Computation in Finitary Quantum Processes

Karoline Wiesner
James P. Crutchfield

SFI WORKING PAPER: 2006-09-031

SANTA FE INSTITUTE

Santa Fe Institute Working Paper 06-08-XXX
arxiv.org/quant-ph/0608XXX

Computation in Finitary 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: August 27, 2006)

We introduce quantum finite-state generators as a first step toward completing a computational description of observing individual quantum systems over time. We develop the mathematical foundations of quantum finite-state machines and compare nondeterministic and deterministic versions to stochastic generators and recognizers, summarizing their relative computational power via a hierarchy of finitary process languages. Quantum finite-state generators are explored via several physical examples, including the iterated beam splitter, the quantum kicked top, 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 measurement protocol.

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

Contents

I. INTRODUCTION

Recent developments in molecular spectroscopy [1–3] and experimental quantum computation [4] allow for the preparation and control of individual quantum systems that can be observed over long times through repeated measurements. Examples abound in single molecule spectroscopy [5–9] where the experimentalist ties a single molecule to a substrate and records its time-dependent fluorescence over milliseconds. In quantum computation the experimentalist subjects information stored in an initially coherent set of physical degrees of freedom—the so-called *qubits* stored, for example, in electronic levels [10, 11] or photon polarization [11, 12]—to a selected sequence of manipulations. The system's resulting state is

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

measured and interpreted as the output of a computation.

Quantum theory, though, often focuses only on predicting the expectation of outcomes from an ensemble of isolated measurements, which is the mean value of an observable [13]. For molecular and quantum computation experiments this is insufficient, since one needs to describe a system's *behavior*. Quantum mechanics can be extended to address behavior [14], but this still 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? 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 these issues.

Due to the range of topics, in the following we try to give a self-contained treatment. We develop what is needed from automata and formal language theory as needed, though familiarity with those areas is helpful. However, readers are expected to be familiar with basic quantum physics. Citations to reference texts are given at the appropriate points.

A. Computation versus dynamics

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 [15, 16]. 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 [17]. 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 recent years the study of computation with elementary components that obey quantum mechanical laws has developed into a highly active research area; see, for instance, Ref. [18] and citations therein. This is driven, in part, by the invention in 1994 of an algorithm for factoring integers that requires a number of quantum computational steps polynomial in the number of digits in an integer [19], as opposed to an exponential number of classical logical operations. Current studies of quantum computation are focused, on the experimental side, on implementing realizations in various physical substrates [10, 20–27] and, on the theoretical, on designing schemes for suppressing error accumulation and the decoherence of quantum states; see, for example, Refs. [28–30].

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 [31]. A well analyzed example is found in the eigenvalue statistics and energy level spacing in atoms [32]. Here one oft-studied prototype model is that of the quantum kicked rotor [33].

Studies of quantum chaos are, in effect, extensions of the theory of nonlinear classical dynamics. Quantum maps mimicking classical nonlinear discrete-time maps have been developed for many chaotic systems, such as the Baker's map [34] and the logistic map [35]. In these and other cases, phase space exhibits (transient) chaotic regions, dense periodic orbits, and other characteristics from nonlinear classical systems. A different approach to quantum chaos is found in the study of discrete finite-dimensional dynamical systems. An example is the well studied toy system of the quantum kicked top—a finite-dimensional spin system [14, 36]. The approach there focuses on the von Neumann entropy, a measure of how mixed a quantum state is. The von Neumann entropy is used as a signature of the transition from the (linear) quantum regime to the semi-classical chaotic regime.

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 it generates [37]. 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 [38–40]. 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.

B. Overview

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.

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 model *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. 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, generators are a key tool for answering questions about the information processing capabilities inherent in natural processes [41].

The theory of stochastic processes [42], in contrast to computation theory, concerns itself almost exclusively with the generation of behavior. There one finds methods to quantitatively characterize statistical properties and predictive models of processes. It, however, does not typically address questions of how stochastic processes store and manipulate information. That is, it does not directly analyze the question of how structured stochastic processes are. The following attempts to answer this question by using both calculation methods from stochastic process theory and architectural classifications from discrete computation theory.

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 a presumed 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. (Methods that will be presented elsewhere.) 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. 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.

To lay the foundations for a computational perspective on quantum dynamical systems we introduce a class of finite-state automata called *quantum finite-state generators*. We arrive at this by recapitulating the steps in the brief history of automata theory with which we opened. In the next section we review various kinds of formal languages and the finite-state machines that recognize and generate them, both probabilistic and nonprobabilistic. 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 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. FORMAL LANGUAGES FOR BEHAVIOR

Our use of formal language theory differs from most 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 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. Similarly, Σ^+ is the set of all words of at least length one or longer. 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:*

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

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

There are several additional notions that we will need later.

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.*

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 $0.\sigma^L = \sum_{t=0}^{L-1} \sigma_t 2^{-t} \in [0, 1]$. 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 $[0, 1]$ [43]. 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 $\phi = \frac{1}{2}(1+\sqrt{5})$.

III. STOCHASTIC FINITE-STATE MACHINES

Our development concerns only models that have finite memory. As was the case in the early history of automata theory, finite-memory models are an important starting point for more sophisticated computational devices. Moreover and more practically, given the very finite state of affairs in experimental realizations, this class of models appropriately describes the behavior of currently feasible quantum computation devices. With this in mind the use of *finite* needs clarification. It turns out that using "finite" in the present setting can be misleading, since some finitely specified processes have various kinds of infinite memory. (Examples will be discussed.) And so we refer to the broad class generically as *finitary*. For a direct definition see Ref. [44], although the examples in the following illustrate some of the subtleties.

Automata with finite memory—*finite-state machines*—consist of a finite set of states and transitions between them [45]. As already noted, typically they are used as *recognition* devices, whereas we are interested in the generation of words in a process language. We therefore will review models for both generation and recognition in the following. In addition to laying out some

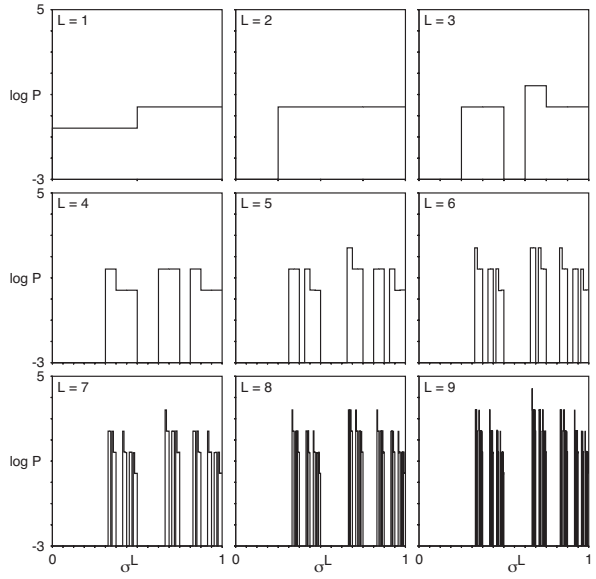

FIG. 1: Example of a process language: In the Golden Mean Process, with alphabet $\Sigma = \{0, 1\}$, word 00 and all words containing it have zero probability. All other words have nonzero probability. The logarithm base 2 of the word probabilities is plotted versus the binary string σ^L, represented 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.

required tools, the review will give a concrete sense of the differences between recognition and generation.

Notation. *Let $\mathcal{L}(\mathcal{M})$ denote the language recognized or generated by automaton $\mathcal{M}$.*

The immediate goal is to extend conventional finite-state machines to include first probabilistic and finally quantum components. We do this by adapting Rabin's and Paz's *stochastic sequential machines* [46, 47] to define stochastic *recognizing* and *generating* finite-state machines. To be consistent with our simplified naming scheme, which will be discussed shortly, we refer to these simply as *stochastic machines* (SMs).

Generally, a SM operates by reading in symbols that, along with the current state, determine the next state(s) and output symbol(s). A SM 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. SMs 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.

But, first, several definitions are in order.

Definition. *A square matrix T is* stochastic *if all its entries T_{ij} are non-negative and each row sums to 1: i.e., $\sum_j T_{ij} = 1$ and $0 \leq T_{ij} \leq 1$. The* dimension dim T *of T is the number of rows or columns. A* substochastic *matrix is defined similarly except that $\sum_j T_{ij} \leq 1$. A* stochastic set $\{T(\sigma) : \sigma \in \Sigma\}$ *is a collection of substochastic matrices whose component-wise sum:*

$$T = \sum_{\sigma \in \Sigma} T(\sigma) \tag{4}$$

is stochastic. Finally, a doubly stochastic matrix *is a stochastic matrix in which column elements sum to one as well: $\sum_i T_{ij} = 1$.*

Definition. *The set of matrices $\{T(y|x)\}$ is defined over input symbols $x \in X$ and output symbols $y \in Y$, where each $T(y|x)$ is a substochastic matrix. For each $x \in X$ we have the stochastic matrix:*

$$T(x) = \sum_{y \in Y} T(y|x) \ . \tag{5}$$

Given a distribution $\Pr(x)$ *over input symbols, the stochastic matrix T associated with the set $\{T(x)\}$ of stochastic matrices is:*

$$T = \sum_{x \in X} \Pr(x) T(x) \ . \tag{6}$$

Definition. *[47] A* stochastic machine *(SM) $\mathcal{M}$ is a tuple $\{S, X, Y, \{T(y|x)\}\}$ where*

1. *S is a finite set of states.*
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 stochastic set of square matrices of* dim $|S|$. *There are $|X| \times |Y|$ such matrices. The components $T_{ij}(y|x) > 0$ give the probability of moving to state s_j and emitting output y when reading input x in state s_i.*
4. *At each step a symbol $x \in X$ is read from the input word, a symbol $y \in Y$ is output, and the machine updates its state. Thus, $\sum_{y \in Y} \sum_j T_{ij}(y|x) = 1$. (The notation $y|x$ is the label for a transition and does not itself imply a conditional probability.)*

A. Structural properties

The set $\{T(y|x)\}$ can be represented as a directed graph $G(T)$ 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 transition probability from state s_i to state s_j. Edges are labeled $x|p|y$ with the input symbol $x \in X$, transition probability $p = T_{ij}(y|x)$, and output symbol $y \in Y$.

In a SM one associates outputs with transitions. (In fact, what we have defined is a *Mealy SM*, which differs from the alternative, and equivalent, *Moore SM* in which output is associated with a state [47].)

Definition. *A directed graph G is* connected *if there is at least one* path *(sequence of edges) with $T_{ij} > 0$ between every pair of states s_i and s_j in G.*

Definition. *A directed graph G is* strongly connected *if for every pair of states, s_i and s_j, there is at least one path from s_i to s_j and at least one from s_j to s_i.*

A SM's states can be classified as follows.

Definition. Recurrent states *of a SM $\mathcal{M}$ are the strongly connected subsets of S in the directed graph $G(\mathcal{M})$.*

Definition. *A SM's* asymptotically recurrent states *are subsets of recurrent states from which no path leaves.*

Definition. *A SM's* transient states *are those that are not asymptotically recurrent.*

Generally speaking, a SM 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 SMs that have only a single set of asymptotically recurrent states.

B. Process languages

SMs generate, recognize, and transform process languages. Before we can discuss the languages associated with a SM, though, we must introduce the matrix notation required for analysis.

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

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 s_i. *The vector is normalized in probability: $\sum_{i=0}^{|S|-1} \pi_i = 1$. The* initial state distribution *is denoted $\langle\pi^0|$.*

The state-to-state transition probabilities of a SM, independent of inputs and outputs, are given by the *state-to-state transition matrix*:

$$T = \sum_{x \in X, y \in Y} \Pr(x) T(y|x) \ . \tag{7}$$

This is a stochastic matrix. It depends on the distribution of inputs that drive the SM.

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

$$\langle\pi^s| = \langle\pi^s| T \ , \tag{8}$$

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

Notation. *For a series of L input and output symbol pairs $(y_i|x_i)$ the action of the corresponding SM 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 s_i to s_j on reading input x^L and generating output y^L.

If the SM starts with state distribution $\langle\pi^0|$, the probability of generating y^L when reading x^L is

$$\Pr(y^L|x^L) = \langle\pi^0| T(y^L|x^L) |\eta\rangle \ . \tag{9}$$

As mentioned already, the notation $y|x$ on the righthand side of the above equation reflects transition labeling. On the lefthand side it should be read as connoting conditional probability. (For those familiar with formal language theory, $|\eta\rangle$ here represents the assumption that all states are accepting. This, in turn, is a consequence of our focusing on process languages, which are subword closed.)

If one is ignorant about the input, the probability $\Pr(y^L)$ of generating output word y^L when starting in state distribution $\langle\pi^0|$ is obtained by summing over all inputs:

$$\Pr(y^L) = \sum_{x^L \in X^L} \langle\pi^0| T(y^L|x^L) |\eta\rangle \ . \tag{10}$$

The state distribution $\langle\pi(x^L)|$ after reading in word x^L starting in state distribution $\langle\pi^0|$ (and ignorant of the output generated) is

$$\langle\pi(x^L)| = \sum_{y^L \in Y^L} \langle\pi^0| T(y^L|x^L) \ . \tag{11}$$

Similarly, the state distribution after having produced output y^L is:

$$\langle\pi(y^L)| = \sum_{x^L \in X^L} \langle\pi^0| T(y^L|x^L) \ . \tag{12}$$

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.

