (of a quantum automaton) will be used interchangeably.

Definition. *A state vector $\langle\psi|$ is a unit vector in the system's state space. It can be expanded in terms of basis states $\langle\phi_i|$:*

$$\langle\psi| = \sum_{i=0}^{n-1} \langle\phi_i|\, c_i \ , \tag{26}$$

with $c_i \in \mathbb{C}$ and $\sum_{i=0}^{n-1} c_i^* c_i = 1$ and where n is the dimension of the state space.

The fact that a quantum pure state can be a superposition of basis states is regarded as the extra structure of quantum mechanics that classical mechanics does not have. In the following, we respect this distinction, building a hierarchy of quantum states that goes from basis states to superpositions of basis states to mixtures of superpositions. The analogous classical hierarchy goes from internal states to distributions over internal states to distributions over distributions over internal states. Due to the linearity of classical probability, a distribution over a distribution is in itself a distribution.

B. Measurement

Having delineated the various types of state for a quantum automaton and their analogs in a quantum dynamical system, we now turn to the measurement process which is crucial to the physical observation of a quantum dynamical system. In setting up an experiment, one makes choices of how and when to measure the state of a quantum system. These choices typically affect what one observes and in a way that differs radically from classical physical systems.

Measurement is the experimental means of characterizing a system in the sense that it is the observed symbols that determine the stochastic language and any subsequent prediction of the system's behavior. The measurement of a quantum mechanical system is mathematically described by a Hermitian operator that projects the current state onto one of the operator's eigenstates. After a measurement, the system is, with certainty, in one eigenstate. Such an operator is also called an *observable* and the *eigenvalues* corresponding to the eigenstates are the observed measurement *outcomes*.

When performing experiments on a quantum automaton, a measurement is defined similarly through an operator that projects the automaton's current state vector onto one of its internal (basis) states. The "observed" measurement outcome is emitted as a symbol labeling the transition entering that internal state.

VI. QUANTUM TRANSDUCERS

The study of quantum finite-state automata and the languages they recognize has produced a veritable zoo of alternative models which we review in Section VII B. Since we are, again, interested in recognition, generation, and transduction of process languages, we start out defining a quantum-finite state transducer. By specializing the latter we develop a series of quantum finite-state automaton models that are useful for recognition and generation and, ultimately, for modeling intrinsic computation in finitary quantum processes.

The quantum transducers we introduce shortly are designed to model a general experiment on a quantum dynamical system. As such they should be compared to the transducers of Ref. [38]. The transducers defined there include a measurement to determine acceptance, rejection, or continuation of the computation. In addition, they have a function mapping the current quantum state onto an output. This function, however, is not associated with a measurement process and lacks physical meaning. The quantum transducer defined below is in one-to-one correspondence to the quantum mechanical description of a physical experiment.

Definition. *A QT is a tuple* $\{Q, \langle\psi| \in \mathcal{H}, X, Y, \mathbf{T}(Y|X)\}$ *where*

1. $Q = \{q_i : i = 0, \ldots, n-1\}$ *is a set of* n internal states.
2. *The* state vector $\langle\psi|$ *lives in an* n*-dimensional Hilbert space* $\mathcal{H}$*; its initialization is defined as start state* $\langle\psi^0|$.
3. X *and* Y *are finite alphabets for input and output symbols, respectively.*
4. $\mathbf{T}(Y|X)$ *is a set of* $n \times n$ transition matrices $\{T(y|x) = U(x)P(y), x \in X, y \in Y\}$ *that are products of*
   (a) *a unitary matrix* $U(x) \in \mathbf{U} \equiv \{U(x) : x \in X\}$. $\mathbf{U}$ *is a set of* n*-dimensional* unitary operators *that govern the state vector's evolution; and*
   (b) *a projection operator* $P(y) \in \mathbf{P} \equiv \{P(y) : y \in Y \cup \{\lambda\}\}$. $\mathbf{P}$ *is a set of* n*-dimensional* projection operators. λ *is the null symbol and* $P(\lambda) = I$.

At each time step a QT reads a symbol $x \in X$ *from the input, outputs a symbol* $y \in Y$*, and updates its state vector.*

The operation of a QT is described by the evolution of a bra (row) vector. We make this choice, which is unconventional in quantum mechanics, for two reasons. First, the state of a classical finite-state machine is described via a row vector. And second, the graphical meaning of a transition from state i to j is reflected in the transition matrix entries T_{ij}, only if one uses row vectors and left multiplication with T. Our previous discussion of state leads to the following definition of a QT's internal states and state vector.

Definition. *One associates an* internal state $q_i \in Q$ *with a basis vector* $\langle\phi_i|$ *such that:*

1. *For each* $q_i \in Q$ *there is a basis vector* $\langle\phi_i| = (0, \ldots, 1, \ldots, 0)$ *with a* 1 *in the* i^{th} *component.*
2. *The set* $\{\langle\phi_i| : i = 0, 1, \ldots, n-1\}$ *spans the Hilbert space* $\mathcal{H}$.

The projection operators are familiar from quantum mechanics and can be defined in terms of the internal states as follows.

Definition. *A* projection operator $P(y)$ *is the linear operator*

$$P(y) = |\phi_i\rangle \langle\phi_i| \ , \tag{27}$$

where ϕ_i *is the eigenvector of the observable with eigenvalue* y*. In the case of degeneracy* $P(y)$ *sums over a complete set of mutually orthogonal eigenstates:*

$$P(y) = \sum_i |\phi_i\rangle \langle\phi_i| \ . \tag{28}$$

Each P *is Hermitian* ($P^\dagger = P$) *and idempotent* ($P^2 = P$).

In the eigenbasis of a particular observable the corresponding matrices only have 0 and 1 entries. In the following we assume such a basis. The special case of $P(\lambda) = I$, where I is the identity matrix, is regarded as separate. Since λ is a place holder for "no output", $P(\lambda)$ is not included in the calculation of word probabilities, for example.

The identification of internal and basis states connects the machine view of a quantum system with a vocabulary that is familiar from standard developments of quantum mechanics. The mathematical representation of a QT state is given by its current state vector. At each time step a symbol is read in, which selects a unitary operator. The operator is applied to the state vector and the latter is measured. The result, an eigenvalue of the observable, is output as a symbol. In the case of no measurement, the null symbol λ is output. This is different from a nonselective measurement where a projection takes place but the outcome is not detected. The decision whether to perform a measurement or not should be considered as an input to the QT.

Each projection operator projects the state vector onto one eigenstate of the observable and the corresponding

eigenvalue is the observed quantity—the outcome of the measurement. The probability of a particular measurement outcome can be calculated from the projected state vector before renormalization. We base our analysis on the class of projective measurements, applicable to closed quantum systems [52]. (Open systems will be considered elsewhere.)

In quantum mechanics, one distinguishes between *complete* and *incomplete* measurements [39]. A *complete measurement* projects onto a one-dimensional subspace of $\mathcal{H}$. That is, the operators in a set of complete measurements all have distinct eigenvalues. In contrast, the operators associated with an *incomplete measurement* have degenerate eigenvalues. Such an operator has an effective dimension greater than 1 and projects onto a higher-dimensional subspace of $\mathcal{H}$. After such a measurement the QT is potentially in a superposition of states $\sum_i c_i \langle\phi_i|$, where i sums over a complete set of mutually orthogonal eigenstates. Just as degeneracy leads to interesting consequences in quantum physics, we will see in the examples to follow that degenerate eigenvalues lead to interesting quantum languages.

A. Word distributions

We can now describe a QT's operation as it scans its input. Starting in state $\langle\psi^0|$ it reads in a symbol $x \in X$ from an input word and updates its state by applying the unitary matrix $U(x)$. Then the state vector is projected with $P(y)$ and renormalized. Finally, symbol $y \in Y$ is emitted. That is, a single time-step of a QT is given by:

$$\begin{aligned}\langle\psi(y|x)| &= \frac{\langle\psi^0|\, T(y|x)}{\sqrt{\langle\psi^0|T(y|x)T^\dagger(y|x)|\psi^0\rangle}} \\ &= \frac{\langle\psi^0|\, U(x)P(y)}{\sqrt{\langle\psi^0|U(x)P(y)U^\dagger(x)|\psi^0\rangle}} \ , \end{aligned} \tag{29}$$

where † is the complex transpose. In the following we drop the renormalization factor in the denominator to enhance readability. It will be mentioned explicitly when a state is not to be normalized.