C. Classifications

Now we are ready 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 after the whole word is read in: *accept* or *reject*. In contrast, a generator does not have input. It, however, spontaneously produces an ongoing output. A generator has no built-in stop signal, unlike a recognizer which uses the end of its input word. We view generators as running constantly and for a long time.

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 [48].

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 M. These will be specialized to *recognizers*, abbreviated R, and *generators*, denoted G. Within these basic machine types, there will be various alternative implementations. For example, we will discuss *stochastic* (S) and *quantum* (Q) versions. We refer to machine types familiar from formal language and automata theory, however, implicitly; that is, by not using the adjectives stochastic or quantum.

Within these classes we will also distinguish additional subsidiary properties, such as *determinism*, denoted D. If a property label is not explicitly given, then we intend to convey the opposite property. So, if no D appears in an abbreviation, the implication is that the corresponding machine is *nondeterministic*.

Thus, QM refers to a transducer built out of quantum states, SR to a stochastic nondeterministic recognizer, and SDG to a stochastic deterministic generator. Of course, this use of language and abbreviations (and any seeming conflicts, such as " stochastic deterministic") will make sense once the various computational models have been introduced.

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.

D. Recognizers

When machines are used to determine membership of a word in a language we call them *recognizers* (Rs). We define recognizers as a special case of SMs: They ignore the probabilistic aspects of transitions and, once a word is read, produce a single output symbol that indicates whether or not the word is accepted. We start by defining recognizers, using matrix notation, which is typically not found in their presentation in formal language theory. As we will see, though initially cumbersome, this approach is adapted to facilitate introducing the various generalized models to come and it allows one to directly calculate many properties of interest.

Definition. *A* recognizer *(R) is a SM with transition matrices* $\{T(y|x) = T(x)\}$ *and output alphabet* $Y = \{accept, reject\}$.

Definition. *A R* accepts *an input word if, on reading each symbol in sequence, it follows a series of allowed transitions (*$T_{ij}(x) > 0$*). A word is* rejected *if a disallowed transition (*$T_{ij}(x) = 0$*) is encountered.*

We will not explicitly represent the zero-probability transitions. This, the fact that all states are accepting, and the simple notion of recognition mean that one, in effect, ignores the SM's sequence of outputs.

The unique language $\mathcal{L}$ that is accepted by a R is defined as the largest subset of X^* such that all and only $w \in \mathcal{L}$ are accepted.

An example R is shown in Fig. 2 which recognizes the support of the Golden Mean process language of Fig. 1.

There is a duality between devices and the languages that they recognize. In the present case we have the following definition of languages that are recognized by finite-memory automata.

Definition. *A* regular language *is a formal language recognized by some R.*

A R's substochastic matrices $T(x)$ can contain more than one nonzero element per row and so starting in the state associated with that row, there can be multiple destination states when reading in $x \in X$. The result is that when reading a word it is possible to follow multiple paths through the machine. One views a R as potentially being in multiple states at any given time.

In automata theory this is referred to as *nondeterminism*. It does not mean "probabilistic", but rather that the input word *does not determine* a unique sequence of transitions. That is, nondeterminism means simply that there can be more than one transition per symbol read. In this case, recognition of the input word requires that there be *at least* one path of allowed transitions followed when reading the word. Note that the possibility of rejecting a word is implicit in the graphical representation: we do not display disallowed transitions.

If one wishes to emphasize this aspect of a R, one can refer to a *nondeterministic finite-state recognizer*. Here when we refer to a R, we intend to mean the general case of nondeterminism which, as mathematical habit goes, includes determinism.

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

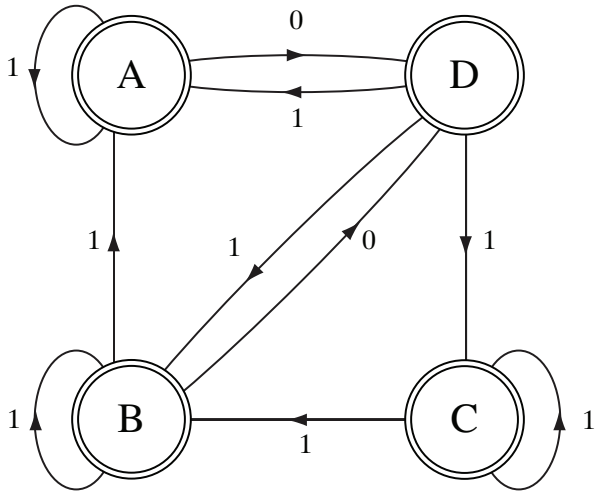

FIG. 2: Example of a nondeterministic finite-state recognizer that accepts the process language (Fig. 1) of words with no consecutive 0*s* (Golden Mean Process). Double circles denote start states, edges are labeled with input symbols $\sigma \in \Sigma = \{0, 1\}$. Disallowed transitions are omitted for simplicity.

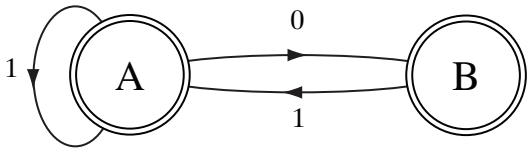

FIG. 3: Example of semi-deterministic recognizer that recognizes the same formal language as the preceding R.

Semi-determinism means that transitions from each state lead to one and only one next state for each input symbol. If no allowed transition exists for an input symbol encountered while reading a word, the word is rejected. An example of a semi-deterministic finite-state recognizer is shown in Fig. 3. It recognizes words in the same process language, shown in Fig. 1, that the preceding R (Fig. 2) recognizes.

With a semi-deterministic recognizer we do not know in which state to start. This leads to a slightly refined, fully deterministic version in which the choice of start state is removed.

Definition. *[45] A* deterministic recognizer *(DR) is a semi-deterministic finite-state recognizer with a designated start state $s_0 \in S$.*

For a DR each recognized input word corresponds to one and only one path or series of transitions. Moreover, due to determinism, one views a DR as being in one and only one state at each time step. The state-to-state transition matrix for a DR is simply given by

$$T = \sum_{x \in X} T(x) \ . \tag{13}$$

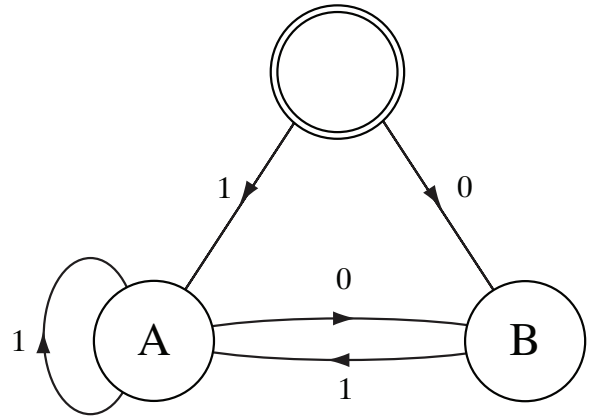

FIG. 4: Example of a deterministic finite-state recognizer that accepts the same language as the semi-deterministic recognizer in Fig. 3.

We can now draw out several comparisons. Contrast the above equation for T to Eq. (7). While the stochastic set of matrices for a R is defined only over the input alphabet, for SMs the state-to-state transition matrix T requires specifying an output alphabet and a distribution over input symbols. That is, the latter is not part of the construction of a SM.

Figure 4 shows an example of a DR that recognizes the support of a process language with no consecutive 0s shown in Fig. 1, the same formal language that is accepted by the R in Fig. 2 and the semi-deterministic recognizer in Fig. 3. In turning the semi-deterministic recognizer into a DR a single start state has been added; it is a transient state.

Note that for all machine types that recognize the formal language of Fig. 1 there are no consecutive transitions on a 0. That is, all reject any word containing two consecutive 0s. This leads, in turn, to the self-similar structure of the word probability distribution noted in Fig. 1. A useful way to characterize this property is to give a process language's list of *irreducible forbidden words*—the shortest words that are disallowed. 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.

We end our discussion of recognizers (as studied in formal language theory) by recalling several important properties.

Proposition 1. *A R can be transformed into an equivalent DR that recognizes the same language.*

Proof. *The conversion method is given in Ref. [45].* □

Corollary 1. *A semi-deterministic recognizer can be transformed into an equivalent DR.*

Proof. *Apply the method just cited.* □

The well known consequence of these results is that Rs, semi-DRs, and DRs all recognize the same class of languages—the regular languages [45].

E. Stochastic recognizers

Probabilistic automata [46, 47] are typically defined as recognizers with a threshold. A probabilistic automaton assigns a probability to the word it is reading and, if that probability is above the automaton's threshold, the word is recognized; if below, it is rejected. Here we introduce a different kind of probabilistic automaton—a *stochastic recognizer* (SR)—that recognizes both the *set* of words and the associated word distribution. Again, we concentrate on process languages. For a similar approach to probabilistic automata see Refs. [49, 50].

The first criterion for accepting a word is that the word leads the machine through a series of transitions with positive probability. The second, and new, criterion is that the probability assigned by the SR is equal to the word's probability within a tolerance δ. This is essentially different and more restrictive than employed in alternative developments. A SR not only tests for membership in a formal language, it also recognizes a *function*: the probability distribution of the language.

Definition. *A* stochastic recognizer (SR) *is a finite-state recognizer that assigns the number* $\langle \pi^0 | T(w) | \eta \rangle$ *to a word* w.

Definition. *Given a probability distribution* $Pr(w)$*, a SR* accepts*, with respect to state distribution* π^0 *and with a word-probability tolerance* $0 \leq \delta \leq 1$*, a word* $w = x_0 x_1 \ldots x_{L-1}$*,* $x_t \in X$*, if, on reading each symbol,*

1. *it follows a series of allowed transitions* ($T_{ij}(x_t) > 0$) *and*
2. $|\Pr(w) - \langle \pi^0 | T(w) | \eta \rangle| \leq \delta$.

In other words, a SR accepts a stochastic language $\mathcal{S}$ if it assigns a positive probability to each $w \in \mathcal{S}$ and zero probability to $w \notin \mathcal{S}$. In addition, starting the SR in $\langle \pi^0 |$ the probability the SR assigns must be within tolerance δ of that specified by $\Pr(w)$ for all $w \in \mathcal{S}$. For example, if $\delta = 0$ the SR accepts exactly the probability distribution. If $\delta > 0$ it accepts the probability distribution with some "fuzziness", still rejecting all probability-0 words.

Paralleling the definitions of recognizers above, we have the following special classes of stochastic recognizers.

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

Definition. *A* stochastic deterministic recognizer *(SDR) is a stochastic semi-deterministic recognizer with a designated start state* $s_0 \in S$.

If we have a semi-deterministic recognizer and assume that we extract a word produced by a process which has been running for a long time, we take the initial state distribution to be the stationary distribution: $\langle \pi^0 | = \langle \pi^s |$. If, however, we have a deterministic recognizer and know in which state the process started, we take $\langle \pi^0 | = (1, 0, \ldots, 0)$, which is the initial state distribution with all of the probability concentrated in the presumed start state. These two cases effectively model two different experimental set-ups. In the first case, the process has been operating for a while and the experimentalist has lost track of (or never knew) the exact state it was in. In the latter case, the system is carefully prepared in a particular start state or start-state distribution and so the experimentalist has full knowledge about the prepared state.

As with the deterministic recognizer introduced in the previous section, a word that is accepted by a 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 a SDR assuming the process starts in state distribution $\langle \pi^0 | = (1, 0, \ldots, 0)$:

$$\Pr(x^L) = T_{s_0 s_1}(x_0) T_{s_1 s_2}(x_1) \cdots T_{s_{L-2} s_{L-1}}(x_{L-1}) \ , \quad (14)$$

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

There is an important difference here with Eq. (10) for word probabilities assigned by SMs. 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 SMs.

Figure 5 shows an example of a SDR that recognizes the Golden Mean process language, including its probability distribution (Fig. 1). If we take the tolerance 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:

$$T(0) = \begin{pmatrix} 0 & 0 & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix} \text{ and } T(1) = \begin{pmatrix} \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 \end{pmatrix} . \quad (15)$$

We use Eq. (14) with the start state distribution $\langle \pi^0 | = (1, 0, 0)$ to calculate the $L = 1$ word probabilities:

$$\begin{aligned} \Pr(0) &= \langle \pi^0 | T(0) | \eta \rangle = \tfrac{1}{3} \ , \\ \Pr(1) &= \langle \pi^0 | T(1) | \eta \rangle = \tfrac{2}{3} \ . \end{aligned} \quad (16)$$

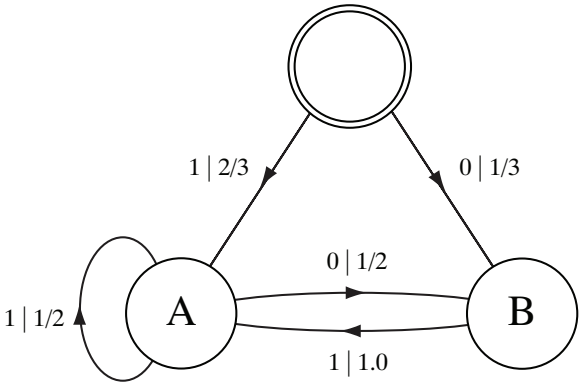

FIG. 5: Stochastic deterministic recognizer for the Golden Mean process language of Fig. 1. The edges are labeled $x|p$, where $x \in X$ and $p = T_{ij}(x)$. The initial state $\langle \pi^0 | = (1, 0, 0)$ is double circled.

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

$$\begin{aligned} \Pr(011) &= \langle \pi^0 | T(011) | \eta \rangle \\ &= \langle \pi^0 | T(0)T(1)T(1) | \eta \rangle \\ &= \tfrac{1}{6}. \end{aligned} \tag{17}$$

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:

$$\begin{aligned} \Pr(1|0) &= \frac{\Pr(01)}{\Pr(0)} \\ &= \frac{\langle \pi^0 | T(0)T(1) | \eta \rangle}{\langle \pi^0 | T(0) | \eta \rangle} \\ &= 1 \ . \end{aligned} \tag{18}$$

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

F. 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. [38, 40, 41, 43, 51–54], for example.

As we also noted, automata that only generate outputs are less often encountered in formal language theory than automata operating as recognizers. One reason is that re-defining 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, but assuming an inherent probabilistic mechanism.

A probabilistic generator is a SM that operates independent of input. It has an internal clock and at each step makes a transition from one state to another and generates an output symbol.

Definition. *A* generator *(G) is a SM with a stochastic set* $\{T(y|x) = T(y)\}$.

Definition. *A* semi-deterministic generator *is a 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. Also, as before, semi-deterministic generators do not necessarily specify a unique start state. And this observation leads to a by-now familiar refinement.

Definition. *A* deterministic finite-state generator *(DG) is a semi-deterministic generator with a designated start state* $s_0 \in S$.

In the following we concentrate on deterministic finite-state generators. As an example, consider the generator for the Golden Mean process language, shown in Fig. 6. Its matrix representation is almost the same as for the Golden Mean recognizer given in Eqs. (15) and Fig. 4. Due to the latter's determinism, one can construct a generator simply by swapping input symbols to output symbols and eliminating the transient state. (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 DG's matrix representation:

$$T(0) = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & 0 \end{pmatrix} \text{ and } T(1) = \begin{pmatrix} \frac{1}{2} & 0 \\ 1 & 0 \end{pmatrix}. \tag{19}$$

The stationary state distribution $\langle \pi^s |$ is calculated as the left eigenvector of the state-to-state transition matrix T, Eq. (13):

$$\langle \pi^s | = \langle \tfrac{2}{3}, \tfrac{1}{3} | \ . \tag{20}$$

Thus, each state is assigned a probability $\Pr(s_i) = \pi_i^s$, where π_i^s 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. (14) with the stationary distribution $\langle \pi^s |$ to calculate the $L = 1$ word probabilities:

$$\Pr(0) = \langle \pi^s | T(0) | \eta \rangle = \tfrac{1}{3} \ , \tag{21a}$$

$$\Pr(1) = \langle \pi^s | T(1) | \eta \rangle = \tfrac{2}{3} \ . \tag{21b}$$

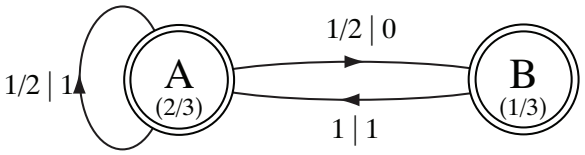

FIG. 6: A semi-deterministic generator for the Golden Mean Process. Edges are labeled $p \,|\, y$, where $y \in Y$ and $p = T_{ij}(y)$. The numbers in parentheses give a state's asymptotic probability.

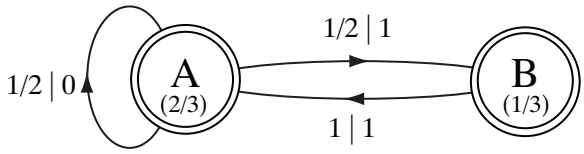

FIG. 7: A semi-deterministic generator of the Even Process: Blocks of an even number of 1s are separated by 0s. Edges are labeled $p \,|\, y$, where $y \in Y$ and $p = T_{ij}(y)$. The numbers in parentheses give a state's asymptotic probability.

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

$$\begin{aligned} \Pr(011) &= \langle \pi^s | T(011) | \eta \rangle \\ &= \langle \pi^s | T(0)T(1)T(1) | \eta \rangle \\ &= \tfrac{1}{6}. \end{aligned} \tag{22}$$

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 follows in a similarly straightforward manner:

$$\begin{aligned} \Pr(1|0) &= \frac{\Pr(01)}{\Pr(0)} \\ &= \frac{\langle \pi^s | T(0)T(1) | \eta \rangle}{\langle \pi^s | T(0) | \eta \rangle} \\ &= 1 \ . \end{aligned} \tag{23}$$

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

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

$$T(0) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix} \text{ and } , T(1) = \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & 0 \end{pmatrix} \ . \tag{24}$$

The corresponding graph is shown in Fig. 7. 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 semi-DG representation let's us see how. In particular, the set of irreducible forbidden words is countably infinite [55]: $\mathcal{F} = \{01^{2k+1}0 : \ k = 0, 1, 2, \ldots\}$. 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-finitary: 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.

G. Stochasticity, recognition, and generation

We can now describe the similarities and differences between stochastic and other kinds of recognizers and between the various classes of generators.

Consider the process language $\mathcal{P} = \{\{1^+\}, \Pr(1^+) = 1\}$—the language of all 1s. The support of this language is accepted by the Rs in Figs. 2, 3, and 4. Since the support and the probability distribution differ substantially from those of the Golden Mean process language, though, $\mathcal{P}$ is not accepted by the SDR of Fig. 5, for example, at any tolerance threshold.

Recall that $\mathcal{L}(\mathcal{M})$ is the language associated with (recognized or generated by) automaton $\mathcal{M}$. For (non-stochastic) recognizers $\mathcal{M}$, $\mathcal{L}(\mathcal{M})$ is a regular language. If $\mathcal{M}$ is a stochastic machine then, $\mathcal{L}(\mathcal{M})$ is a process language.

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 2. *Every regular language is generated by some G.*

Proof. *Let R be a finite-state machine recognizing the regular language $\mathcal{L}$. Consider R's transition matrices $T(x)$ and form a new set $T(y)$ in which $Y = X$. The $T(y)$ define a generator, the support of whose output language is $\mathcal{L}$.* □

Proposition 3. *Every $\mathcal{L}(G)$ is recognized by some SR.*

Proof. *Consider G's transition matrices $T(y)$ and form a new set $T(x)$ in which $X = Y$. The $T(x)$ define a SR that recognizes $\mathcal{L}(G)$.* □

Proposition 4. *For every G,* supp $\mathcal{L}(G)$ *is a regular language.*

Proof. *Consider G's transition matrices $T(y)$ and form a new set $T(x)$ in which $X = Y$. The $T(x)$ define a SR that recognizes the process language $\mathcal{L}(G)$ and so* supp $\mathcal{L}(G)$ *is a regular language, by definition.* □

Proposition 5. *The language generated by a G is a process language.*

Proof. *The first property to establish is that the set of words produced by a G is subword closed: if* $\Pr(y^L) > 0$*, then all $w \in \text{sub}(y^L)$ have* $\Pr(w) > 0$*. Since* $\Pr(y^L) = \langle \pi^s | T(y^L) | \eta \rangle > 0$*, then there is at least one path $s_0 s_1 \ldots s_L$ through G along which the single-symbol transition probabilities are positive: $T_{s_t s_{t+1}}(y_t) > 0, t = 0, \ldots, L-1$. A subword $w \in \text{sub}(y^L)$ corresponds to a segment of the same path, which therefore also has positive transition probabilities. Therefore,* $\Pr(w) > 0$.

The second property to establish is that the word distribution $\Pr(y^L)$ *is normalized at each L. This follows by direct calculation:*

$$\begin{aligned}
\sum_{y^L \in Y^L} \Pr(y^L) &= \langle \pi^s | \sum_{y^L \in Y^L} T(y^L) | \eta \rangle \\
&= \langle \pi^s | \sum_{y_{L-1} \in Y} T(y_{L-1}) \sum_{y^{L-1} \in Y^{L-1}} T(y^{L-1}) | \eta \rangle \\
&= \langle \pi^s | T \sum_{y^{L-1} \in Y^{L-1}} T(y^{L-1}) | \eta \rangle \\
&= \langle \pi^s | \sum_{y^{L-1} \in Y^{L-1}} T(y^{L-1}) | \eta \rangle \\
&= \ldots \\
&= \langle \pi^s | \eta \rangle = 1 \ . \qquad (25)
\end{aligned}$$

□

The remainder of the results in this section use essentially the same proof methods as the preceding ones or draw together previous results and so they are stated without proof.

Proposition 6. *Every $\mathcal{L}(SR)$ is generated by some G.*

Corollary 2. $\mathcal{L}(SR) = \mathcal{L}(G)$.

Proposition 7. *Every $\mathcal{L}(SDR)$ is generated by some DG.*

Proposition 8. *Every $\mathcal{L}(DG)$ is recognized by some SDR.*

Corollary 3. $\mathcal{L}(SDR) = \mathcal{L}(DG)$.

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

Proposition 9. *There exists a G such that $\mathcal{L}(G)$ is not recognized by any SDR.*

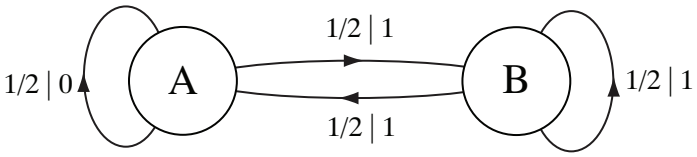

FIG. 8: A nondeterministic generator that produces a process language not recognized by any (finite-state) SDR. Edges are labeled $p \,|\, y$, where $y \in \{0, 1\}$ and $p = T_{ij}(y)$.

Proof. *We establish this by example. Consider the nondeterministic generator in Fig. 8—the* Simple Nondeterministic Source *(SNS). The construction of a R given in the proof of Prop. 2 does not yield a SDR. To show that there is no other possible construction of a 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. [44] gives a SDR for this process that is minimal, but has a countably infinite number of states. Note that supp $\mathcal{L}(SNS)$ is the Golden Mean formal language.

Proposition 10. *There exists a SR such that $\mathcal{L}(SR)$ is not generated by any DG.*

Proof. *The proof method is very nearly the same as the previous one and so we will not repeat it.* □

These propositions say, in essence, that deterministic machines generate or recognize only a subset of the finitary process languages. In particular, Props. 7 and 9 imply proper containment: $\mathcal{L}(SDR) \subset \mathcal{L}(G)$. As do Props. 8 and 10: $\mathcal{L}(DG) \subset \mathcal{L}(SR)$. This is in sharp contrast with the familiar result in formal language theory: deterministic and nondeterministic automata recognize the same class of languages—the regular languages [45].

This ends our review of classical machines and their various specializations. We now enter the discussion of their quantum analogs, using a strategy that will be familiar. The reader should have a working knowledge of quantum theory at the level of, say, Refs. [13] and [56].

IV. QUANTUM PROCESSES AND LANGUAGES

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 [19]. Other results include Grover's quantum search algorithm from 1996 [57]. These results assume the use of powerful computational architectures, such as quantum Turing machines [58], that are decidedly more powerful than finite-state machines.

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* [26]. Quantum finite-state automata have drawn much interest in the last decade for this reason. They reflect the capabilities of currently feasible quantum computers. (For a review of theoretical and experimental studies of quantum computation see, for example, Ref. [18].) 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.

The study of quantum finite-state automata and the languages they recognize has produced a veritable zoo of alternative models. Of these, here we can mention only those incorporating the most basic concepts, since this allows for a comparison with the classical (stochastic) models just covered. To simplify this endeavor we start out defining a basic quantum-finite state automaton. By specializing the latter we develop a series of quantum finite-state automaton models that are useful for modeling intrinsic computation in finitary quantum processes.

A. Quantum languages

As in the classical setting, we link a quantum system's behavior to a language that describes it. In particular, a word corresponds to a path in the state space of a physical quantum system. Quantum mechanics assigns a *complex amplitude* to each path. This amplitude can be interpreted as consisting of a probability and a *phase*. Phase is a fundamental property of quantum states that has no classical counterpart. The phase of a state, however, is never directly measured, according to the dictum of quantum mechanics. Relative phase, on the other hand, is measurable indirectly through interference phenomena. Thus, the only aspect of a "quantum language" that is observable is the resulting stochastic language, possibly affected by phase interference. We therefore use the apparatus of stochastic languages to describe the behavior of quantum dynamical systems.

B. States in quantum systems

The concept of state in quantum mechanics is often not as clear as that in classical mechanics. There are, in fact, several distinct notions of state. It is helpful, therefore, first to describe alternative notions of quantum state by drawing a parallel with classical stochastic automata, which we just reviewed. Then we give the definition of a quantum finite-state machine and, finally, return to clarifying the distinctions between various kinds of state.