When a QT reads in a length-L word $x^L \in X^L$ and outputs a length-L word $y^L \in Y^L$, the transition matrix becomes

$$T(y^L|x^L) = U(x_0)P(y_0)U(x_1)P(y_1)\cdots U(x_{L-1})P(y_{L-1}) \tag{30}$$

and the updated state vector is

$$\langle\psi(y^L|x^L)| = \langle\psi^0|\, T(y^L|x^L) \ . \tag{31}$$

The QT state after reading in symbol x and emitting symbol y is given in Eq. (29). Starting the QT in $\langle\psi^0|$ the conditional probability $\Pr(y|x)$ of the output symbol y given the input symbol x can be calculated from the state vector in Eq. (29), before renormalization.

$$\Pr(y|x) = \langle\psi(y|x)|\psi(y|x)\rangle \ . \tag{32}$$

The probability $\Pr(y^L|x^L)$ of output sequence y^L conditioned on input sequence x^L can be calculated from the corresponding state vector in Eq. (31).

$$\Pr(y^L|x^L) = \langle\psi(y^L|x^L)|\psi(y^L|x^L)\rangle \ . \tag{33}$$

B. Properties

Properties of QTs are related to a subclass of STs, those with doubly stochastic transition matrices. It is useful to recall the relationship between unitary and doubly stochastic matrices to get a more intuitive understanding of the properties of QTs.

Definition. *Given a unitary matrix U, matrix M with $M_{ij} = |U_{ij}|^2$ is called a* unistochastic *matrix.*

A unistochastic matrix is always doubly stochastic, which follows directly from the properties of unitary matrices. Compared to stochastic transducers, the structure of QTs is constrained through unitarity and this constraint is reflected in the architecture of the machine. We define the existence of a *path* between node i and node j based on the unistochastic matrix M as the condition $M_{ij} > 0$. An equivalent description of a quantum transducer is given by its graphical representation.

At any particular point in time the QT is in one or several internal states. During one time step the QT reads in a symbol and follows all outgoing edges from each occupied internal state labeled with the input symbol. It then chooses probabilistically an output symbol and ends in those states that are connected by an edge labeled with that symbol.

Recall the various types of graph states reviewed in Section III A, we find that only a subset of them can be found in graphs of quantum transducers. The following Proposition states that there are no transient states in the graph of a QT.

Proposition 5. *Every node i of $\mathcal{G}(QT)$, if connected to a set of nodes $j \neq i$, is a member of a strongly connected set.*

Proof. *Given that one path exists from (say) i to j, we must show that the reverse one exists, going from j to i. According to our definition of path it is sufficient to show this for the unistochastic matrix $M_{ij} = |U_{ij}|^2$. A doubly stochastic matrix can always be expressed as a linear combination of permutation matrices. Thus, any vector $(0, 0, \ldots, 1, \ldots)$ with only one 1 entry can be permuted into any other vector with only one 1 entry. This is equivalent to saying that, if there is a path from node i to j there is a path from j to i.*

Corollary 5. *The maximum size of the output alphabet Y of a QT is equal to the dimension of the Hilbert space.*

Proof. *This follows directly from the definition of QTs since the output symbols are directly associated with eigenvalues. The number of eigenvalues is bounded by the dimension of the Hilbert space.*

The discussion of unistochastic matrices leads one to conclude that QT graphs constitute a subset of directed graphs, namely the strongly connected ones. Moreover, there is a constraint on incoming edges to a node.

Proposition 6. *All incoming transitions to an internal state are labeled with the same output symbol.*

Proof. *Incoming transitions to internal state q_i are labeled with output symbol y if $\langle\phi_i|$ is an eigenstate of projection operator $P(y)$. The operators $P(y)$ are orthogonal and so no two operators project onto the same state. So the incoming transitions to any particular state q_i are labeled with the same output symbol representing one eigenvalue.*

Proposition 7. *A QT's transition matrices $T(y|x)$ uniquely determine the unitary matrices $U(x)$ and the projection operators $P(y)$.*

Proof. *Summing the $T(y|x)$ over all y for each x yields the unitary matrices $U(x)$:*

$$\sum_{y\in Y} T(y|x) = \sum_{y\in Y} U(x)P(y) = U(x) \ . \tag{34}$$

The $P(y)$ are obtained through the inverse of $U^{-1}(x) = U^{\dagger}(x)$:

$$P(y) = U^{\dagger}(x)T(y|x) \ . \tag{35}$$

Definition. *A QT is* reversible *if the automaton defined by the transpose of each $U(x)$ and $P(y)$ is also a QT.*

Since unitary matrices always have an inverse, given by their complex conjugate transpose, any (unmeasured) state evolution is reversible. This leads to the result that QTs are always reversible.

Proposition 8. *All QTs are reversible.*

Proof. *The transpose of a unitary matrix is unitary. The transpose of a projection operator is the operator itself.*

Graphically, the reversed QT is obtained by simply switching the direction of the edges. This produces a transducer with the transition amplitudes T_{ji}, formerly T_{ij}. The original input and output symbols, which labeled ingoing edges to state q_i, remain unchanged. Therefore, in general, the languages generated by a QT and its inverse are not the same. Notably, this simple operation applied to an ST does not, in general, yield another ST.

VII. QUANTUM RECOGNIZERS AND GENERATORS

A quantum transducer is the most general object, describing a quantum dynamical process in terms of inputs and outputs. We will now specialize a quantum transducer into recognizers and generators. We do this by following the strategy we adopted for developing classes of stochastic transducers. For each machine class we first give a general definition and then specialize, yielding fully deterministic versions. We establish a number of properties for each type and then compare their descriptive powers. The comparison is done in terms of the process languages each class can recognize or generate. The results are collected together in a computational hierarchy of finitary quantum processes.

A. Quantum Recognizers

Quantum finite-state machines are almost exclusively discussed as recognizing devices. Following our development of a consistent set of quantum finite-state transducers, we can now introduce quantum finite-state recognizers as restrictions of QTs and compare these with alternative models of quantum recognizers. Since we are interested in the recognition of process languages our definition of quantum recognizers differs from those introduced elsewhere, see Sec. VII B below. The main difference is the recognition of a process language including its word distribution. The restrictions that will be imposed on a QT to achieve this are similar to those of the stochastic recognizer.

Definition. *A* quantum finite-state recognizer *(QR) is a QT with $U(x) = U, \forall x \in X$, and $Y = \{accept, reject\}$.*

The definition of a QR is such that the conditions for recognizing process languages in Eq. (7) are fulfilled. In practice, recognition works similar to classical recognition. The experimenter runs an ensemble of QRs on the same input. The frequency of acceptance can then be compared to the probability of the input string computed from the $T(x)$.

Definition. *A QR* accepts *a process language $\mathcal{P}$ with word-probability threshold $0 \leq \delta \leq 1$, if for all $w \in \mathcal{P}$}

$$\left|\Pr(w) - \langle\psi^0|T(w)T^{\dagger}(w)|\psi^0\rangle\right| \leq \delta \ . \tag{36}$$

Acceptance or rejection happens at each time step. It is worth pointing out that the reject state that we encountered in stochastic recognizers as a sink of probability is now part of the recurrent part of the QR's graph. (This follows from the graph properties discussed above in Sec. VI B.)

We also have deterministic versions of QRs.

Definition. *A* quantum deterministic finite-state recognizer *(QDR) is a quantum recognizer with transition matrices $T(x)$ that have at most one nonzero element per row.*

B. Alternative Quantum Recognizers

Quantum finite automata were introduced by several authors in different ways, and they recognize different classes of languages. To our knowledge the first mention of *quantum automata* was made by Albert in 1983 [40]. Albert's results have been subsequently criticized by Peres as being based on an inadequate notion of measurement [41].

Kondacs and Watrous introduced 1-way and 2-way quantum finite-state automata [42]. The 1-way automata read symbols once and from left to right (say) in the input word. Their 2-way automata scan the input word many times moving either left to right or right to left. The automata allow for measurements at every time step, checking for acceptance, rejection, or continuation. They show that a 2-way QFA can recognize all regular languages and some nonregular languages. 1-way QFA are less powerful: They can only recognize a subset of the regular languages. A more powerful generalization of a 1-way QFA is a 1-way QFA that allows mixed states, introduced by Aharonov et al. [43]. They also allow for nonunitary evolution. Introducing the concept of mixed states simply adds classical probabilities to quantum probabilities and is inherent in our model of QTs.

The distinctions between these results and the QRs introduced here largely follow from the difference between regular languages and process languages. Thus, the result in Ref. [42], that no 1-way quantum automaton can recognize the language $\{0,1\}^*0$, does not apply to QTs. It clearly is a regular language, but not a process language. Also, the result by Bertoni and Carpentieri, that quantum automata can recognize non-regular languages, does not apply here [44]. A quantum automaton that is not measured at each time step and, in addition, does not underlie normalization constraints can recognize a nonregular language. Measuring the whole system, not only parts of it, causes information stored in the quantum phase to be immediately destroyed. On the other hand, in order to represent a process language, the measurement operators must be constant over time. We return to this point in Theorem 4 below.

Interestingly, the number of states of a quantum automaton is of the order of 2^n, when a classical deterministic automaton needs only n states [45]. Moore and one of the authors introduced a 1-way quantum automaton (without using the term "1-way") [46]. It is less powerful than the 1-way automaton by Kondacs and Watrous, since it allows only for a single measurement, after the input has been read in. They also introduced a generalized quantum finite-state automaton whose transition matrices need not be unitary, in which case all regular languages are recognized. Freivalds and Winter introduced quantum transducers, mentioned earlier [38]. The model, however, lacks a direct physical interpretation.