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. That prediction is determined by a distribution over the internal states, which 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, following the above interpretation, we will take the state of the automaton to be a "distribution" over them which we call the *state vector*. The crucial difference with classical (stochastic) automata is that this distribution over internal states is most generally a superposition of quantum amplitudes (and is not a probability distribution). Thus, internal states potentially interfere, which 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. Under certain circumstances we will speak of a *mixed state*, which is a statistical combination (or distribution) of pure states. In this case, the component pure states are not in superposition with each other, but are assigned classical probability weights. One can imagine, for example, a collection of separate 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 states of a quantum dynamical system. A pure state of the latter is represented with respect to a set of *basis states*. The set of basis states is determined by the observable chosen for the experiment. These basis states span the system's state space, which is a Hilbert space. The correspondence between the basis states of the quantum dynamical system and the internal states of the automaton is straightforward: A state

vector, expressed in the basis set, is the state of the automaton. Thus, we will use the terms *internal states* and *basis states* interchangeably, as well as the terms *state vector* and *state* interchangeably.

C. Measurements

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 and when not 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 linear 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 a linear 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.

V. QUANTUM FINITE-STATE MACHINES

Our basic *quantum machine* (QM) is defined as follows.

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

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

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

Our previous discussion of state leads to the following definition of a QM's internal state 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* $\mathcal{H}$*.*

This 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.

Definition. *A state vector* $\langle\psi| = (c_0, c_1, \ldots, c_{n-1})$ *can be expanded in terms of basis states*

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

with $c_i \in \mathbb{C}$ *and* $\sum_{i=0}^{n-1} c_i^* c_i = 1$*.*

A. Operation

The operation of a QM is described by the evolution of a row vector. (Row and column-vectors are also called bra and ket vectors, respectively.) We make this choice, which is unconventional for notation in quantum mechanics for two reasons. First, classical finite-state machines are described via row vectors. And second, the graphical meaning of a transition from state s_i to s_j is reflected in the transition matrix entries T_{ij}, only if one uses row vectors and left multiplication with T.

We can now describe a QM'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 QM is given by:

$$\begin{aligned} \langle\psi(y|x)| &= \frac{\langle\psi^0| \, T(y|x)}{\sqrt{\langle\psi^0|T(y|x)T^\dagger(y|x)|\psi^0\rangle}} \\ &= \frac{\langle\psi^0| \, U(x)P(y)}{\sqrt{\langle\psi^0|U(x)P(y)U^\dagger(x)|\psi^0\rangle}} \ , \end{aligned} \tag{27}$$

where † is the complex transpose. In the following we drop the renormalization factor in the denominator.

When a QM 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

$$T(y^L|x^L) = U(x_0)P(y_0)U(x_1)P(y_1)\cdots U(x_{L-1})P(y_{L-1}) \tag{28}$$

and the updated state vector is

$$\langle\psi(y^L|x^L)| = \langle\psi^0|\, T(y^L|x^L) \ . \tag{29}$$

B. Observation

Modeling observation explicitly is often taken as one of the hallmarks of quantum mechanics, as compared to classical. Since it will come up repeatedly we need to clarify our use of the term *observation*, especially as it concerns *measured* and *unmeasured* quantum states. Physically, a state is *measured* whenever an observable is chosen and the system is measured with respect to that observable. Mathematically, a state is *measured* whenever a projection operator in the observable's eigenbasis is applied to the state vector. 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.

A QM uses the same notion of measurement. The state of a QM is measured by applying projection operators. A measurement outcome is represented as symbol. If the state is not projected, it is *unmeasured* and no symbol is observed. We denote this case with the null symbol λ. The decision whether to perform a measurement or not should be considered as an input to the QM.

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*

$$P(y) = |\phi_i\rangle\langle\phi_i| \ , \tag{30}$$

where ϕ_i is the eigenvector of the observable with eigenvalue y. In the case of degeneracy $P(y)$ sums over all eigenstates with the same eigenvalue:

$$P(y) = \sum_i |\phi_i\rangle\langle\phi_i| \ , \tag{31}$$

Each P is Hermitian: $P^\dagger = P$ and $P^2 = P$.

In a QM we associate a projection operator with each output symbol $y \in Y$ such that

$$\sum_{y\in Y} P(y) = I \ , \tag{32}$$

where I is the identity matrix. The projection operators $\mathbf{P} = \{P(y) : y \in Y\}$ are mutually orthogonal and span the Hilbert space. 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.

In the simplest setting, one identifies a QM's state with a particular observable so that the QM's internal states become the observable's eigenstates. The operators $P(y)$ each project onto an eigenstate of the observable, which is identified with a particular internal state. The output symbol y labels the eigenvalue of that eigenstate. Here we simply label the eigenvalues with discrete symbols $y \in Y$.

In quantum mechanics, one distinguishes between *complete* and *incomplete* measurements [13]. 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. In our notation 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 QM is potentially in a superposition of states $\sum_i c_i \langle\phi_i|$, where i sums over the degenerate eigenstates. Just as degeneracy leads to interesting consequences in quantum physics, we will see in the examples to follow that degenerate observables lead to interesting quantum languages.

C. Word distributions

The QM state after reading in symbol x and emitting symbol y is given in Eq. (27). Starting the QM in $\langle\psi^0|$ the conditional probability $\Pr(y|x)$ of the output symbol y given the input symbol x is then calculated as:

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

The QM state after reading in word x^L and emitting word y^L is given in Eq. (29). The probability $\Pr(y^L|x^L)$ of output sequence y^L conditioned on input sequence x^L is:

$$\Pr(y^L|x^L) = \langle\psi(y^L|x^L)|\psi(y^L|x^L)\rangle \ . \tag{34}$$

QMs are transducers, mapping inputs to outputs. As such they should be compared to the transducers of Ref. [59]. One difference is that the output symbols are used as orthogonal subspaces for state storage and so only serve as an additional resource for recognition. These transducers are also not required to represent word distributions.

D. Properties

Properties of QMs are related to those of SMs 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 QMs.

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

A unistochastic matrix is doubly stochastic, which follows directly from the properties of unitary matrices. Recall that the stationary distribution of a Markov chain with a doubly stochastic transition matrix is uniform [60].

Focusing on the graph structure of QMs in the following, we revisit the definition of a path between node i and node j when $T_{ij} > 0$. Since a QM's transition matrix generally has complex entries we define a path between node i and node j when $|T_{ij}|^2 > 0$.

Compared to stochastic machines the structure of QMs is constrained through unitarity and these constraints are reflected in a number of symmetries. For example, if there is a path from internal state q_i to state q_j, there is one from q_j to q_i.

Proposition 11. *Every node i of G(QM), if connected to a set of nodes $j \neq i$, is a member of a strongly connected set.*

Proof. *If two nodes, i and j, are connected, then there is a path between them. If we denote the input word along this path x^L, then*

$$|\langle\phi_i|U(x^L) + U^{-1}(x^L)|\phi_j\rangle| > 0 \ . \tag{35}$$

It follows that there must be at least one path between i and j:

$$\begin{aligned} 0 &< |\langle\phi_i|U(x^L) + U^{-1}(x^L)|\phi_j\rangle| \\ &\leq |\langle\phi_i|U(x^L)|\phi_j\rangle| + |\langle\phi_i|U^{-1}(x^L)|\phi_j\rangle| \\ &= |\langle\phi_i|U(x^L)|\phi_j\rangle| + |\langle\phi_j|U(x^L)|\phi_i\rangle| \ . \end{aligned} \tag{36}$$

That is, either

$$|\langle\phi_i|U(x^L)|\phi_j\rangle| > 0 \tag{37}$$

or

$$|\langle\phi_j|U(x^L)|\phi_i\rangle| > 0 \ . \tag{38}$$

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. □

Proposition 11 is a consequence of the properties of unitary matrices.

The following Corollary is related to the structure of the projection operators.

Corollary 4. *The maximum size of the output alphabet Y of a QM is equal to the dimension of the Hilbert space.*

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

Proposition 12. *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.* □

Proposition 13. *A QM'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 $U(x)$:*

$$\sum_{y\in Y} T(y|x) = \sum_{y\in Y} U(x)P(y) = U(x) \ . \tag{39}$$

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

$$P(y) = U^\dagger(x)T(y|x) \ . \tag{40}$$

□

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

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

Proposition 14. *All QMs are reversible.*

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

Graphically, the reversed QM is obtained by simply switching the direction of the edges. This produces a machine with the transition amplitudes T_{ji}, formerly T_{ij}. The original input and output symbols, which labeled ingoing edges to state q_i, are again attached to the edges coming into state q_i, not going out of q_i. Therefore, in general, the languages generated by a QM and its inverse are not the same.

E. Summary

Before we discuss specialized classes of quantum machines we should summarize their common properties. The mathematical representation of a QM 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. The vector representation allows one to calculate word probabilities which, in turn, represent the system's temporal behavior.

An equivalent description of a quantum machine is given by its graphical representation. At any particular point in time the QM is in one or several internal states. During one time step the QM 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. The discussion of unistochastic matrices leads one to conclude that QM graphs constitute a subset of directed graphs, namely the strongly connected ones. Moreover, all incoming edges to a node are labeled with the same output symbol.

VI. QUANTUM RECOGNIZERS AND GENERATORS

A quantum machine is the most general object, describing a quantum dynamical process in terms of inputs and outputs. We will now specialize a quantum machine into recognizers and generators. We do this by following the strategy we adopted for developing classes of stochastic machines. For each machine class we first give a general definition and then specialize, eventually yielding fully deterministic versions. We establish a number of properties for each type and the compare their descriptive powers. The comparison is done in terms of the process languages each class can recognize or generate. The results are collated together as 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 machines—that are, in fact, transducers—we can now introduce quantum finite-state recognizers as restrictions of QMs and compare these with alternative models of quantum recognizers.

Definition. *A* quantum recognizer *(QR) is a QM with a set of transition matrices* $\{T(x) = U(x)P(\lambda)\}$*, output alphabet* $Y = \{accept, reject\}$*, and a start state* $\langle\psi^0|$*. A QR assigns the number* $\left|\langle\psi^0|T(w)P(accept)T(w)|\psi^0\rangle\right|$ *to a word* w*.*

Definition. *Given a probability distribution* $\Pr(w)$*, a QR* accepts*, with word-probability threshold* $0 \leq \delta \leq 1$*, a word* $w = x_0x_1 \ldots x_{L-1}$*,* $x \in X$*, if, on reading each symbol* x_t *it*

1. *follows only allowed transitions* $|T_{ij}(x_t)|^2 > 0$ *and*

2. *assigns a word probability within* δ*:*

$$\left|\Pr(w) - |\langle\psi^0|T(w)P(accept)T(w)|\psi^0\rangle|\right| \leq \delta \ . \quad (41)$$

As with Rs, the set of output symbols consists of only the symbols "accept" and "reject".

An example of a nonregular process language $\mathcal{L}$ is $\{w \in \mathcal{L} : w = 0^m1^n,\ m \neq n,\ \Pr(w) = sin^2(\pi k\theta), k = \#_0(w) - \#_1(w), \theta = (\sqrt{5}-1)/2\}$. $\mathcal{L}$ contains all the words in which contiguous 0- and 1-blocks have an unequal number of 0s and 1s. (This example extends the nonregular formal language in [61] to a nonregular process language.) An example of a QR that recognizes $\mathcal{L}$ with $\delta = 0$ is given in Fig. 9.

Classically, recognizing the support of this language requires a pushdown automaton—an infinite-state device that is more powerful than any finite-state machine [45]. A pushdown automaton employs memory organized as a stack—the last element stored is the first retrieved. In this case, as the pushdown automaton reads in symbols, it uses the stack to balance the number of 0s and 1s by pushing a token onto the stack's top for each 0 read and by popping a token off the stack top, if a 1 is read. If a 0 is read after a 1 when there is still a token on the stack, the input is accepted and the stack is reset. If a 1 is read in and the stack is empty, the input is also accepted. Otherwise it is rejected.

A QR that recognizes the support and the probability distribution of $\mathcal{L}$ consists of only two states. The transition matrices are given by

$$\begin{aligned} T(00) &= U(0) \ , \\ T(01) &= U(0) \ , \\ T(10) &= U(1)P(y) \ , \\ T(11) &= U(1) \ , \end{aligned} \quad (42)$$

with

$$\begin{aligned} U(0) &= \begin{pmatrix} \cos(\pi\theta) & -\sin(\pi\theta) \\ \sin(\pi\theta) & \cos(\pi\theta) \end{pmatrix} \\ U(1) &= \begin{pmatrix} \cos(\pi\theta) & \sin(\pi\theta) \\ -\sin(\pi\theta) & \cos(\pi\theta) \end{pmatrix} \ , \end{aligned} \quad (43)$$

where $\theta = (\sqrt{5}-1)/2$ is the irrational above. Note, that $U(1)$ is the inverse of $U(0)$. The start vector is $\langle\psi^0| = \langle 1, 0|$ and the projection operators $P(y)$ are

$$P(\text{accept}) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } P(\text{reject}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \ . \quad (44)$$

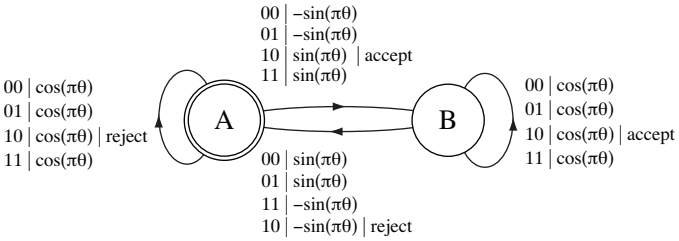

FIG. 9: A quantum finite-state recognizer that accepts a nonregular language: words with contiguous 0- and 1-blocks of unequal length. Classically, this language requires a pushdown (infinite-state) automaton. Edge notation: $x|p$, $x \in X$ and $p = T_{ij} \in \mathbb{C}$. The start state is indicated with an inscribed circle.

The QR reads in a string by sliding a two-symbol window over the string. It operates by rotating the state vector by an irrational angle $\pi\theta$ for each $1x$ read and $-\pi\theta$ for each $0x$ read, where the x is the second symbol in the two-symbol window. Every time a 10 is read in the state vector is projected. According to Eq. (41), for $\delta = 1$ the support is recognized. For $\delta = 0$ the probability distribution is recognized in addition.

Every time the state vector is projected it is reset to $\langle\psi_0|$. The machine continues to read in symbols. At the end the string is accepted if all outputs are "accept". If one or more output symbols are "reject", the string is rejected.

Where the classical automaton required a stack to store the number of 0s and 1s being read in, the QR stores this information in the state vector itself; namely in the state vector's phase. No auxiliary storage device, like a stack, is needed to augment the finite-state control.

This example serves to illustrate how quantum devices can be more powerful than their classical counterparts. By way of comparison, the state of a stochastic automaton does not have phase and thus cannot recognize this language.

This QR is nondeterministic. As before, though, we also have deterministic versions of QRs.

Definition. *A* quantum deterministic recognizer *(QDR) is a quantum recognizer with a designated start state $q_0 \in Q$ and transition matrices $T(x)$ that have at most one nonzero element per row.*

B. Alternative quantum recognizers

The recent literature has proposed several constructions for quantum finite-state recognizers—recognizers that accept formal languages (the support of a stochastic language). One is a 1-way automaton that reads symbols once and from left to right (say) in the input word [62, 63]. Another is a 2-way automata that reads the input word many times moving either left to right or right to left [63]. The model by Moore and Crutchfield [62] includes one-time measurement at the end of the input word to determine acceptance. In contrast, the automata of Kondacs and Watrous [63] allow for measurements that check for acceptance, rejection, or continuation at every time step. Moore and Crutchfield introduced a generalized quantum finite-state automaton whose transition matrices need not be unitary. A third quantum finite-state automaton model was introduced by Nayak [64] that allows for any orthogonal measurement as a valid intermediate computational step.

These three alternatives appear to be the most widely discussed. There are others, however, and so the above list is by no means complete. Generally, though, these constructions do not serve our needs, as outlined in the introduction, as models of quantum processes.

C. Quantum generators

We now introduce quantum finite-state generators as restrictions of QMs and as a complement to recognizers. In contrast to recognizing quantum finite-state machines, purely generating quantum finite-state machines appear to not have been discussed before. A quantum generator is an input-independent QM with its own inner clock. At each step it makes a transition from one state to another and emits a symbol. In this section, we introduce quantum generators and then discuss their properties. In the next section, we illustrate the concepts with examples.

Similar to how a G serves as a computational model for a classical dynamical system, quantum generators are computational models for quantum dynamical systems. The generated language describes the quantum system's behavior. We will illustrate these points with a number of examples in the following section.

Definition. *A* quantum generator *(QG) is a QM with a set of transition matrices $T(y) = UP(y)$.*

As in the classical case there are nondeterministic (just defined) and deterministic QGs.

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

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

Definition. *A* complete quantum generator *(CQG) is a QG observed via complete measurements.*

Proposition 15. *CQGs 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 $\mathcal{L}(CQG) \subset \mathcal{L}(QDG)$.

Definition. *Given a QDG $\mathcal{M} = \{U, P(y)\}$, the* equivalent *(classical) DG $\mathcal{M}' = \{T(y)\}$ 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.

D. Properties

In the following we derive a number of properties of QGs and QDGs. We will use these to characterize the kinds of stochastic languages they can generate. Before we do so, we need to review the density matrix formalism, found in any standard text book on quantum mechanics. It will be very useful in describing the *state* of a QG. We refer the reader back to Section IV B for an overview of the various notions of state. Most importantly, the state of a QM is a *pure state* at any time. However, as soon as one compares QM's states over time one must refer to *mixed states*. The main difference with common usage of "mixed state" is that we compare the same state *over time*; whereas, usually different *systems* are compared at one single point in time. Nevertheless, in both cases, the density matrix formalism is useful.

Let a system be described by a state vector $\langle\psi_i|$ at time t. If we don't 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 real state of the system 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 $\langle\psi_i|$:

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{45}$$

Note that the $\langle\psi_i|$ are pure states, but not necessarily basis states $\langle\phi|$.

With this notation at hand, we can now formulate and prove a number of properties of QGs and QDGs.

Definition. *The* stationary state $\langle\phi^s|$ *of a QM is the mixed state which is invariant under unitary evolution. That is,*

$$\rho^s = U^\dagger \rho^s U \ . \tag{46}$$

Theorem 1. *The stationary state of a deterministic QM is the following maximally mixed state:*

$$\rho^s = |Q|^{-1} \sum_{i=0}^{n-1} \left|\phi_i\right\rangle \left\langle\phi_i\right| \ . \tag{47}$$

Note that, since the $\langle\phi_i|$ are basis states, ρ^s is a diagonal matrix, whose diagonal components form $\langle\phi^s|$.

Proof. *To prove the theorem we repeatedly use the completeness relation:*

$$\sum_{i=0}^{n-1} \left|i\right\rangle \left\langle i\right| = I \ , \tag{48}$$

where $\langle i|$ is shorthand for basis state $\langle\phi_i|$. Any matrix M can then be written as

$$\sum_{i,j} \left|i\right\rangle \left\langle j\right| M_{ij} \ . \tag{49}$$

Since the QM is deterministic its state at any point in time is a single internal state. The QM's average state can then expressed as the density matrix over basis states:

$$\rho = \sum_i p_i \left|i\right\rangle \left\langle i\right| , \tag{50}$$

which is a diagonal matrix. We now show that the coefficients of ρ^s are $p_i = |Q|^{-1}$:

$$\begin{aligned} U^\dagger \rho^s U &= \sum_{i,j} \left|i\right\rangle \left\langle j\right| U^\dagger_{ij} \cdot \sum_k p_k \left|k\right\rangle \left\langle k\right| \cdot \sum_{l,m} \left|l\right\rangle \left\langle m\right| U_{lm} \\ &= \sum_{i,j} \left|i\right\rangle \left\langle j\right| U^\dagger_{ij} \cdot \sum_{k,m} p_k \left|k\right\rangle \left\langle m\right| U_{km} \\ &= \sum_{i,k,m} \left|i\right\rangle \left\langle m\right| U^\dagger_{ik} p_k U_{km} \end{aligned} \tag{51}$$

Setting the coefficients to $p_i = |Q|^{-1}$ we obtain for the righthand side:

$$\begin{aligned} U^\dagger \rho^s U &= |Q|^{-1} \sum_{i,m} \left|i\right\rangle \left\langle m\right| \left(\sum_k U^*_{ki} U_{km} \right) \\ &= |Q|^{-1} \sum_{i,m} \left|i\right\rangle \left\langle m\right| \delta_{im} \\ &= |Q|^{-1} \sum_i \left|i\right\rangle \left\langle i\right| = \rho^s \ , \end{aligned} \tag{52}$$

where δ_{im} is the Kronecker symbol and the second to last row follows from the orthogonality of the rows of unitary matrices. □

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:

$$\begin{aligned} \Pr(y) &= \mathrm{Tr}\left(T^\dagger(y) \rho^s T(y)\right) \\ &= \mathrm{Tr}\left(P^\dagger(y) U^\dagger \rho^s U P^\dagger(y)\right) \\ &= \mathrm{Tr}\left(\rho^s P(y)\right) \ , \end{aligned} \tag{53}$$

where Tr is the trace. Similarly, the asymptotic word probabilities $\Pr(y^L)$ are:

$$\Pr(y^L) = \mathrm{Tr}\left(T^\dagger(y^L) \rho^s T(y^L)\right) \ . \tag{54}$$

No further simplification is possible for the general case.

Equation (53), however, can be further simplified for single symbol probabilities. As a result we find a concise expression for single-symbol probabilities of QGs.

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

Proof. *Eq. (53) simplifies as follows:*

$$\begin{aligned}\Pr(y) &= \mathrm{Tr}(\rho^s P(y)) \\ &= |Q|^{-1}\dim P(y) \end{aligned} \quad (55)$$

□

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

E. Recognition and generation, quantum and classical

To better appreciate what these machines are capable of in this section we amortize the effort in developing the preceding results to describe the similarities and differences between quantum recognizers and generators, as well as between classical stochastic and quantum automata. We collect the results, give a summary and some interpretation, and present a road map (Fig. 10) that lays out the computational hierarchy of finitary quantum processes. As above, when we refer to $\mathcal{L}(M)$ we mean the language or set of languages produced by a machine in class M.

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

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

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

The word probabilities $\Pr_{\mathcal{M}}(w)$ *for $\mathcal{M}$ are calculated using Eq. (54) and the QDG's transition matrices $T_{\mathcal{M}}$:*

$$\begin{aligned}\Pr_{\mathcal{M}}(w) &= \mathrm{Tr}\left(T^\dagger_{\mathcal{M}}(w)\rho^s T_{\mathcal{M}}(w)\right) \\ &= \sum_i \langle i| T^\dagger_{\mathcal{M}}(w)\rho^s T_{\mathcal{M}}(w) |i\rangle \\ &= |Q|^{-1}\sum_i \langle i| \sum_{j,k} |j\rangle \langle k| T^\dagger_{jk,\mathcal{M}}(w) \sum_l |l\rangle \langle l| \\ &\quad \cdot \sum_{m,n} |m\rangle \langle n| T_{mn,\mathcal{M}}(w) \ . \end{aligned} \quad (56)$$

Due to the orthogonality of the basis states this simplifies to

$$\begin{aligned}\Pr_{\mathcal{M}}(y^L) &= |Q|^{-1}\sum_{i,j} T^\dagger_{ji,\mathcal{M}}(y^L) T_{ij,\mathcal{M}}(y^L) \\ &= |Q|^{-1}\sum_{i,j} T^*_{ij,\mathcal{M}}(y^L) T_{ij,\mathcal{M}}(y^L) \ . \end{aligned} \quad (57)$$

The word probabilities $\Pr_{M'}(w)$ *for M' are calculated using Eq. (10) and the DG's transition matrices $T_{\mathcal{M}}$:*

$$\begin{aligned}\Pr_{\mathcal{M}'}(y^L) &= \langle \pi^0 | T_{\mathcal{M}'}(y^L) | \eta \rangle \\ &= \sum_{i=0}^{n-1} \left(\pi^0_i \sum_j (T_{\mathcal{M}'}(y^L))_{ij} \right) \\ &= |Q|^{-1} \sum_{i,j=0}^{n-1} (T_{\mathcal{M}'}(y^L))_{ij} \ . \end{aligned} \quad (58)$$

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

A given DG can be observationally equivalent to more than one QDG. 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 5. *For every QDG,* supp *$\mathcal{L}(QDG)$ is a regular language.*

Proof. *This follows directly from Thm. 3 and Prop. 4.* □

Corollary 6. *For every QDG, $\mathcal{L}(QDG)$ is a process language.*

Proof. *This follows directly from Thm. 3 and Prop. 5.*

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

Proposition 16. *For every $\mathcal{L}(QDG)$ there is a QDR recognizing it: $\mathcal{L}(QDG) \subseteq \mathcal{L}(QDR)$.*

Proof. *Let QDG generate the process language $\mathcal{L}$. Consider QDG's transition matrices $T(y)$ and form a new set $T(x)$ in which $X = Y$. The $T(x)$ define a deterministic recognizer accepting $\mathcal{L}$.* □

Proposition 17. *For every $\mathcal{L}(QDR)$ there is a QDG generating it: $\mathcal{L}(QDR) \subseteq \mathcal{L}(QDG)$.*

Proof. *Let QDR accept the process language $\mathcal{L}$. Consider QDR's transition matrices $T(x)$ and form a new set $T(y)$ in which $Y = X$. The $T(y)$ define a DG generating $\mathcal{L}$.* □

Corollary 7. *$\mathcal{L}(QDR) = \mathcal{L}(QDG)$.*

Proof. *From Props. 16 and 17.*

Note that determinism is crucial for the above results.