These alternative models for quantum automata appear to be the most widely discussed. There are others, however, and so the above list is by no means complete. Our motivation to add yet another model of quantum finite-state recognizer to this list is the inadequacy of the alternatives to recognize process languages—languages that represent quantum dynamical systems subject to repeated measurement.

C. Quantum Generators

We now introduce quantum finite-state generators as restrictions of QTs and as a complement to recognizers. They serve as a representation for the behavior of autonomous quantum dynamical systems. In contrast to quantum finite-state recognizers, quantum finite-state generators appear to not have been discussed before. A quantum generator is a QT with only one input. As in the classical case, one can think of the input as a clock signal that drives the machine through its transitions.

Definition. *A* quantum finite-state generator *(QG) is a QT with* $|X| = 1$.

At each step it makes a transition from one state to another and emits a symbol. As in the classical case there are nondeterministic (just implicitly defined) and deterministic QGs.

Definition. *A* quantum deterministic finite-state generator *(QDG) is a QT in which each matrix* $T(y)$ *has at most one nonzero entry per row.*

Interestingly, there is a mapping from a given QDG to a classical automaton.

Definition. *Given a QDG* $\mathcal{M} = \{U, P(y)\}$*, the* equivalent *(classical) SDG* $\mathcal{M}' = \{T(y)\}$ *has unistochastic state-to-state transition matrix* T *with components* $T_{ij} = [U_{ij}]^2$.

We leave the technical interpretation of "equivalence" to Thm. 3 below.

As mentioned earlier, in quantum mechanics one distinguishes between complete and incomplete measurements. Having introduced the different types of quantum generators, we can now make a connection to complete measurements.

Definition. *A* quantum complete finite-state generator *(QCG) is a QG observed via complete measurements.*

D. Density Matrix Formalism

Coming back to the various notions of "state" discussed in Section V A we now extend the formalism of quantum automata to distributions over states, so called *mixed states*. This enables us to average over observations. Let a system be described by a state vector $\langle\psi_i|$ at time t. If we do not know the exact form of $\langle\psi_i|$ but only a set of possible $\langle\psi_i|$, then we give the best guess

as to the system's state in terms of a statistical mixture of the $\langle\psi_i|$. This statistical mixture is represented by a *density operator* ρ with weights p_i assigned to the $\langle\psi_i|$:

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{37}$$

The main difference from common usage of "mixed state" is that we compare the same state *over time*; whereas, typically different *systems* are compared at a single time. Nevertheless, in both cases, the density matrix formalism applies.

With this notation in hand, we can now establish a number of properties of the quantum machines.

Definition. *The* stationary state ρ^s *of a QR or QG is the mixed state which is invariant under unitary evolution:*

$$\rho^s = \sum_{\sigma\in\Sigma} P(\sigma)U^\dagger \rho^s U P(\sigma) \ . \tag{38}$$

Theorem 1. *The stationary state of a QDR or QDG is the following maximally mixed state:*

$$\rho^s = |Q|^{-1} \sum_{i=0}^{n-1} \left|\phi_i\right\rangle \left\langle\phi_i\right| \tag{39}$$

$$= |Q|^{-1} \mathbb{1} \ . \tag{40}$$

Since the $\langle\phi_i|$ are basis states, ρ^s is a diagonal matrix equal to the identity multiplied by a factor.

Proof.

$$\rho^s = \sum_{\sigma\in\Sigma} P(\sigma)U^\dagger \rho^s U P(\sigma) \tag{41}$$

$$= |Q|^{-1} \sum_{\sigma\in\Sigma} P(\sigma)U^\dagger U P(\sigma) \tag{42}$$

$$= |Q|^{-1} \sum_{\sigma\in\Sigma} P(\sigma) \tag{43}$$

$$= |Q|^{-1} \mathbb{1} \tag{44}$$

Recall that the stationary distribution of a Markov chain with doubly stochastic transition matrix is always uniform [47].

Having established the concept of *stationary state* we can now use it to give asymptotic symbol probabilities conditioned on the stationary state ρ^s. We find:

$$\begin{aligned} \Pr(\sigma) &= \mathrm{tr}\left(T^\dagger(\sigma)\rho^s T(\sigma)\right) \\ &= \mathrm{tr}\left(P^\dagger(\sigma)U^\dagger \rho^s U P(\sigma)\right) \\ &= \mathrm{tr}\left(U^\dagger \rho^s U P(\sigma)\right) \ , \end{aligned} \tag{45}$$

where tr is the *trace operator*. Similarly, the asymptotic word probabilities $\Pr(\sigma^L)$ are:

$$\Pr(\sigma^L) = \mathrm{tr}\left(T^\dagger(\sigma^L)\rho^s T(\sigma^L)\right) \ . \tag{46}$$

No further simplification is possible for the general case.

Equation (45), however, can be further simplified for single-symbol probabilities. As a result we find a concise expression for single-symbol probabilities of QGs and QRs.

Theorem 2. *The symbol distribution generated by a QG or recognized by a QR only depends on the dimension of the projection operators and the dimension $|Q|$ of the Hilbert space.*

Proof. *Eq. (45) simplifies as follows:*

$$\begin{aligned} \Pr(\sigma) &= \mathrm{tr}(\rho^s P(\sigma)) \\ &= |Q|^{-1} \dim P(\sigma) \ . \end{aligned} \tag{47}$$

Although the single-symbol distribution is determined by the dimension of the subspaces onto which the $P(\sigma)$ project, distributions of words σ^L with $L > 1$ are not similarly restricted.

E. Hierarchy of Finitary Process Languages

To better appreciate what these machines are capable of we amortize the effort in developing the preceding results to describe the similarities and differences between quantum recognizers and generators, as well as between stochastic and quantum automata. We collect the results, give a summary and some interpretation, and present a road map (Fig. 5) that lays out the computational hierarchy of finitary quantum processes. As above, when we refer to $\mathcal{P}(M)$ we mean the language produced by a machine in class M.

Proposition 9. *QCGs are deterministic.*

Proof. *Since all projection operators have dimension one, all transition matrices have at most one nonzero element per row. This is the condition for being a QDG.*

Complete measurements always define a QDG. There are incomplete measurements, however, that also can lead to QDGs, as we will show shortly. One concludes that $\mathcal{P}(QCG) \subset \mathcal{P}(QDG)$.

We now show that for any QDG there is an SDG generating the same process language. Thereby we establish *observational* equivalence between the different classes of machine.

Theorem 3. *Every $\mathcal{P}(QDG)$ is generated by some SDG: $\mathcal{P}(QDG) \subseteq \mathcal{P}(SDG)$.*

Proof. *We show that the SDG generating $\mathcal{P}(QDG)$ is the* equivalent *SDG, as defined in Sec. VII C, and that the QDG $\mathcal{M}$ and its* equivalent *SDG $\mathcal{M}'$ generate the same word distribution and so the same process language.*

The word probabilities $\Pr_{\mathcal{M}}(y^L)$ *for* $\mathcal{M}$ *are calculated using Eq. (46) and the QDG's transition matrices* $T_{\mathcal{M}}$*:*

$$\begin{aligned}\Pr_{\mathcal{M}}(y^L) &= \mathrm{tr}\left(T^\dagger_{\mathcal{M}}(y^L)\rho^s T_{\mathcal{M}}(y^L)\right)\\ &= |Q|^{-1}\mathrm{tr}(T^\dagger T)\\ &= |Q|^{-1}\sum_i [T^\dagger T]_{ii}\\ &= |Q|^{-1}\sum_i\sum_j T^\dagger_{ij}T_{ji}\\ &= |Q|^{-1}\sum_{ij} T^2_{ij}\ .\end{aligned}$$

The word probabilities $\Pr_{M'}(y^L)$ *for* M' *are calculated using Eq. (19) and the SDG's transition matrices* $T_{\mathcal{M}}$*:*

$$\begin{aligned}\Pr_{\mathcal{M}'}(y^L) &= \langle \pi^0 | T_{\mathcal{M}'}(y^L)|\eta\rangle\\ &= \sum_{i=0}^{n-1}\left(\pi^0_i \sum_j (T_{\mathcal{M}'}(y^L))_{ij}\right)\\ &= |S|^{-1}\sum_{i,j=0}^{n-1}(T_{\mathcal{M}'}(y^L))_{ij}\ . \qquad (48)\end{aligned}$$

Since $(T_{\mathcal{M}}(y^L))^2_{ij} = (T_{\mathcal{M}'}(y^L))_{ij}$*, from the definition of an* equivalent *SDG, the claim follows.*

A given QDG can be observationally equivalent to more than one SDG. This occurs because the phases of the transition amplitudes cancel in the transformation from a QDG. We can now easily check the languages produced by QDGs.

Corollary 6. *For every QDG,* supp $\mathcal{P}(QDG)$ *is a regular language.*

Proof. *This follows directly from Thm. 3 and Cor. 1.*

Corollary 7. *For every QDG,* $\mathcal{P}(QDG)$ *is a process language.*

Proof. *This follows directly from Thm. 3 and Cor. 2.*

With this we can begin to compare the descriptive power of the different machine types.

Proposition 10. *QGs and QRs are equivalent: They recognize and generate the same set of languages, respectively:* $\mathcal{P}(SG) = \mathcal{P}(SR)$*.*

Proof. *Consider QG's transition matrices* $T(y) = UP(y)$ *and form a new set* $T(x) = UP(x)$ *in which* $P(accept|x) = P(y)$*. The* $T(accept|x)$ *together with* $T(reject|x)$*, where* $T(reject|x) = UP(reject|x) = \mathbb{1} - P(accept|x)$*, define a QR that recognizes* $\mathcal{P}(QG)$*. It follows that* $\mathcal{P}(QG) \subseteq \mathcal{P}(QR)$*.*

Now consider QR's transition matrices $T(accept|x) = UP(accept|x)$ *and form a new set* $T(y)$ *in which* $P(y) = P(accept|x)$*. The* $T(y)$ *define a QG that generates* $\mathcal{P}(QR)$*. It follows that* $\mathcal{P}(QG) = \mathcal{P}(QR)$*.*

Corollary 8. *QDGs and QDRs are equivalent: They recognize and generate the same set of languages, respectively:* $\mathcal{P}(QDG) = \mathcal{P}(QDR)$*.*

Proof. *This follows directly from Prop. 10 and the fact that QDRs and QDGs are special cases of QRs and QGs, respectively.*

Corollary 9. *For every QDR,* supp $\mathcal{P}(QDR)$ *is a regular language.*

Proof. *This follows directly from Cor. 8 and Cor. 1.*

Corollary 10. *For every QDR,* $\mathcal{P}(QDR)$ *is a process language.*

Proof. *This follows directly from Cor. 8 and Cor. 7.*

Theorem 4. *For every QG,* supp $\mathcal{P}(QG)$ *is a regular language.*

Proof. *We show this by making the argument that any quantum-like quantity such as superposition of states or relative phases only affects a measurement once and not repeatedly. It is "used up" in one measurement and can not serve as a permanent stack.*

Classically, what is needed for generating a nonregular language is a stack, which is an auxiliary storage device. Quantum mechanically, a relative phase can serve as stack, creating a state such as $\langle\psi| = \frac{1}{\sqrt{2}}(\langle\phi_A| + e^{i\theta}\langle\phi_B|)$*. A projective measurement turns any relative phase between states of two different subspaces into an absolute phase, such as* $e^{i\theta}\langle\phi_B|$ *which is unmeasurable. Any phase between states in the same subspace has no measurable effect since the eigenvalues of those states are degenerate.*

Thus, a quantum stack can only be probed once, after which it is emptied.

Proposition 11. *There exists an SDG such that* $\mathcal{P}(SDG)$ *is not generated by any QDG.*

Proof. *The process language generated by the SDG given by*

$$T(0) = \left(\tfrac{1}{\sqrt{2}}\right) \text{ and } T(1) = \left(1 - \tfrac{1}{\sqrt{2}}\right) \qquad (49)$$

(a biased coin) cannot be generated by any QDG. According to Thm. 2,

$$\Pr(y) = \frac{\dim P(y)}{n}\ , \qquad (50)$$

which is a rational number, whereas $\Pr(y)$ *for the above biased coin is irrational.*

Corollary 11. $\mathcal{P}(QDG) \subset \mathcal{P}(SDG)$*.*

Proof. *From Thm. 3 and Prop. 11.*

Corollary 12. $\mathcal{P}(QDR) \subset \mathcal{P}(SDR)$*:*

Proof. *From Cor. 8, Cor. 3, Thm. 3, and Prop. 11.*

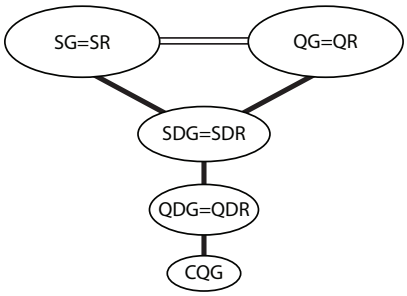

FIG. 5: Finitary process language hierarchy: Each circle represents the set of process languages recognized or generated by the inscribed machine class. Increasing height indicates proper containment; machine classes at the same height are not linearly comparable. The hierarchy summarizes the theorems, propositions, and corollaries in Secs. IV C and VII E.

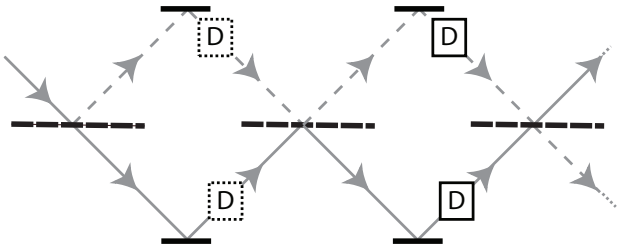

FIG. 6: Experimental set-ups for the iterated beam splitter: Solid lines are mirrors; beam splitters, horizontal dashed lines. Photon nondemolition detectors, marked as D, are placed between every pair of beam splitters. Under measurement protocol I all detectors are in operation; under protocol II only the solid-line detectors are activated. The apparatus is repeated indefinitely to the right.

At this point it is instructive to graphically summarize the relations between recognizer and generator classes. Figure 5 shows a machine hierarchy in terms of languages recognized or generated. The class of QCGs is at the lowest level. This is contained in the class of QDGs and QDRs. The languages they generate or recognize are properly included in the set of languages generated or recognized by classical deterministic machines—SDGs and SDRs. These, in turn, are included in the set of languages recognized or generated by classical nondeterministic machines, SGs and SRs, as well as QRs and QGs.

Finally, an important comparison at a lower level remains open.

Conjecture 1. *There exists a QG such that $\mathcal{P}(QG)$ is not recognized by any QDR.*

The preceding results and conjecture serve to indicate how the finitary process hierarchy is organized. For example, analyzing how varying the acceptance threshold δ modifies the hierarchy awaits further investigation.

VIII. QUANTUM GENERATORS AND FINITARY PROCESSES: EXAMPLES

It will be helpful at this point to illustrate various features of QGs by modeling example quantum processes. We start out with deterministic QGs before we arrive at the last example which illustrates a (nondeterministic) quantum transducer (i.e., a QT) with input and output.

A. Two-State Quantum Processes

According to Thm. 2 the symbol distribution generated by a QG only depends on the dimension of the projection operator and the dimension of the Hilbert space. What are the consequences for two-state QGs? First of all, according to Cor. 5 the maximum alphabet size is 2. The corresponding projection operators can either have dimension 2 (for a single-letter alphabet) or dimension 1 for a binary alphabet. The only symbol probabilities possible are $\Pr(y) = 1$ for the single-letter alphabet and $\Pr(y) = 1/2$ for a binary alphabet. So we can set aside the single-letter alphabet case as a bit too simple.

At this point, we see that a binary-alphabet QDG can produce only a highly restricted set of process languages. It is illustrative to look at the *equivalent* SDG. Its state-to-state transition matrix is given by

$$T = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix} . \tag{51}$$

For $p = 0.5$, for example, this is the fair coin process. It becomes immediately clear that the Golden Mean and the Even processes, which are modeled by two-state classical automata, cannot be represented with a two-state QDG. (The three-state models are given below.)

1. Iterated Beam Splitter

We now turn to a physical two-state process and build a quantum generator for it.

The *iterated beam splitter* is an example that, despite its simplicity, makes a close connection with real experiment. Figure 6 shows the experimental apparatus. Photons are sent through a beam splitter (thick dashed line), producing two possible paths. The paths are redirected by mirrors (thick horizontal solid lines) and recombined at a second beam-splitter. From this point on the same apparatus is repeated indefinitely to the right. After the second beam-splitter there is a third and a fourth and so on. Single-photon quantum nondemolition detectors are located along the paths, between every pair of beam-splitters. One measures if the photon travels in the upper path and another determines if the photon follows the lower path. In practice one detector would be sufficient.

This is a quantum dynamical system: a photon passing repeatedly through various beam splitters. It has a two-dimensional state space with two eigenstates—"above" and "below". Its behavior is given by the evolution of the state vector $\langle\psi|$. The overall process can be represented in terms of a unitary operation for the beam splitter and projection operators for the detectors. The unitary operator for the beam splitter is the Hadamard matrix U_H:

$$U_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} . \tag{52}$$

The measurement operators have the following matrix representation in the experiment's eigenbasis:

$$P(0) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} , \tag{53}$$

where the measurement symbol 0 stands for "above" and symbol 1 stands for "below".

Before we turn to constructing a quantum finite-state generator to model this experiment we can understand intuitively the measurement sequences that result from running the experiment for long times. If entering the beam splitter from above, the detectors record the photon in the upper or lower path with equal probability. Once the photon is measured, though, it is in that detector's path with probability 1. And so it enters the beam splitter again via only one of the two possible paths. Thus, the second measurement outcome will have the same uncertainty as the first: the detectors report "above" or "below" with equal probability. The resulting sequence of measurements after many beam splitter passages is simply a random sequence. Call this measurement protocol I.