Proposition 18. *There exists a QR such that* supp $\mathcal{L}(QR)$ *is not a regular language.*

Proof. *By example: The* supp $(\mathcal{L})$ *of the QR in Fig. 9 is a nonregular language.*

Corollary 8. *There exists a QR such that $\mathcal{L}(QR)$ is not recognized by any SR.*

Proof. *This follows from Cor. 2 and Prop. 18.* □

Corollary 9. *There exists a QR such that $\mathcal{L}(QR)$ is not generated by any QDG.*

Proof. *This follows from Cor. 5 and Prop. 18.* □

Corollary 10. *For every $\mathcal{L}(QDR)$ there is a DG generating it.*

Proof. *This follows from Prop. 17 and Thm. 3.* □

Proposition 19. *There exists a DG such that $\mathcal{L}(DG)$ is not generated by any QDG.*

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

$$T(0) = \left(\tfrac{1}{\sqrt{2}}\right) \text{ and } T(1) = \left(1 - \tfrac{1}{\sqrt{2}}\right) \tag{59}$$

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

$$\Pr(y) = \frac{\dim P(y)}{n} \ , \tag{60}$$

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

Corollary 11. $\mathcal{L}(QDG) \subset \mathcal{L}(DG)$.

Proof. *From Thm. 3 and Prop. 19.*

Corollary 12. $\mathcal{L}(QDR) \subset \mathcal{L}(SDR)$*: There exists a SDR such that $\mathcal{L}(SDR)$ is not recognized by any QDR. For every $\mathcal{L}(QDR)$ there is a SDR recognizing it.*

Proof. *From Cor. 7, Prop. 19, and Thm. 3.* □

Proposition 20. *There exists a SR such that $\mathcal{L}(SR)$ is not recognized by any QR.*

Proof. *Theorem 7 in Ref. [62].* □

At this point it is instructive to graphically summarize the relations between recognizer and generator classes. Figure 10 shows a machine hierarchy in terms of languages recognized or generated. The class of CQGs 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—DGs and SDRs. These, in turn, are included in the set of languages recognized or generated by classical nondeterministic machines, Gs and SRs, as well as QRs and QGs.

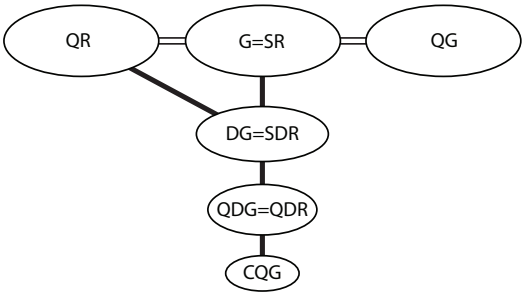

FIG. 10: Finitary process language hierarchy: Each circle represents the set of process languages recognized or generated by the inscribed machine class. Increasing height indicates proper containment; machine classes at the same height are not linearly comparable. The hierarchy summarizes the theorems, propositions, and corollaries in Secs. III G and VI E.

F. Conjectures

Notably, the relative capabilities of the quantum nondeterministic generators (QGs) and recognizers (QRs) vis a vis their stochastic counterparts, which are equivalent, have not been fully delineated.

For example, investigating several QGs revealed that the word distributions they generate depend significantly on the initial state vector. This makes their characterization markedly more difficult than for QDGs. Additionally, the relation between QGs and QRs is unclear. On the one hand, QRs can recognize stochastic languages with nonregular support and they are not measured until they accept or reject inputs. In contrast, QGs can be measured at varying times during their operation according to the measurement protocol. One might be led, therefore, to think that QRs contain QGs as a subclass. On the other hand, the simple proof methods that showed that $G = SR$, would suggest otherwise. Given these competing and incomplete pictures, we close with several conjectures that summarize the open remaining issues. Only two of these require proof, as the others follow as noted.

Conjecture 1. *QRs and QGs are equivalent: For every $\mathcal{L}(QG)$ there is a QR recognizing it and for every $\mathcal{L}(QR)$ there is a QG generating it.*

Conjecture 2. *Finite quantum machines are incomparable to finite stochastic machines.*

This would follow from the preceding corollary and Prop. 20.

Finally, important comparisons at a lower level remain open.

Conjecture 3. *There exists a QG such that $\mathcal{L}(QG)$ is not recognized by any SDR.*

Conjecture 4. *There exists a QG such that $\mathcal{L}(QG)$ is not recognized by any QDR.*

This would follow from the immediately preceding corollary and Cor. 7.

The preceding results and conjectures serve only to indicate how the finitary process hierarchy is organized. Analyzing how varying the acceptance threshold δ and how using differences between word distributions modifies the hierarchy awaits further investigation.

We now turn to the effects of measurement protocol and find a perhaps unexpected richness.

G. Language diversity

Let's review the concept of *language diversity*, introduced in Ref. [65], which adds an important component to the comparison of language classes as well as to the computation-theoretic description of quantum processes. Language diversity shows how QGs can be more powerful than their classical counterparts (Gs)—a comparison that the language hierarchy of Fig. 10 does not address.

The notion of a measurement protocol is familiar from quantum mechanics: We define the *measurement period* as the number of unitary evolution time steps relative to the application of a projection operator. For a classical system this is less familiar, but it can be used in the same way. The measurement period in that case is the number of internal state transitions relative to observing an output symbol. The internal dynamics remain unaltered in the classical case, whether the system is measured or not. In the quantum case, as we have emphasized, the situation is quite different. Applying a projection operator disturbs the internal dynamics.

Definition. *[65] A process observed with* measurement period p *is measured every* p *time steps.*

This model of a measurement protocol, subsampling the output time series, reminds one of von Mises' version of probability theory based on "collectives" [66].

The process language generated by quantum finite-state generator $\mathcal{M}$ for measurement period p is labeled $\mathcal{L}^p(\mathcal{M})$. The resulting evolution equation for the state vector is:

$$\langle \psi_{t+p}| = \langle \psi_t| U^p P(y) \ . \tag{61}$$

Consider now the set $\{\mathcal{L}^p(\mathcal{M})\}$ of languages generated by $\mathcal{M}$ when varying measurement period. The following is another central result, connecting language theory and quantum processes.

Proposition 21. *There is a QG such that* $\{\mathcal{L}^p(QG)\}$ *obtained by allowing unmeasured steps cannot be generated by a G.* □

Proof. *Consider the example of a QR in Sec. VI A where the* $U(x)$ *are rotation matrices about angle* $\theta = \sqrt{2}$. *Transforming this into a QG,* $\{\mathcal{L}^p(QG)\}$ *is the set of biased coin processes for all possible biases, which contains an infinite number of languages.* □

The above result shows that the process language hierarchy changes as a function of measurement protocol. For more details, see Ref. [65]. This result adds an additional motivation for focusing on measured quantum dynamical systems and their capacity for intrinsic computation.

VII. QUANTUM GENERATORS AND DYNAMICAL SYSTEMS: 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 QM) with input and output.

A. Two-state quantum processes

According to Thm. 2 the symbol distribution generated by a QDG only depends on the dimension of the projection operator and the dimension of the Hilbert space. What are the consequences for two-state QDGs? First of all, according to Cor. 4 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* DG. Its state-to-state transition matrix is given by

$$T = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix} \ . \tag{62}$$

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 physical two-state processes—various quantum dynamical systems—and build quantum generators for them.

The *iterated beam splitter* is an example that, despite its simplicity, makes a close connection with real experiment. Figure 11 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

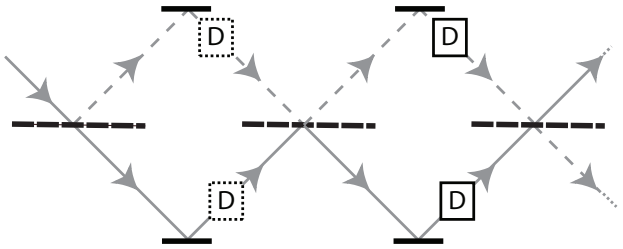

FIG. 11: Experimental set-ups for the iterated beam splitter: Solid lines are mirrors; beam splitters, horizontal dashed lines. Photon detectors, marked as D, are placed between every pair of beam splitters. Under measurement protocol I all detectors are in operation; under protocol II only the solid-line detectors are activated. The apparatus is repeated indefinitely to the right.

apparatus is repeated indefinitely to the right. After the second beam-splitter there is a third and a fourth and so on. Nondestructive 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:

$$U_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} . \tag{63}$$

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

$$P(0) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} , \tag{64}$$

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. (63)-(64) and the stationary state. 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 transition matrices are:

$$T(0) = U_H P(0) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} , \tag{65a}$$

$$T(1) = U_H P(1) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} . \tag{65b}$$

The resulting QDG is shown in Fig. 12. What we previously called protocols I and II correspond to, following the notation introduced in Sec. VI G, measurement protocols with $p = 1$ and $p = 2$, respectively.

The word distribution for the process languages generated by protocols I and II are obtained from Eq. (54). Word probabilities for protocol I (measurement at each time step) are, to give some examples:

$$\Pr(0) = |Q|^{-1} \dim(P(0)) = \frac{1}{2} , \tag{66a}$$

$$\Pr(1) = |Q|^{-1} \dim(P(1)) = \frac{1}{2} , \tag{66b}$$

$$\Pr(00) = \mathrm{Tr}\left(T^\dagger(0) T^\dagger(0) \rho^s T(0) T(0)\right) = \frac{1}{4} , \tag{66c}$$

$$\Pr(01) = \Pr(10) = \Pr(11) = \frac{1}{4} . \tag{66d}$$

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)

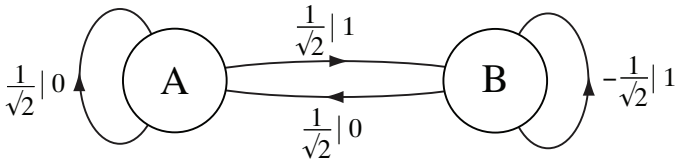

FIG. 12: Quantum finite-state machine for the iterated beam splitter: The resulting symbol sequences are statistically identical to the measurement sequences obtained with the measurement protocols I and II shown in Fig. 11. When no measurement is made, transitions along all edges occur.

we find:

$$\Pr(0) = \mathrm{Tr}\left(T^\dagger(0\lambda)\rho^s T(\lambda 0)\right) = \frac{1}{2}\ , \tag{67a}$$

$$\Pr(1) = \mathrm{Tr}\left(T^\dagger(1\lambda)\rho^s T(\lambda 1)\right) = \frac{1}{2}\ , \tag{67b}$$

$$\Pr(00) = \mathrm{Tr}\left(T^\dagger(0\lambda 0\lambda)\rho^s T(\lambda 0\lambda 0)\right) = \frac{1}{2}\ , \tag{67c}$$

$$\Pr(11) = \mathrm{Tr}\left(T^\dagger(1\lambda 1\lambda)\rho^s T(\lambda 1\lambda 1)\right) = \frac{1}{2}\ , \tag{67d}$$

$$\Pr(10) = \Pr(01) = 0\ . \tag{67e}$$

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 DG reveals several crucial differences in performance. Following the recipe from Sec. VI E, on how to build a DG from a QDG, gives the classical generator shown in Fig. 13(a). Its transition matrices are:

$$T(0) = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \text{ and } T(1) = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}\ . \tag{68}$$

The measurement sequence generated by this DG for protocol I is the uniform distribution for all lengths, as can be easily verified using Eq. (10) or, since it is deterministic, Eq. (14). 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 DG has to be constructed from the QG's transition matrices for set-up II. This is done by carrying out the matrix product first and then forming its equivalent DG. The result is shown Fig. 13(b). Its transition matrices are:

$$T(0) = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } T(1) = \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\ . \tag{69}$$

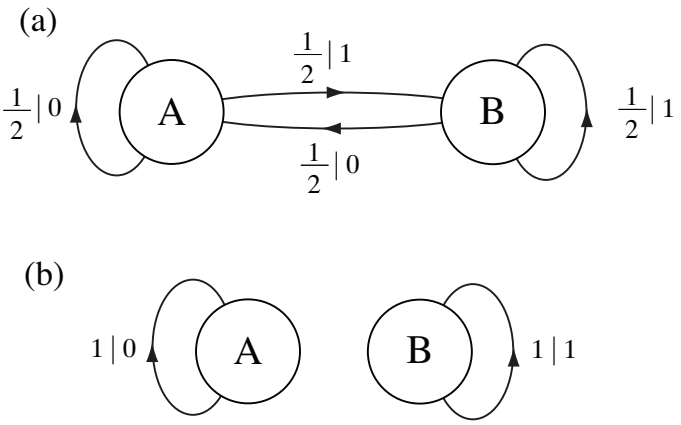

FIG. 13: Classical deterministic generators for the iterated beam splitter: (a) Protocol I, measurement period $p = 1$ and (b) protocol II, $p = 2$. (Cf. Fig. 11.)

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

2. *Quantum kicked top*

The periodically kicked top is a generic model of a finite-dimensional quantum system whose classical limit exhibits various degrees of chaotic behavior depending on its control parameter values. The classical limit is approached as spin $j \to \infty$ [67]. The quantum kicked top is dynamically interesting since it is the quantum analog of the iterated *twist and turn map* [36].

Here we construct a QDG for the quantum kicked top in its simplest form, with spin $j = \frac{1}{2}$, and then investigate the resulting process languages. The results illustrate where QMs are useful in investigating quantum dynamical systems and especially in the latter's transition to classical dynamics.