Now consider altering the experiment slightly by removing the detectors after every other beam splitter. In this configuration, call it protocol II, the photon enters the first beam splitter, does not pass a detector and interferes with itself at the next beam splitter. That interference, as we will confirm shortly, leads to destructive interference of one path after the beam splitter. The photon is thus in the same path after the second beam splitter as it was before the first beam splitter. A detector placed after the second beam splitter therefore reports with probability 1 that the photon is in the upper path, if the photon was initially in the upper path. If it was initially in the lower path, then the detector reports that it is in the upper path with probability 0. The resulting sequence of upper-path detections is a very predictable sequence, compared to the random sequence from protocol I.

We now construct a QDG for the iterated-beam splitter using the matrices of Eqs. (52)-(53) and the stationary state of Eq. (40). The output alphabet consists of two symbols denoting detection "above" or "below": $Y = \{0, 1\}$. The set of states consists of the two eigenstates of the system "above" and "below": $Q = \{A, B\}$. The

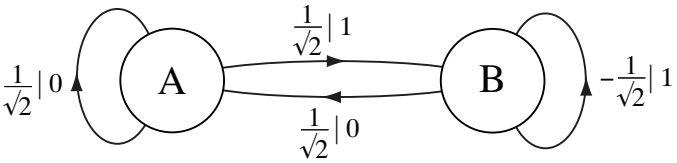

FIG. 7: Quantum finite-state machine for the iterated beam splitter: The resulting symbol sequences are statistically identical to the measurement sequences obtained with the measurement protocols I and II shown in Fig. 6. When no measurement is made, transitions along all edges occur.

transition matrices are:

$$T(0) = U_H P(0) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} , \tag{54a}$$

$$T(1) = U_H P(1) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} . \tag{54b}$$

The resulting QDG is shown in Fig. 7.

The word distribution for the process languages generated by protocols I and II are obtained from Eq. (46). Word probabilities for protocol I (measurement at each time step) are, to give some examples:

$$\Pr(0) = |Q|^{-1}\dim(P(0)) = \frac{1}{2} , \tag{55a}$$

$$\Pr(1) = |Q|^{-1}\dim(P(1)) = \frac{1}{2} , \tag{55b}$$

$$\Pr(00) = \mathrm{tr}\left(T^\dagger(0)T^\dagger(0)\rho^s T(0)T(0)\right) = \frac{1}{4} , \tag{55c}$$

$$\Pr(01) = \Pr(10) = \Pr(11) = \frac{1}{4} . \tag{55d}$$

Continuing the calculation for longer words shows that the word distribution is uniform at all lengths $\Pr(y^L) = 2^{-L}$.

For protocol II (measurement every other time step) we find:

$$\Pr(0) = \mathrm{tr}\left(T^\dagger(0\lambda)\rho^s T(\lambda 0)\right) = \frac{1}{2} , \tag{56a}$$

$$\Pr(1) = \mathrm{tr}\left(T^\dagger(1\lambda)\rho^s T(\lambda 1)\right) = \frac{1}{2} , \tag{56b}$$

$$\Pr(00) = \mathrm{tr}\left(T^\dagger(0\lambda 0\lambda)\rho^s T(\lambda 0\lambda 0)\right) = \frac{1}{2} , \tag{56c}$$

$$\Pr(11) = \mathrm{tr}\left(T^\dagger(1\lambda 1\lambda)\rho^s T(\lambda 1\lambda 1)\right) = \frac{1}{2} , \tag{56d}$$

$$\Pr(10) = \Pr(01) = 0 . \tag{56e}$$

If we explicitly denote the output at the unmeasured time step as λ, the sequence 11 turns into $\lambda 1\lambda 1$, as do the other sequences in protocol II. As one can see, the word probabilities calculated from the QDG agree with our earlier intuitive conclusions.

Comparing the iterated beam splitter QDG to its classically equivalent SDG reveals several crucial differences

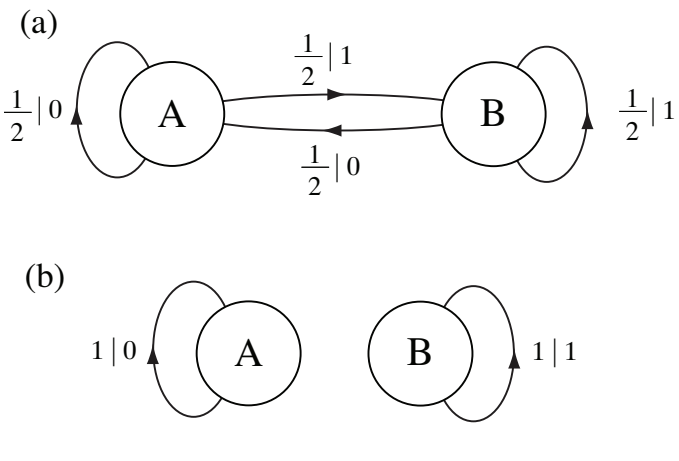

FIG. 8: Classical deterministic generators for the iterated beam splitter: (a) Protocol I and (b) protocol II, $p = 2$. (Cf. Fig. 6.)

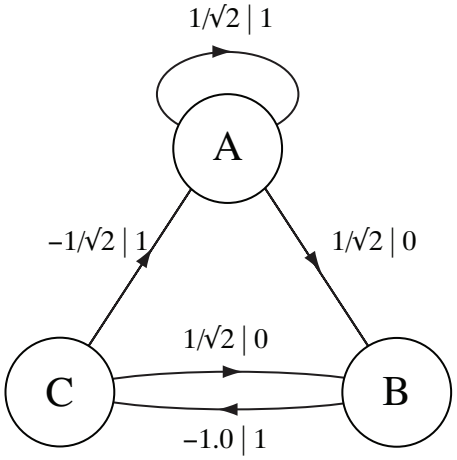

FIG. 9: Quantum generator for the Golden Mean Process.

in performance. Following the recipe from Sec. VII E, on how to build an SDG from a QDG, gives the classical generator shown in Fig. 8(a). Its transition matrices are:

$$T(0) = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \text{ and } T(1) = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} . \quad (57)$$

The measurement sequence generated by this SDG for protocol I is the uniform distribution for all lengths, as can be easily verified using Eq. (19) or, since it is deterministic, Eq. (20). This is equivalent to the language generated by the QDG. However, the probability distribution of the sequences for the generator under protocol II, ignoring every second output symbol, is still the uniform distribution for all lengths L. This could not be more different from the language generated by the QDG in protocol II.

The reason is that the classical machine is unable to capture the interference effects present in experimental set-up II. A second SDG has to be constructed from the QDG's transition matrices for set-up II. This is done by carrying out the matrix product first and then forming its equivalent SDG. The result is shown Fig. 8(b). Its transition matrices are:

$$T(0) = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } T(1) = \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} . \quad (58)$$

The two classical SDGs are clearly (and necessarily) different. Thus, a single QG can model a quantum system's dynamics for different measurement periods. Whereas an SG only captures the behavior of each individual experimental set-up. This illustrates the utility of QGs over SGs in modeling the behavior of quantum dynamical systems.

B. Three-State Quantum Processes

1. Golden Mean Quantum Machine

Recall the classical Golden Mean generator of Section IV B. A QDG, which generates the same process language, is shown in Fig. 9. Consider a spin-1 particle subject to a magnetic field that rotates its spin. The state evolution can be described by the unitary matrix

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & -1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} , \quad (59)$$

which is a rotation in $\mathbb{R}^3$ around the y-axis by angle $\frac{\pi}{4}$ followed by a rotation around the x-axis by $\frac{\pi}{2}$.

Using a suitable representation of the spin operators J_i [48, p.199], such as: $J_x = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & i \\ 0 & -i & 0 \end{smallmatrix}\right)$, $J_y = \left(\begin{smallmatrix} 0 & 0 & i \\ 0 & 0 & 0 \\ -i & 0 & 0 \end{smallmatrix}\right)$, and $J_z = \left(\begin{smallmatrix} 0 & i & 0 \\ -i & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)$, the relation $P_i = 1 - J_i^2$ defines a one-to-one correspondence between the projector P_i and the square of the spin component along the i-axis. The resulting measurement poses the yes-no question, Is the square of the spin component along the i-axis zero? Consider measuring J_y^2. Then U and the projection operators $P(0) = |100\rangle\langle 100| + |001\rangle\langle 001|$ and $P(1) = |010\rangle\langle 010|$ define a quantum generator.

The transition matrices $T(y)$ are then

$$T(0) = UP(0) = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} , \quad (60a)$$

$$T(1) = UP(1) = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & -1 \\ -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix} . \quad (60b)$$

To illustrate that this QDG produces the Golden Mean word distribution we show how to calculate several of the

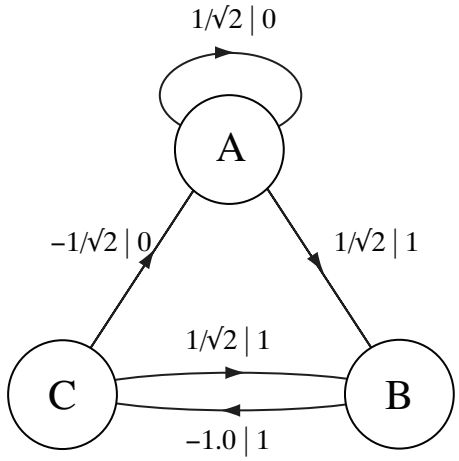

FIG. 10: Quantum generator for the Even Process.

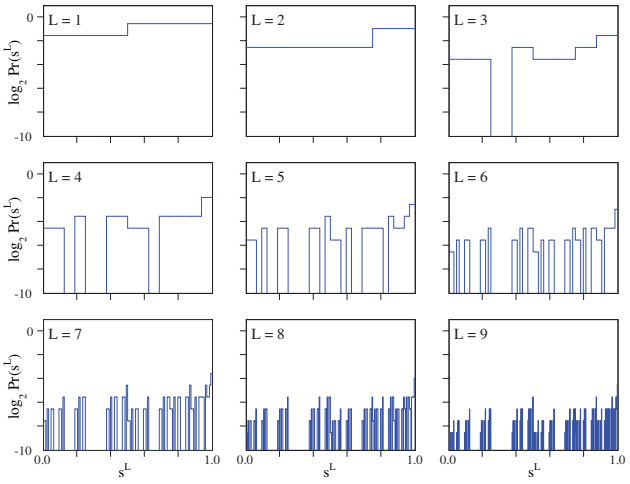

FIG. 11: Process language of the Even QDG.

word probabilities using Thm. 2 and Eq. (46):

$$\Pr(0) = |Q|^{-1}\dim(P(0)) = \frac{1}{3}\ , \tag{61a}$$

$$\Pr(1) = |Q|^{-1}\dim(P(1)) = \frac{2}{3}\ ,$$

$$\Pr(011) = \mathrm{tr}\left(T^{\dagger}(011)\rho^{s}T(011)\right) = \frac{1}{6}. \tag{61b}$$

2. *Quantum Even Process*

The next example is a quantum representation of the Even Process. Consider the same spin-1 particle. This time the J_y^2 component is chosen as observable. Then U and $P(0) = |100\rangle\langle 100|$ and $P(1) = |011\rangle\langle 011|$ define a quantum finite-state generator. The QDG is shown in Fig. 10. The word distributions for lengths up to $L = 9$ are shown in Fig. 11.

Note that the unitary evolution for the Golden Mean Process and the Even Process are the same, just as the state-to-state transition matrices were the same for their classical versions. The partitioning into subspaces induced by the projection operators leads to the (substantial) differences in the word distributions.

The dependence on subspace partitioning indicates a way to count the number of QGs for each unitary evolution U. For 3-dimensional Hilbert spaces this is rather straightforward. For each unitary matrix and with a binary alphabet we have three choices for partitioning subspaces of the Hilbert space: one subspace is two-dimensional and the others one-dimensional. This yields three QGs that are distinct up to symbol exchange ($0 \leftrightarrow 1$). For the unitary matrix that generates the Golden Mean and the Even Process (Eq. (59)). The third QG turns out to be nondeterministic. But no phase interference is possible and it generates the Golden Mean process language. The sofic nature of these quantum processes has been discussed in Ref. [19].

This very limited number of possible QGs for any given unitary matrix is yet another indication of the limitations of QGs. Classical SGs do not have the same structural restrictions, since they are not bound by orthogonal partitioning into subspaces, for example. The saving grace for QGs is that they have complex transition amplitudes and so can compute with phase, as long as they are not observed. This is reflected in the distinct languages generated by one QG under different measurement protocols.

C. Four-State Quantum Process

We are now in the position to explore the full capabilities of QTs, turning from generators to transducers. The following example illustrates quantum machines using the tools required to investigate information processing of quantum dynamical systems.

1. *Quantum Transducer For Trapped Ions*

Consider an atom exposed to short wavelength radiation—the core of numerous experiments that investigate electronic structure and dynamics. The usual protocol is a one-time experiment, exposing the atom to radiation and monitoring changes in structure through electron or photon detectors. As a particular set-up we choose ion-trap experiments found in low-temperature physics and quantum computation implementations, as described in Ref. [7]. For our present purposes it will be sufficient to review the general physical setting.

Imagine a pair of ions kept in a trap by laser fields and static electromagnetic fields. Only two of the electronic levels of each ion are of interest: the ground state and an excited state. Call these level 0 and level 1, respectively. A third auxiliary level is required for laser cooling and

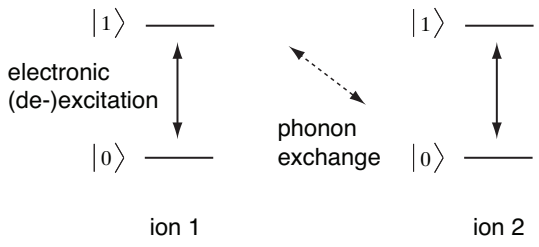

FIG. 12: Schematic view of two vibrationally-coupled trapped ions undergoing electronic excitation. Only the two electronic levels of interest are drawn.

other operations, which we leave aside here since it has no significance for the description of the process. The two ions are coupled to each other through phonon exchange, as shown schematically in Fig. 12.

By choosing suitable wavelengths several distinct operators can be implemented. One of them is a Hadamard operator that produces a superposition of electronic states $|0\rangle$ and $|1\rangle$. Another is a phase operator that yields an entangled state of the two ions. The respective laser pulses, so-called *Rabi* pulses, induce an electronic excitation and a vibrational excitation. The result is vibrational coupling of the four levels. All other operations are combinations of these two; see Ref. [7]. The operators are named U_a, U_b, and U_c; matrix representations are given below. As is already familiar from the iterated beam splitter, the operators are activated repeatedly one after the other in a closed loop.

To model the quantum dynamical system the state vector and operator matrices need to be specified. The four basis states spanning the Hilbert space are given by [53]:

$$\begin{aligned}\langle\phi_A| &= \langle 00| \ , \\ \langle\phi_B| &= \langle 01| \ , \\ \langle\phi_C| &= \langle 10| \ , \\ \langle\phi_D| &= \langle 11| \ .\end{aligned}$$

The three unitary operations in matrix form are:

$$U_a = H \otimes H = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \ , \tag{62a}$$

$$U_b = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \ , \tag{62b}$$

$$U_c = H \otimes I = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \ . \tag{62c}$$

The projection operators are chosen to measure the electronic state of ion 1 only and have the matrix form:

$$P(0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \ . \tag{63}$$

The QT is now easily assembled. The set of states and the input and output alphabets are, respectively:

$$\begin{aligned} Q &= \{A, B, C, D\} \ , \\ X &= \{a, b, c\} \ , \text{ and} \\ Y &= \{0, 1\} \ . \end{aligned}$$

This QT's graph is shown in Fig. 13.

To illustrate its operation we consider two measurement protocols. For each we use input sequence $(abc)^+$.

- Measurement protocol I: Measure ion 1 after each unitary operation. The resulting state vector evolution is:

$$\langle\psi_{t+1}| = \langle\psi_t| U_a P(y) \ , \tag{64a}$$
$$\langle\psi_{t+2}| = \langle\psi_{t+1}| U_b P(y) \ , \tag{64b}$$
$$\langle\psi_{t+3}| = \langle\psi_{t+2}| U_c P(y) \ . \tag{64c}$$

- Measurement protocol II: Measure ion 1 only after three unitary operations. This leads to evolution according to

$$\langle\psi_{t+3}| = \langle\psi_t| U_a U_b U_c P(y) \ . \tag{65}$$

The probability distributions of the observed sequences are shown in Figs. 14 and 15. The two distributions differ substantially. On the one hand, protocol II simply yields the process language of alternating 0s and 1s. Protocol I, on the other hand, yields a much larger set of allowed words. In particular, it is striking that supp $\mathcal{P}^{\text{II}}$ is forbidden behavior under protocol I. The words 0101 and 1010 are forbidden under protocol I, whereas they are the only allowed words of length $L = 4$ under protocol II.

Not only does this example illustrate that a simple change in measurement protocol leads to a substantial change in the observed dynamics. It is also not clear a priori when a more complicated behavior is to be expected. That is, more frequent measurement yields more complicated behavior. Without quantifying how complex that complicated behavior is, it turns out that it is not always the longer period of coherent, unperturbed unitary evolution that yields more complex processes. This will have consequences for feasible implementations of quantum computational algorithms. For a quantitative discussion of the languages generated by quantum processes see Ref. [24].