The periodically kicked top is a spin-j system evolving in a $2j + 1$-dimensional Hilbert space. It is exposed to a constant magnetic field in the y-direction and periodically twisted around the z-axis due to a kick caused by an orthogonal magnetic field which has the form of a δ-function in time. The unitary time evolution operator for the Schrödinger equation is given by

$$U = e^{-ikJ_z^2/2j} e^{-i\pi J_y/2}\ , \tag{70}$$

where $\pi/2$ is the angle of rotation around the y-axis between two twists, the magnitude of the magnetic field causing the twist is controlled by k, and J_y and J_z are the angular momentum operators.

p	supp $(\mathcal{L}^p)$	$\mathcal{L}^p$
$1,3$	$(0+1)^*$	$\Pr(y^L) = 2^{-L}$
2	sub$((01)^*)$	$\Pr(((01)^*)^L) = 1/2$
		$\Pr(((10)^*)^L) = 1/2$
	sub$((10)^*)$	$\Pr(((10)^*)^L) = 1/2$
		$\Pr(((01)^*)^L) = 1/2$
4	0^*	$\Pr(y^L) = 1$
	1^*	$\Pr(y^L) = 1$

TABLE I: Spin-1/2 quantum kicked top process languages $\mathcal{L}^p$: Calculated using the QDG (see text) for various measurement periods $p = 1, 2, 3, 4$. The word distribution is given for all $y^L \in \mathcal{L}^p$.

For the case of a spin-1/2 system, such as an electron, we obtain the following matrix for U in the basis $\{|\uparrow_z\rangle, |\downarrow_z\rangle\}$ for the z-direction:

$$U = \begin{pmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{pmatrix} \begin{pmatrix} e^{-ik} & 0 \\ 0 & e^{-ik} \end{pmatrix} . \quad (71)$$

The output alphabet $Y = \{0, 1\}$ represents *spin-up* (0) and *spin-down* (1), respectively. The projection operators are given by:

$$P(0) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} . \quad (72)$$

And, finally, the transition matrices are:

$$T(0) = UP(0) \text{ and } T(1) = UP(1) . \quad (73)$$

The first result is that the kicking strength k does not change the generated process language. This is clear from the transition matrices, since k merely adds a global phase and this does not have an observable effect.

The second result is that, as one varies the measurement period p, a number of distinct process languages $\mathcal{L}^p$ are generated. It turns out, though, that any combination of measurement periods generates a process language that is a linear combination of only three different process languages. Choosing the first four measurement periods $p = 1, 2, 3$ and 4 at which these languages are seen, three distinct behaviors are observed. These are given in Table I. The word distributions for these process languages are shown in Fig. 14. $\mathcal{L}^1$ is a random sequence of 0s and 1s. $\mathcal{L}^2$ is a period-2 sequence of alternating 0s and 1s. $\mathcal{L}^3$ is again a random sequence. And $\mathcal{L}^4$ is a sequence of all 0s or all 1s.

Translating this into a more physical language we obtain the following behavior. The spin-1/2 particle is rotating around the y-axis. Each time step corresponds to a $\pi/2$ rotation. After one, two, three, or four time steps its spin in the z-direction is measured. The three process languages originate in the $\frac{\pi}{2}$ rotation *mod* 2π. Four unmeasured time steps correspond to a 2π rotation after which the behavior repeats. The δ-kick by the magnetic field simply adds a global phase to the spin components, which does not affect the measurement outcome.

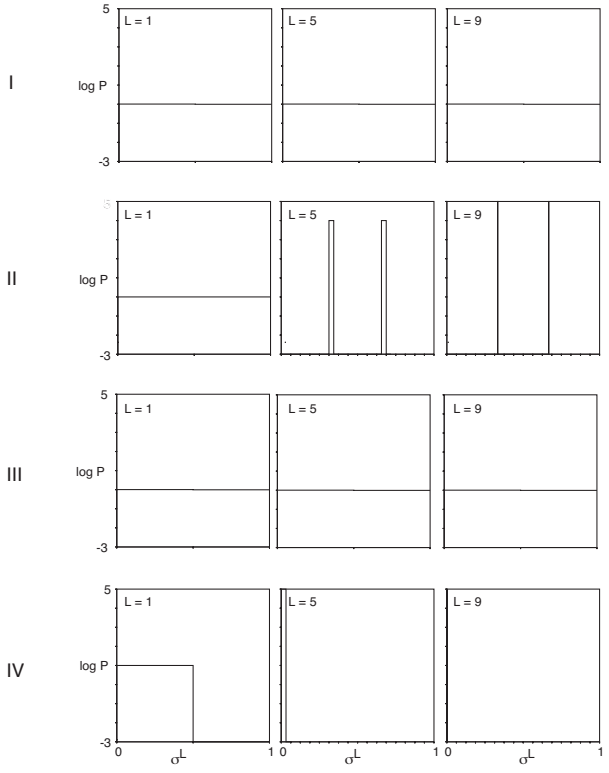

FIG. 14: The process languages $\mathcal{L}^p$ of the quantum kicked top for measurement protocols I ($p = 1$), II ($p = 2$), III ($p = 3$), and IV ($p = 4$).

The number of process languages generated is a characteristic of the types of behavior the quantum dynamical system can exhibit. It should be emphasized that the common way to analyze the quantum kicked top's behavior is in terms of an ensemble, where expectation values of spin observables are calculated, using density operators. The approach evolves the spin ensemble for a number of time steps. At the end, the expectation values are calculated and interpreted. In effect, this approach assumes quantum behavior should be modeled statistically.

Here we took a different approach by following an individual spin system, measuring it repeatedly, and so monitoring its dynamics under observation. The result is a realistic physical situation in which the system is perturbed by its environment via measurement acts.

B. Three-state quantum processes

1. Golden mean quantum machine

Recall the classical Golden Mean generator of Fig. 6. A QDG, which generates the same process language, is shown in Fig. 15. The Golden Mean QDG has three states compared to its classical counterpart, which has

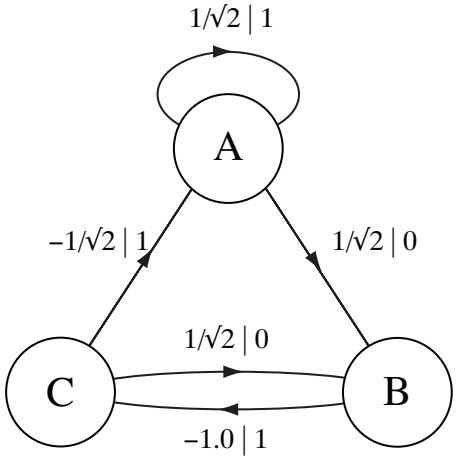

FIG. 15: Quantum generator for the Golden Mean Process.

only two. It has the following matrix representation. The unitary evolution is given by:

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & -1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} . \tag{74}$$

The system is observed with measurement operators:

$$P(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \tag{75}$$

The transition matrices $T(y)$ are then

$$T(0) = UP(0) = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} , \tag{76a}$$

$$T(1) = UP(1) = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & -1 \\ -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix} . \tag{76b}$$

To illustrate that this QDG produces the Golden Mean word distribution we show how to calculate several of the word probabilities using Thm. 2 and Eq. (54):

$$\Pr(0) = |Q|^{-1}\dim(P(0)) = \frac{1}{3} , \tag{77a}$$

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

$$\Pr(011) = \mathrm{Tr}\left(T^\dagger(011)\rho^s T(011)\right) = \frac{1}{6}. \tag{77b}$$

To illustrate the effect of different measurement protocols, cf. Prop. 21, we investigate the generated process language for less frequent measurements, choosing $p = 5$. We find the process language shown in Fig. 16. The word distribution is obviously very different from the one (recall Fig. 1) obtained with measurement period $p = 1$.

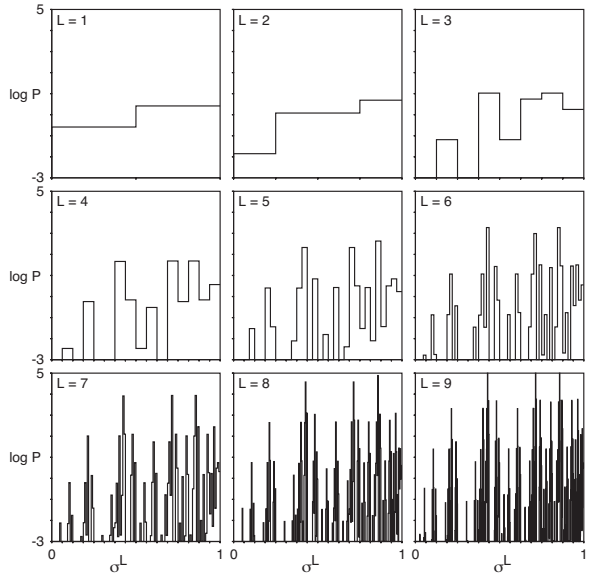

FIG. 16: Quantum Golden Mean QDG observed with a measurement period of $p = 5$. The state vector is measured after 5 steps of unitary evolution. The generated word distributions, although based on the same unitary evolution, differs substantially from those obtained when measuring at each time step ($p = 1$); cf. Fig. 1.

2. *Quantum even process*

The next example is a quantum representation of the Even Process. The QDG is shown in Fig. 17 and has the following matrix representation. The unitary evolution is given by:

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & -1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} , \tag{78}$$

The measurements by:

$$P(0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \tag{79}$$

The word distributions for lengths up to $L = 9$ are shown in Fig. 18.

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 QDGs 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

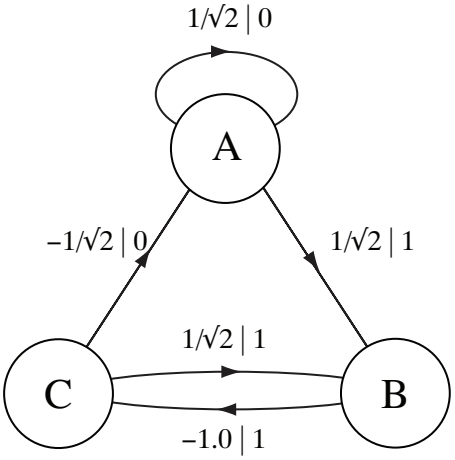

FIG. 17: Quantum generator for the Even Process.

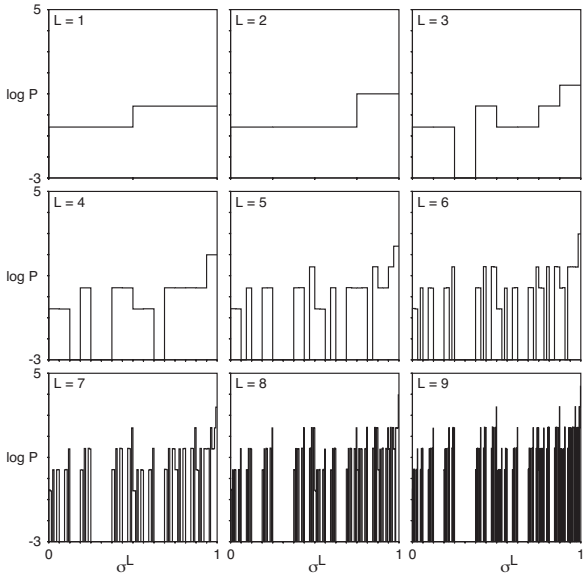

FIG. 18: Process language of the Even QDG observed with measurement period $p = 1$.

subspaces of the Hilbert space: one subspace is two-dimensional and the other, one-dimensional. This yields three QDGs that are distinct up to symbol exchange ($0 \leftrightarrow 1$). For the unitary matrix that generates the Golden Mean and the Even Process (Eq. (78)) two of the three QDGs are equivalent: Row 1 and row 3 of U only differ in phase which has no effect on the language. Thus, the 3-state QDGs defined through U in Eq. (78) are precisely the ones generating the Golden Mean and the Even process languages, respectively.

This very limited number of possible QDGs for any given unitary matrix is yet another indication of the limitations of QMs. Classical Gs do not have the same structural restrictions, since they are not bound by orthogonal partitioning into subspaces, for example. The saving grace for QMs 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 language diversity of QMs.

C. Four-state quantum process

We are now in the position to explore the full capabilities of QMs, turning from generators to transducers—QMs with input as well as output. 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. [18]. 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 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. 19.

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 subsets of these two; see Ref. [18]. The operators are named U_a, U_b, and U_c; matrix representations are given shortly. 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

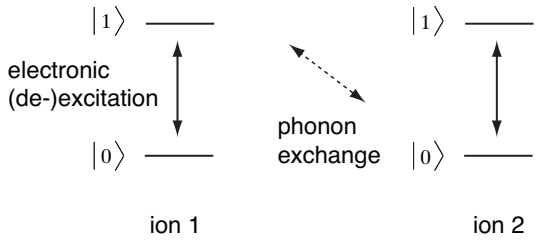

FIG. 19: Schematic view of two vibrationally-coupled trapped ions undergoing electronic excitation. Only the two electronic levels of interest are drawn.

basis states spanning the Hilbert space are given by:

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

The three unitary operations in matrix form are:

$$U_a = H \otimes H = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \ , \tag{80a}$$

$$U_b = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \ , \tag{80b}$$

$$U_c = H \otimes I = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \ . \tag{80c}$$

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

$$P(0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } P(1) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \ . \tag{81}$$

The QM 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 QM's graph is shown in Fig. 20.