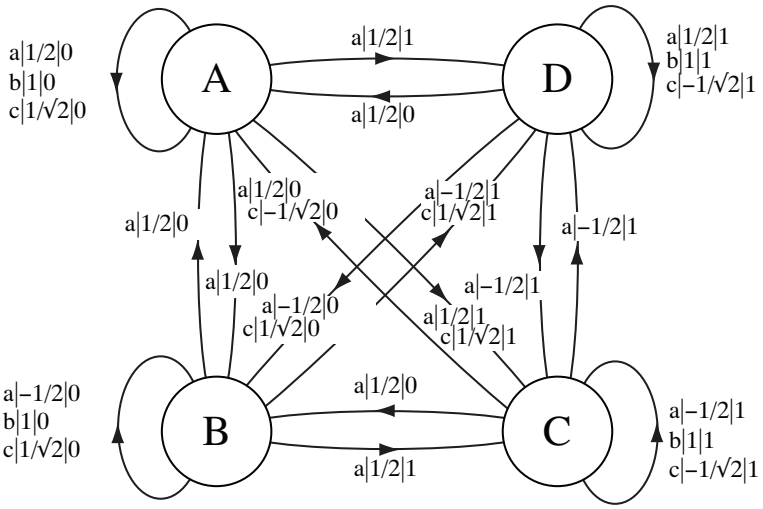

FIG. 13: Quantum transducer for a trapped-ion system exposed to radiation of various wavelengths. The input alphabet $X = \{a, b, c\}$ and output alphabet $Y = \{0, 1\}$ represent unitary operations and electronic states, respectively.

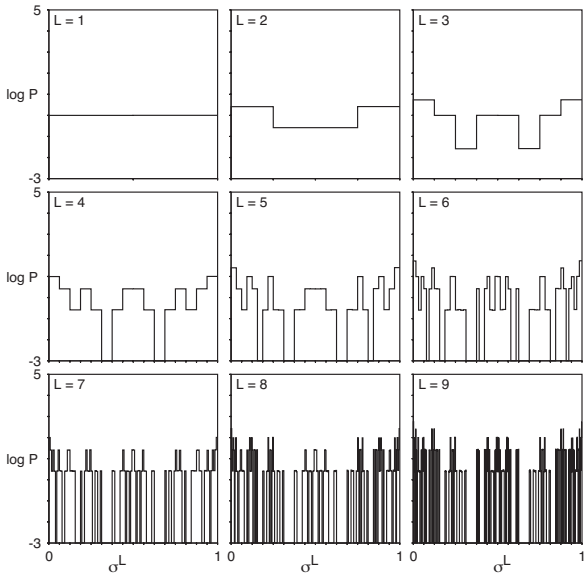

FIG. 14: Process language generated by the trapped-ion quantum dynamical system of Fig. 12 for protocol I (measurements performed at each time step).

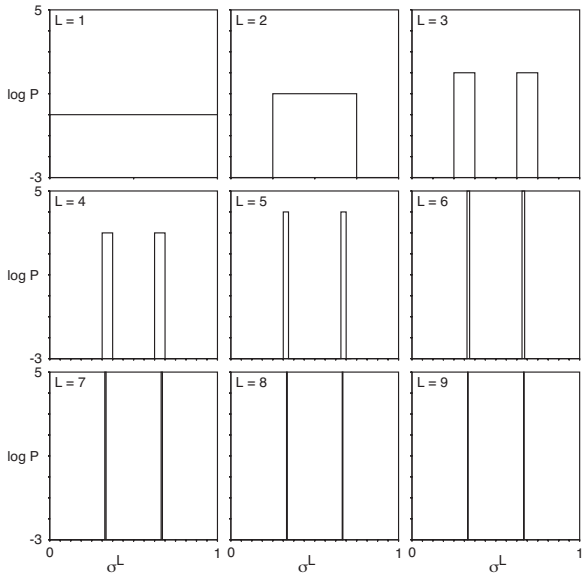

FIG. 15: The generated process languages of the trapped-ion dynamical system from Fig. 12 for measurements performed every three time steps.

2. *Deutsch Algorithm as a Special Case*

It turns out that the trapped-ion experiment implements a quantum algorithm first introduced by Deutsch [6]. The algorithm provided an explicit example of how a quantum machine could be superior to a classical one.

Consider a binary-valued function $f : \{1, 2, \ldots, 2N\} \to \{0, 1\}$. Let U be the device that computes the function f. If we successively apply f to $1, 2, \ldots, 2N$, we get a string x^{2N} of length $2N$. The problem then is to find a true statement about x^{2N} by testing the following two properties:

A: f is not constant: There are not only 0s or only 1s in x^{2N}.

B: f is not balanced: There are not as many 0s as 1s in x^{2N}.

If statement A is false, we can be certain that statement B is true and vice versa. Deutsch and Josza [49] showed that a quantum computer can determine the true state-

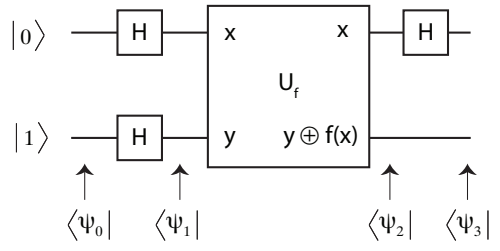

FIG. 16: Deutsch algorithm to classify balanced and constant functions ($N = 2$) depicted as a quantum circuit.

1. Two qubits put in states $\langle 0\|$ and $\langle 1\|$, respectively.	$\langle\psi^0\| = \langle 01\|$
2. Hadamard transform applied to both qubits.	$\langle\psi_1\| = (H \otimes H)\,\langle\psi^0\|$
3. Operation U_f implementing the function $f(x)$ is applied.	$\langle\psi_2\| = (-1)^{f(x)}(I \otimes I)\,\langle\psi_1\|$
4. Hadamard transform applied to the first qubit.	$\langle\psi_3\| = (H \otimes I)\,\langle\psi_2\|$
5. First qubit is measured.	$\langle\psi_3\|\,P(0)$

TABLE I: Deutsch algorithm to determine if $f(x)$ is balanced or constant. H and I are the Hadamard and identity matrices, respectively. $\otimes$ denotes the tensor product.

ment, either A or B, after only two invocations of the operation U, whereas a classical computer requires $N+1$ calls in the worst case. Taking into account the computational steps for establishing the start state and reading out the result, a quantum computer can evaluate the function f in constant time, whereas a classical computer needs a time polynomial in N.

To compare the algorithm with the trapped-ion dynamical system, and to keep issues simple but still informative, we use the basic version ($N = 2$) of the Deutsch algorithm of Ref. [50, p. 32]. (Recall that in our notation $\langle\psi|$ is the state vector, not $|\psi\rangle$, as is common elsewhere.) Figure 16 shows the algorithm as a quantum circuit. Each qubit occupies one horizontal line and the applied unitary transformations are shown as boxes. The overall procedure is summarized in Table I. The unitary operations H and U_f in Fig. 16 are the same as H and U_b in the trapped-ion experiment. The unitary operator in the trapped-ion system is that for a balanced function.

The implementation of the Deutsch algorithm is equivalent to the trapped-ion system under measurement protocol II, with U_b chosen accordingly. Measuring ion 1 after three time steps delivers the desired answer as output (0=A or 1=B). Thus, implementing the Deutsch algorithm corresponds to the trapped-ion system running for three time steps.

The Deutsch algorithm task is solved with a considerable speed-up compared to a classical implementation. Our approach is an extension of this that focuses on what type of computation is carried out intrinsically by the system under continuous external driving and observation. Comparing these two different views of quantum information manipulation—designed quantum computing versus quantum intrinsic computation—suggests that the analysis of NMR experiments with single atoms or molecules in terms of quantum finite-state machines will be a straightforward extensions of the preceding analysis of the Deutsch algorithm.

IX. CONCLUDING REMARKS

We developed a line of inquiry complementary to both quantum computation and quantum dynamical systems by investigating intrinsic computation in quantum processes. Laying the foundations for a computational perspective on quantum dynamical systems we introduced quantum finite-state transducers. Residing at the base of the computational hierarchy, it is the most general representation of a finitary quantum process. It allows for a quantitative description of intrinsic computation in quantum processes—the number of internal states and allowed transitions and the process languages they generate. As far as we are aware, this has not been developed before in the quantum setting.

We laid out the mathematical foundations of these models and developed a hierarchy of classical (stochastic) and quantum machines in terms of the set of languages they recognize or generate. In many cases it turned out that quantum devices were less powerful than their classical analogs. We saw that the limitations of quantum finite-state machines originate in the unitarity of the transition matrices. This suggested that QTs, being reversible, are less powerful than nonreversible classical automata, since the inverse condition constrains the transition matrices.

However, one must be careful to not over-interpret this state of affairs. It has been known for some time that any universal computation can be implemented in a reversible device [51]. Typically, this requires substantially more resources, largely to store outcomes of intermediate steps. In short, reversibility does not, in general, imply less power for classical computers. At the end of the day, computational resources are variables that trade-off against each other. The 2-state QDG examples of the Beam Splitter process illustrated such a trade-off. Although the QDG needs more states than the equivalent SDG to generate the same process language, different measurement protocols yielded a new set of process languages—an aspect that makes QDGs more powerful than SDGs.