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

- Measurement protocol I ($p = 1$): Measure ion 1 after each unitary operation. The resulting state vector evolution is:

$$\langle\psi_{t+1}| = \langle\psi_t| U_a P(y) \ , \tag{82a}$$
$$\langle\psi_{t+2}| = \langle\psi_{t+1}| U_b P(y) \ , \tag{82b}$$
$$\langle\psi_{t+3}| = \langle\psi_{t+2}| U_c P(y) \ . \tag{82c}$$

- Measurement protocol II ($p = 3$): Measure ion 1 only after three unitary operations. This leads to evolution according to

$$\langle\psi_{t+3}| = \langle\psi_t| U_a U_b U_c P(y) \ . \tag{83}$$

The probability distributions of the observed sequences are shown in Figs. 21 and 22. 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 $\mathcal{L}^{\text{II}}$ 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.

2. *Deutsch algorithm as a special case*

It turns out that the trapped-ion experiment implements a quantum algorithm first introduced by Deutsch [58]. 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 [68] showed that a quantum computer can determine the true statement, either A or B, after only two invocations of the operation U, whereas a classical computer requires $N+1$

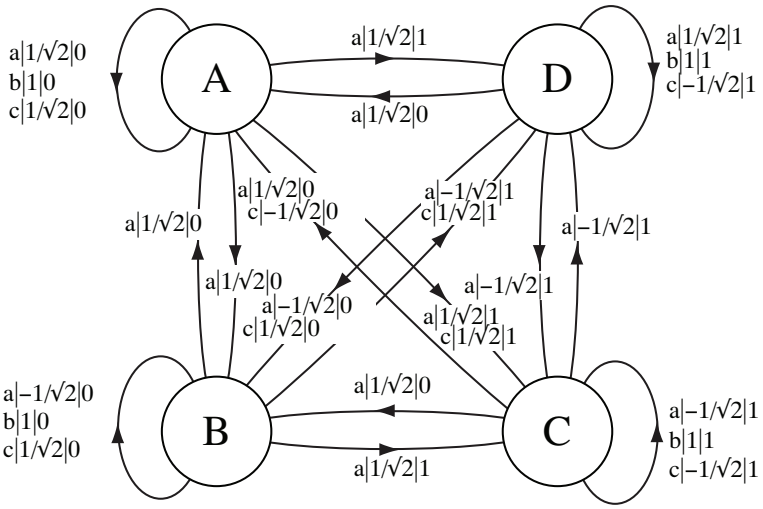

FIG. 20: Quantum machine for a trapped-ion system exposed to radiation of various wavelengths. The input alphabet $X = \{a, b, c\}$ and output alphabet $Y = \{0, 1\}$ represent unitary operations and electronic states, respectively.

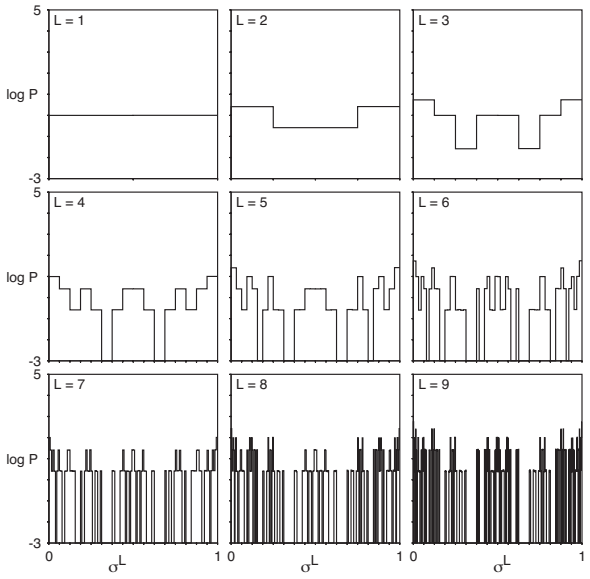

FIG. 21: Process language generated by the trapped-ion quantum dynamical system of Fig. 19 for protocol I (measurements performed at each time step).

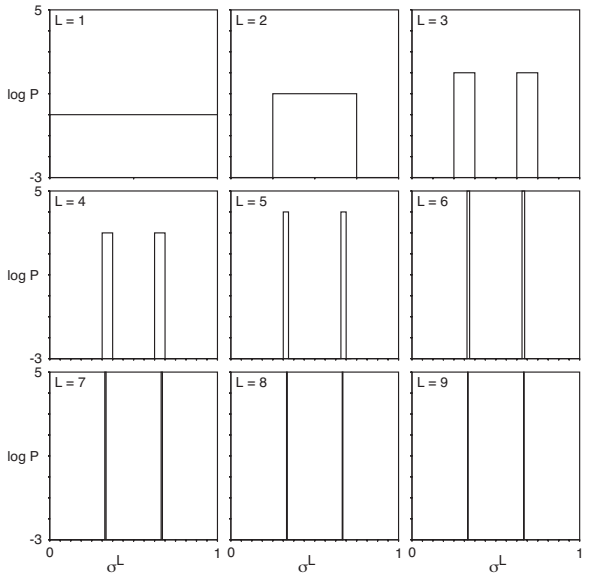

FIG. 22: The generated process languages of the trapped-ion dynamical system from Fig. 19 for measurements performed every three time steps.

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. [11, p. 32]. (Recall that in our notation $\langle\psi|$ is the state vector, not $|\psi\rangle$, as is common elsewhere.) Figure 23 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 II. The unitary operations H and U_f in Fig. 23 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 al-

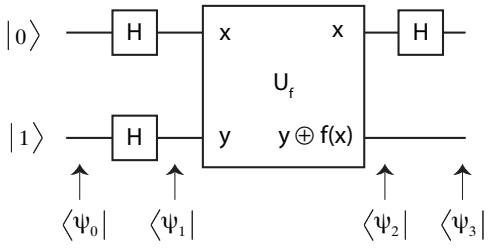

FIG. 23: Deutsch algorithm to classify balanced and constant functions ($N = 2$) depicted as a quantum circuit.

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 II: 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.

gorithm 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.

VIII. CONCLUDING REMARKS

We introduced quantum finite-state generators as a first step toward a computation-theoretic description of quantum dynamical processes. A quantum process is modeled by a quantum finite-state machine and its behavior is represented by the language it generates. This allowed us to build a computational hierarchy of finitary quantum processes.

Starting at the lowest level of Chomsky's hierarchy we discussed finite-state recognizers—machine's that determine whether or not a given sentence obeys the grammatical rules of a language. Translating that task into stochastic languages led to stochastic and then quantum finite-state recognizers. We extended these concepts from recognizing to generating devices. As far as we are aware, in the quantum setting, this has not been discussed before.

We laid out the mathematical foundations of these objects and developed a hierarchy of classical 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 QMs, 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 [69]. Typically, this requires substantially more resources, largely to store outcomes of intermediate steps. In short, reversibility does not imply less power for classical computers. At the end of the day computational resources are variables that trade-off against each other. The 3-state QDG examples of the Golden Mean and the Even processes illustrated such a trade-off. Although the QDG needs more states than the equivalent DG to generate the same process language, different measurement protocols yielded a new set of process languages—an aspect that makes QDGs more powerful than DGs.

An exception to the view that finite quantum machines are less powerful than their classical counterparts was found as one varied the measurement protocol. We saw the the language diversity of quantum systems is greater than classical analogs. This is notable since all experiment interacts with quantum systems through measurement,

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 quantum algorithms in terms of their information processing power and do so independent of particular physical implementations. Results on this will appear elsewhere.

In a complementary way, we showed that any quantum process can be investigated in terms of its intrinsic information processing power. A quantum finite-state machine can model the behavior of a quantum dynamical system. The example of the quantum kicked top, a simple quantum dynamical system, illustrated the construction procedure. It allowed for a formal language analysis of the system's behavior and led to insights into the behaviors the system can exhibit. Having both stochastic (nonquantum) and quantum finite-state machine models allows one to clearly identify which properties of a dynamical system are quantum mechanical and which are essentially classical and probabilistic.

A next step will be to use tools from information the-

ory and automata theory to define a measure of intrinsic computation inherent in quantum systems. One possible benefit would be methods to classify dynamical behavior in a hierarchy based on relative information storage and generation capacity. These could be used to elucidate the power of future quantum computational substrates. 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—could then be answered. This, we believe, will lead to a new and constructive view of quantum intrinsic computation. In any case, we hope that integrating quantum computation and quantum dynamics will receive further attention.

Acknowledgments

The authors thank R. D'Souza, C. Ellison, D. Feldman, J. Mahoney, I. Rumanov, M. Sánchez-Montañés, and C. Strelioff for helpful comments and discussions. This work was supported at the Santa Fe Institute 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. E. Moerner and M. Orritt. Illuminating single molecules in condensed matter. *Science*, 283:1670–1676, 1999.

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

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

[4] 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.

[5] X. Zhuang, H. Kim, M. J. B. Pereira, H. P. Babcock, and N. G. Walter. Correlating structural dynamics and function in single ribozyme molecules. *Science*, 296:1473–1476, 2002.

[6] A. Yildis, J. N. Forkey, S. A. McKinney, T. Ha, Y. E. Goldman, and P. R. Selvin. Myosin v walks hand-over-hand: single fluorophore imaging with 1.5-nm localization. *Science*, 300:2061–2065, 2003.

[7] T. Ha, T. A. Laurence, D. S. Chemla, and S. Weiss. Polarization spectroscopy of single fluorescence molecules. *J. Phys. Chem. B*, 103:6839–6850, 1999.

[8] H. P. Lu, L. Xun, and X. S. Xie. Single-molecule enzymatic dynamics. *Science*, 282:1877–1882, 1998.

[9] Y. Jia, A. Sytnik, L. Li, S. Vladimirov, and B. S. Cooperman amd R. M. Hochstrasser. Non-exponential kinetics of a single trna^{Phe} molecule under physiological conditions. *Proc. Natl. Acad. Sci. U.S.A.*, 94:7932–7936, 1997.

[10] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabushi, and H. J. Kimble. Measurement of conditional phase shifts for quantum logic. *Phys. Rev. Lett.*, 75:4710, 1995.

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

[12] I. L. Chuang and Y. Yamamoto. Simple quantum computer. *Phys. Rev. A*, 52:3489–3496, 1995.

[13] Claude Cohen-Tannoudji, Bernard Diu, and Franck Lalo e. *Quantum mechanics*. John Wiley & Sons, Singapore, 2005.

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

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

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

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

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

[19] 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.

[20] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74:4091–94, 1995.

[21] N. Gershenfeld and I. Chuang. Bulk spin-resonance quantum computation. *Science*, 275:350–6, 1997.

[22] B. Kane. A silicon-based nuclear spin quantum computer. *Nature*, 393:133–7, 1998.

[23] L. E. Mooij, T. P. Orlando, L. Levitov, L. Tian, C. H. van der Wal, and S. Lloyd. Josephson persistent-current qubit. *Science*, 285:1036–9, 1999.

[24] P. M. Platzman and M. I. Dykman. Quantum computing with electrons floating on liquid helium. *Science*, 284:1967–9, 1999.

[25] D. Leibfried, B. DeMarco, V. Meyer, D. Lucas, M. Barrett, J. Britton, W. M. Itano, and B. Jelenkovi. Experimental demonstration of a robust, high-fidelity geometric two ion-qubit phase gate. *Nature*, 422:412–5, 2003.

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

[27] A. M. Childs, I. L. Chuang, and D. W. Leung. Realization of quantum process tomography in nmr. *Phys. Rev. A*, 64:012314–20, 2001.

[28] J. Preskill. Reliable quantum computers. *Proc. R. Soc. Lond. A*, 454:385–410, 1998.

[29] E. Knill, R. Laflamme, and W. H. Zurek. Resilient quantum computation. *Science*, 279:342–5, 1998.

[30] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434:39–44, 2005.

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

[32] F. Haake. *Quantum signatures of chaos*. Springer, Berlin, 2001.

[33] E. Ott. *Chaos in dynamical systems*. Cambridge Press, 2002.

[34] N. L. Balazs and A. Voros. The quantized Baker's transformation. In G. Casati and B. Chirikov, editors, *Quantum chaos*. Cambridge University Press, 1995.

[35] M. E. Goggin, B. Sundaram, and P. W. Milonne. Quantum logistic map. *Phys. Rev. A*, 41:5705–5708, 1990.

[36] A. Peres. Instability of quantum motion of a chaotic system. In H. A. Ramaswamy, M. C. Gutzwiller, and G. Casati, editors, *Quantum chaos*, pages 73–102. World Scientific, Singapore, 1991.

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

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

[39] 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.

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

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

[42] P. E. Caines. *Linear Stochastic Systems*. Wiley, New York, 1988.

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

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

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

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

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

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

[49] 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.

[50] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines–part ii. *IEEE Trans. Patt. Anal. Mach. Intel.*, 27:1026–1039, 2005.

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

[52] 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.

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

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

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

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

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

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

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

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

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

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

[63] 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.

[64] 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.

[65] K. Wiesner and J. P. Crutchfield. Language diversity in measured quantum processes. *Intl. J. Unconventional Computation*, 2006. in press.

[66] C. Howson. Theories of probability. *The British Journal for the Philosophy of Science*, 46:1–32, 1995.

[67] F. Haake, M. Kús, and R. Scharf. Classical and quantum chaos for a kicked top. *Z. Phys. B*, 65:381–395, 1987.

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

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