These results were then applied to physical systems that could be analyzed in terms of the process languages they generate. One example, that of two trapped ions exhibited a process language of very rich structure. This, and the fact that the system implements a quantum algorithm, opens up a way to an information-theoretic analysis of quantum processes. One can begin to analyze quan-

tum algorithms in terms of their information processing power and do so independent of particular physical implementations.

For example, starting from the computation-theoretic representation of quantum processes presented here, we used tools from information theory, automata theory, and symbolic dynamics to define a measure of intrinsic computation inherent in quantum systems [19, 24]. The basic question one asks about a dynamical system's intrinsic computation—amount of historical information stored, storage architecture, and transformations that produce future behavior—can now be clearly stated. Furthermore, we are currently extending the formalism to more general types of measurements appropriate, for example, to open quantum systems. The power of the resulting quantum transducers is expected to be greater than those here and even greater than the stochastic transducers. In general, we hope that integrating quantum computation and quantum dynamics will receive further attention.

Acknowledgments

The authors thank C. Ellison, D. Feldman, J. Goodwin, J. Mahoney, I. Rumanov, M. Sánchez-Montañés, and C. Strelioff for helpful discussions. This work was supported at the Santa Fe Institute and UCD under the Networks Dynamics Program funded by the Intel Corporation and under the Computation, Dynamics and Inference Program. Direct support was provided by DARPA Agreement F30602-00-2-0583. KW's visit to SFI was partially supported by an SFI Steinmetz Fellowship. KW's postdoctoral fellowship was provided by the Wenner-Gren Foundations, Stockholm, Sweden.

[1] W. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[2] M. A. Arbib, editor. *The Handbook of Brain Theory and Neural Networks*. MIT Press, 2002.

[3] N. Chomsky. Three models for the description of language. *IRE Trans. Info. Th.*, 2:113, 1956.

[4] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, page 124, 1994.

[5] L. K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on the Theory of Computation*, page 212, 1996.

[6] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Roy. Soc. London*, A400:97, 1985.

[7] A Galindo and M. A. Martín-Delgado. Information and computation: Classical and quantum aspects. *Rev. Mod. Phys.*, 74:347–423, 2002.

[8] P. Zoller, Th. Beth, D. Binosi, R. Blatt, H. Briegel, D. Bruss, T. Calarco, J. I. Cirac, D. Deutsch, J. Eisert, A. Ekert, C. Fabre, N. Gisin, P. Grangiere, M. Grassl, S. Haroche, A. Imamoglu, A. Karlson, J. Kempe, L. Kouwenhoven, S. Krll, G. Leuchs, M. Lewenstein, D. Loss, N. Ltkenhaus, S. Massar, J. E. Mooij, M. B. Plenio, E. Polzik, S. Popescu, G. Rempe, A. Sergienko, D. Suter, J. Twamley, G. Wendin, R. Werner, A. Winter, J. Wrachtrup, and A. Zeilinger. Quantum information processing and communication. *Eur. Phys. J. D*, 36:203–228, 2005.

[9] E. Knill, R. Laflamme, R. Martinez, and C.-H. Tseng. An algorithmic benchmark for quantum information processing. *Nature*, 404:368–370, 2000.

[10] J. Berezovsky, M. H. Mikkelsen, O. Gywat, N. G. Stoltz, L. A. Coldren, and D. D. Awschalom. Nondestructive Optical Measurements of a Single Electron Spin in a Quantum Dot. *Science*, 314(5807):1916–1920, 2006.

[11] J. M. Geremia, J. K. Stockton, and H. Mabuchi. Real-Time Quantum Feedback Control of Atomic Spin-Squeezing. *Science*, 304(5668):270–273, 2004.

[12] W. E. Moerner and M. Orritt. Illuminating single molecules in condensed matter. *Science*, 283:1670–1676, 1999.

[13] S. Weiss. Fluorescence spectroscopy of single biomolecules. *Science*, 283:1676–1683, 1999.

[14] M. Rudin. In Ammasi Periasamy and Richard N. Day, editors, *Molecular imaging: FRET microscopy and spectroscopy*. Oxford University Press, Oxford, New York, 2005.

[15] R. Alicki and M. Fannes. *Quantum dynamical systems*. Oxford University Press, 2001.

[16] M. C. Gutzwiller. *Chaos in Classical and Quantum Mechanics*. Springer Verlag, 1990.

[17] S. Habib, K. Jacobs, and K. Shizume. Emergence of chaos in quantum systems far from the classical limit. *Phys. Rev. Lett.*, 96:010403–06, 2006.

[18] D. Lind and B. Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, 1995.

[19] K. Wiesner and J. P. Crutchfield. Computation in sofic quantum dynamical systems. *Lect. Notes Comp. Sci.*, 2007. in press, e-print arxiv/quant-ph/0704.3075.

[20] J. P. Crutchfield. The calculi of emergence: Computation, dynamics, and induction. *Physica D*, 75:11, 1994.

[21] J. P. Crutchfield and Karl Young. Inferring statistical complexity. *Phys. Rev. Let.*, 63:105–108, 1989.

[22] J. P. Crutchfield and K. Young. Computation at the onset of chaos. *Entropy, Complexity and the Physics of Information, SFI Studies in the Sciences of Complexity*, VIII:223–269, 1990.

[23] J. P. Crutchfield and J. E. Hanson. Turbulent pattern basis for cellular automata. *Physica D*, 69:279–301, 1993.

[24] J. P. Crutchfield and K. Wiesner. Intrinsic quantum computation. submitted, e-print arxiv/quant-ph/0611202, 2006.

[25] J. P. Crutchfield. *Reconstructing language hierarchies*, volume 256 of *NATO ASI Series*, pages 45–60. Plenum Press, New York, 1990.

[26] B. Weiss. Subshifts of finite type and sofic systems. *Monastsh. Math.*, 77:462, 1973.

[27] J. P. Crutchfield and D. P. Feldman. Regularities unseen, randomness observed: Levels of entropy convergence. *Chaos*, 13:25 – 54, 2003.

[28] A. Paz. *Introduction to Probabilistic Automata*. New York Academic Press, 1971.

[29] M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[30] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines–Part I. *IEEE Trans. Patt. Anal. Mach. Intel.*, 27:1013–1025, 2005.

[31] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[32] W. Thomas. Automata on infinite objects. In Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 133–164. Elsevier, Amsterdam, 1990.

[33] P. Grassberger. Toward a quantitative theory of self-generated complexity. *Intl. J. Theo. Phys.*, 25:907, 1986.

[34] A. Fraser. Chaotic data and model building. In H. Atmanspacher and H. Scheingraber, editors, *Information Dynamics*, volume Series B: Physics Vol. 256 of *NATO ASI Series*, page 125, New York, 1991. Plenum.

[35] D. Auerbach and I. Procaccia. Grammatical complexity of strange sets. *Phys. Rev. A*, 41:6602–6614, 1990.

[36] W. Yi and H.Xie. Grammatical complexity of unimodal maps with eventually periodic kneading sequences. *Nonlinearity*, 7:1419–1436, 1994.

[37] R. Shankar. *Principles of quantum mechanics*. Plenumy Press, New York, 1994.

[38] R. Freivalds and A. Winter. Quantum finite state transducers. *Lect. Notes Comp. Sci.*, 2234:233–242, 2001.

[39] C. Cohen-Tannoudji, B. Diu, and F. Lalo e. *Quantum mechanics*. John Wiley & Sons, Singapore, 2005.

[40] D. Z. Albert. On quantum-mechanical automata. *Physics Letters*, 98A:249–251, 1983.

[41] A. Peres. On quantum-mechanical automata. *Physics Letters*, 101A:249–250, 1984.

[42] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *38th IEEE Conference on Foundations of Computer Science*, pages 66–75, 1997.

[43] D. Aharonov, A. Kitaev, and N. Nisan. Quantum circiuts with mixed states. *STOC*, pages 20–30, 1998.

[44] A. Bertoni and M. Carpentieri. Analogies and differences between quantum and stochastic automata. *Theor. Comp. Sci.*, 262:69–81, 2001.

[45] A. Nayak. Optimal lower bounds for quantum automata and random access codes. *Proceedings of the 40th FOCS*, 40th Annual Symposium on Foundations of Computer Science:369, 1999. e-print arxiv/quant-ph/9904093.

[46] C. Moore and J. P. Crutchfield. Quantum automata and quantum grammars. *Theor. Comp. Sci.*, 237:275–306, 2000.

[47] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[48] A. Peres. *Quantum Theory: Concepts and Methods*. Kluwer Academic Publishers, 1993.

[49] D. Deutsch and R. Josza. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. London, Mathematical and Physical Sciences*, 439:553–558, 1992.

[50] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000.

[51] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17(6):525–532, 1973.

[52] The generalization to open quantum systems using any (including non-orthogonal) *positive operator valued measures* (POVM) requires a description of the state after measurement in addition to the measurement statistics. Repeated measurement, however, is the core of a quantum process as the term is used here. We therefore leave the discussion of quantum processes observed with a general POVM to a separate study.

[53] The vector $\langle 00|$ is shorthand notation for $\langle 10| \otimes \langle 10|$, where both ions are in state 0.